# RNA-Seq for Gene Expression Analysis.

R. Raja Sekhara Reddy
Clevergene Biocorp Private Limited,
Bangalore, India.

**Table of contents**

## Abstract:

Next Generation sequencing has emerged as the method of choice to answer fundamental questions in biology. The massively Parallel sequencing technology for RNA-Seq analysis enables better understanding of gene expression patterns in model and non model organisms. Sequencing per se has reached the stage of commodity level while analysing and interpreting huge amount of data raise significant challenge. This chapter is aimed at discussing the complexities involved in sequencing and analysis, and tries to simplify sequencing based gene expression analysis. Biologists and experimental scientists were kept in mind while discussing the methods and analysis workflow.

*Key words: RNA, RNAseq, Transcriptome, NGS, Gene Expression*

## 1.1 Introduction:

The Next Generation Sequencing (NGS) or high throughput DNA sequencing methods have emerged as central to answering fundamental biological questions on a genome wide scale, setting forth a revolution in biology. Since the invention of DNA sequencing, the technique has proven vital for studying the genome organization, stability and in turn molecular understanding of traits and diseases. The technical superiority of NGS makes it an excellent first step analysis choice to answer fundamental questions in modern biology [1]. Applications of NGS include genome sequencing, gene expression and epigenome analysis [2]. Molecular level comparison between species aided by NGS based genome decoding facilitated better understanding of tree of life. The knowledge of gene conservation across species is providing insights to the molecular mechanisms of gene regulation.

RNA sequencing (RNA-Seq) is emerging as a standard research tool to address basic questions in biology such as cell cycle regulation, division and divergence. RNA-Seq is superior to microarray because it does not require prior sequence information of the organism and expression can be digitally quantified; it also detects the splicing patterns and post transcriptional modifications[3]. RNA-Seq can be used to analyse the whole transcriptome, including mRNA, ncRNA, smallRNA and it has facilitated gene regulation profiling of non model organisms like never before[4].

RNA-Seq can be performed using different NGS technologies such as pyro sequencing (Roche), sequencing by synthesis (Illumina), semiconductor sequencing (Ion torrent), single molecule real time sequencing (Pacific Biosciences) and nano pore sequencing (Oxford nanopore). However, Illumina's sequencing by synthesis technique is widely used for RNA-Seq because of the data quantity requirements $^{5}$.

## 1.2 Methods:

The NGS work flow can be broadly divided in to three major parts *viz*.

1. Library preparation
2. Sequencing and
3. Data analysis.

Each of these parts contain different steps depending on the research questions to be answered.

### 1.2.1 RNA-Seq Library preparation

Irrespective of the NGS technology, the first part in RNA-Seq experiment is library generation, which depends on the fraction of RNA (mRNA, smallRNA, trancriptome) to be investigated. Various commercial library preparation kits are available for different RNA-Seq applications. With the discovery of regulatory RNAs like small, micro and long-noncoding RNAs (sRNA, miRNA and lncRNA), recent RNA-Seq experiment are targeted towards whole transcriptomes; which can be used to quantitate mRNA and lncRNA transcripts.

First step of whole transcriptome library prep is depletion of rRNA since it constitutes 90-95% of cellular RNA content. Due to probe based design, efficiency of commercial rRNA depletion kits varies depending on the species. Irrespective of the kit used, 5-10% of the sequencing data would still contain rRNA reads. If an experiment's aim is to study the expression profile of protein coding genes/transcripts, enriching poly-A tail RNA molecules (in eukaryotes) is a flexible option.

The quality and quantity of the starting material is important to generate good sequencing libraries. Since library preparation is adapter ligation based, it is essential to quantify the sample accurately using a RNA binding dye and degradation of RNA should be checked on gel or using a fragment analysis method such as Caliper LabChip or Agilent Bioanalyzer/Tapestation.

Most of the commercially available kits include the following steps in RNA library preparation from mRNA or rRNA depleted samples.

1. Chemical fragmentation of RNA

   RNA will be randomly fragmented by utilizing divalent cations ($Mg2^+$) and heat (~95$^o$C).
2. First stand cDNA synthesis

   Random oligos (hexamers) are used to prime the reverse transcription.
3. Second Strand synthesis

   Depending on the requirement, the reaction mixture components would vary. For directional library preparation, dUTPs are used instead dTTPs, enabling quenching of 2nd strand during library enrichment to retain strand information of RNA. (The strand information is utilized to discriminating transcript reads in genomic regions where both strands transcribe to generate different transcripts.)
4. Ligate adapters

   The partial/full sequencing adapters are ligated in this step. When the full adapters are used, independent indexed (barcode) adapters are used for each sample.
5. Size selection

   Depending on the sequencing technology to be used, the size of the library may vary. For example Illumina sequencing the recommended size is (300-500 nt) whereas, 454 platform is known to generate longer reads (800-1000 nt).
6. Enrichment PCR

   Fragments with adapters on both the ends are enriched by PCR. The number of PCR cycles should be optimized depending on the input RNA quantity because increasing PCR cycles causes excessive PCR duplicates in the sequence data.
7. Quantity and quality check of library

   It is essential to check the size distribution of the library using Bioanalyzer/Tapestation as it is vital to calculate the molarity of the library. The Library quantity should be measured with the use of fluorescent dye based assay.

The smallRNA library preparation requires a different approach because of the size of these molecules (18-32nt). Total RNA (500ng -1ug) or enriched smallRNA (20-50ng) would be used with protocols which leverage the presence of 5’ phosphate and 3’OH in mature miRNA to ligate smallRNA specific adapters. Following steps are part of the smallRNA library preparation

1. Ligate adapter(s)

   The adapter(s) will be ligated to the smallRNA considering, only mature miRNAs contain 5’phosphate and 3’OH hence adaptor(s) will only be ligated to those molecules.
2. Reverse transcription

   The smallRNA ligated with adapter(s) will be reverse transcribed using complementary adapter oligos.
3. Enrich library

   Adapter ligated smallRNA molecules are enriched by PCR.
4. Size selection

   It is essential to size select the smallRNA library because of its proximity with the adaptor dimers (~137 nt). Polyacrylamide gels are used to excise the smallRNA library (~145 nt) to avoid primer and primer dimer contamination.
5. Quality and quantity check

   It is essential to check the size distribution of the library using Bioanalyzer/Tapestation as it is vital to calculate the molarity of the library. Library quantity should be measured with the use of fluorescent dye based assay.

The sequence data quality and experiment results are dependent on the quality of the library hence it is essential to generate and QC the sequencing library adhering to recommendations of the sequencing technology provider.

Some important Considerations for Library preparation

1. Target RNA fraction

   Depending on the RNA fraction to be studied, a suitable protocol should be selected. If information of transcription strand is important, stranded/directional RNA library kits should be used. Similarly, if the aim is to study small or miRNA an appropriate kit for smallRNA library preparation should be used.

2. Quality of RNA

   Samples with RNA integrity number (RIN) 7 or more, OD260/280 and OD260/230 close to 2 generate good quality libraries. In case of FFPE samples it is prudent to follow the kit recommendations for RNA QC.
3. Quantity check

   Always use fluorescent dye based assay to quantitate RNA and library. Most of the commercially available library preparation methods require 10ng – 1µg of RNA as starting material but if sample quantity is not a limitation it is better to start with at least 100ng of RNA. Elute/dissolve RNA in 30-50 µl of low TE buffer.
4. Barcoding/Indexing

   For multiplexing, libraries must be indexed with appropriate barcodes and while pooling compatibility of barcodes must be checked to avoid data loss or contamination by adapter hopping.
5. Handling

   Standard laboratory practices for RNA handling like, use of dedicated RNA working area, RNAse/Nuclease free plastic ware, water, RNAse inhibitor, barrier pipette tips and enzyme aliquots to avoid freeze thaw cycles etc. should be followed. To avoid carryover contamination pre and post PCR areas should be separated.

### 1.2.2 Sequencing

To reap best results from NGS data, one should design the experiment appropriately. Any biological experiment without replicates could not deliver insightful results, the same is applicable to RNA-Seq experiments as well. Most of the studies generate large number of reads per sample for an experimental condition, however, increasing number of reads may not provide articulate results [6,7]. The ENCODE consortium in 2011 [8] set up guidelines for RNA-Seq experiments with regards to the number of reads, read length, biological replicates for differential gene expression analysis.

**Important Considerations for sequence data generation**

1. Single or Paired end reads

   With reduction in the cost of sequencing data generation, most researchers are opting for paired end reads because of superior mapping potential of paired end reads. However, single end reads can still be used in model organisms when cost is a limitation.

2. Read length

   Using current sequence aligners, even short reads (<50 nt) can be mapped accurately for a model organism. For non modal organisms or organisms with repetitive regions in the genomes, 100bp or longer reads will increase the mappability. Similarly, *de novo* RNA-Seq experiments require paired reads to generate longer contigs and scaffolds.

3. Number of replicates

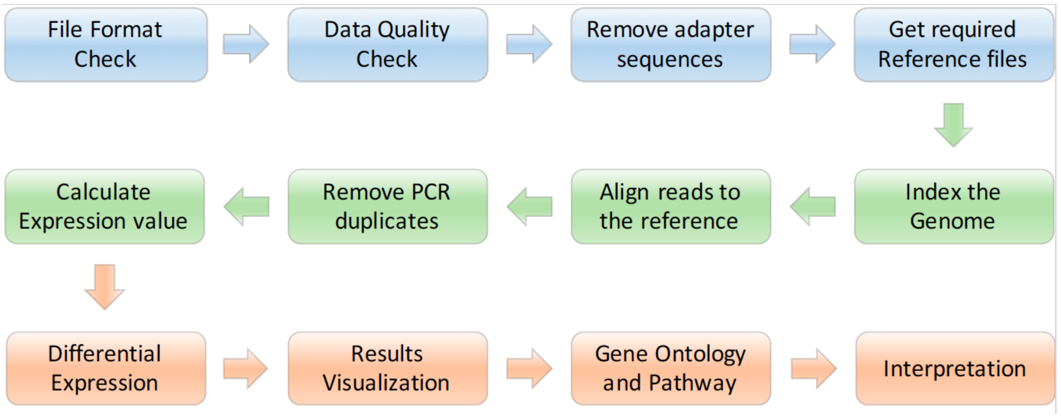
   Minimum, triplicates are recommended for each experimental condition. If high variability between replicates is expected, then increasing the number of replicates would provide better results.

4. Number of reads

   In a differential expression analyses, RNA-Seq data is considered for tag-counting. If the experiment's goal is to identify differential expression of highly expressing transcripts in a model organism, 20-30 million paired reads would deliver the results [910]. If the aim is to study the low expressed transcripts, then ~100 million reads or more would be needed. Whereas, *de novo* RNA-Seq requires relatively more reads to generate optimal transcriptome assembly. To determine the number of reads required, one can perform saturation analysis.

### 1.2.3 Data Analysis

The sequence data should be filtered based on quality and further analyzed to generate interpretable results. Many of the researchers in biology are not or vaguely familiar with the NGS data analysis workflows causing delay in interpreting results. Since most of the analysis programs are command line applications, may researchers face difficulty in using them, hence, in this chapter we discuss the data analysis process in step by step manner (Figure 2.1) using example data sets.



#### 1.2.3.1 Data format

To analyze any data, one should be familiar with format of the data. In NGS, most widely used raw read data format is FASTQ; these files contain read name, read sequence and quality value (of each base) of the read. A single fastq file can contain billions of reads. Once the sequencing run is completed the data will be de-multiplexed to separate sample specific reads and generate respective fastq files. If the data is not de-multiplexed, we need to use some software utilities to de-multiplex such files. A paired end sequence information is stored in two fastq files, one for forward (left) another for reverse (right). We will discuss about other formats of data as we proceed with the analysis.

*Practice*

Most of the Next generation sequence (NGS) data analysis tools were developed in unix/linux operating system environment and many of these software packages run in command line interface and utilize scripting languages like python, perl etc. Hence it is essential to use unix based operating systems like Mac OS X or linux.

To practice this data analysis, a computer with 4-8 GB RAM, 100 GB disk space and quad core processor is required. All the analysis mentioned in this chapter are performed on a MacBook Air. These analyses can also be performed using any computer with Unix based Operating System (like Linux, Biolinux [13], Ubuntu, RHL, Fedora, Linux Mint etc.) or virtual drive with any Linux distribution with enough RAM and disk space.

One can install linux in a computer with windows operating system by following the instructions given in the following website. Once linux is installed make sure to use it by some commands like ‘ls’, ‘mkdir’, ‘rm’ etc.
https://learn.microsoft.com/en-us/windows/wsl/install

This is right time to set few **ground rules for data analysis**

1. Always maintain a log of what command has been used and its result.
2. Do not give space in file and folder names instead – or _ in the file or folder names.
3. Do not use special characters other than – or _ in file and folder names.

4. Read every message shown at command prompt (terminal) as it will help in understanding the cause of error. Most of the times errors are caused due to spelling mistakes.
5. Before installing any software one should always read the system requirement and README file provided by the developers. This will helps us understand the complexities involved in installing a software.
6. Don't use administrative password to install each and every program.

## First steps in Linux

Once the system is ready with the OS, we need to learn navigating through the system. As mentioned earlier many of the NGS analysis tools needs to be executed through command line interface (CLI) hence it is essential to get familiarize with CLI. In linux and mac terminal is the CLI. Open terminal by pressing windows key in the key board or by clicking ubuntu launch button in OS and typing terminal in the search bar. Once the terminal is open we would like to know in which folder we are in by typing

```
pwd
```

The command returned the **present working directory** as /*home*/user*
*user is the user account through which you have logged in.

If we create any file or directory using the CLI now they will be present in the present working directory (pwd). Let's create a directory named Linux_learning

```
mkdir Linux_learning
```

To know if the directory is created or not we need to check the contents of the present working directory by listing all of them.

```
ls
```

In the present working directory there is only one folder names Linux_learning which was created earlier. Now change the working directory to Linux_learning

```
cd Linux_learning
```

'cd' is short form for change directory. Now check the contents of the present working directory by typing.

```
ls
```

There are no file in this directory to list. Lets create a directory named test.

```
mkdir test
ls
```

Now we can see test directory was created within our Linux_learning directory. To deleted the directory

```
rm -rf test
```

'rm' is short form for remove. Many commands come with multiple options which are usually specified using '-'. In the above command we are using r,f options which denote recursive and force respectively. A directory/folder can contain files and sub-folders so we are asking rm to remove such files and folders with 'test' and forcing it to delete the folder.

To know the available options of a command we can use linux manual command 'man'

```
man rm
```

The follow are the few lines of manual for rm command

```
RM(1)                    BSD General Commands Manual                    RM(1)

NAME
      rm, unlink -- remove directory entries

SYNOPSIS
      rm [-dfiPRrvW] file ...
      unlink file

DESCRIPTION
      The rm utility attempts to remove the non-directory type files specified
      on the command line.  If the permissions of the file do not permit writ-
      ing, and the standard input device is a terminal, the user is prompted
      (on the standard error output) for confirmation.

      The options are as follows:

      -d          Attempt to remove directories as well as other types of
                     files.
```

Try to know about ‘ls’ 'cp' and 'mv' commands using man. 'cp' is for copying files and folder from one location to other location in the system. 'mv' is move files and folders. Use these commands and see the difference between them.

## Tab to auto fill

The main limitation users face in CLI is spelling mistakes or typological errors. Pressing tab key after entering few characters of the command will auto fill the command and double pressing the tab key will list all options available with those starting letters. You can try it by creating a folder and changing the pwd to new folder using 'cd'.

Use tab key after entering 2 letters if the new folder. Using tab will reduce our time in typing the folder/file names and writing full commands and paths.
A path is address of a file in the system. For example we have created Linux_learning in home directory of the system. The exact path of the folder is /home/user*/Linux_learning . This is what we get when we type pwd in the terminal.
*the user

**How to install NGS tools**

The Linux distributions (distro) maintain their own package repository. A repository is an archive of different packages that are suitable to install on the operating system (OS). Each distro comes with a package manager. To download and install these packages from these repositories the user should have administrator privileges. Programs that are installed using OS package manager will be present in root folder of the OS. If we want to install different versions of the same software, there would be compatibility issues.
There are third party repositories which are easy to use, and they do not use the root folder for installing the packages. We can also install multiple versions of the same softwares in different environments. For NGS tools brew and conda are the well suited third party repositories. We can use both the repositories in the same system. However, it would be good to start with conda.

To install conda in your system
visit https://docs.conda.io/en/latest/miniconda.html and download conda for python 3.*. Follow the installation instructions.
**Important: Those woking on WSL (Linux in windows) and linux choose the Minicond3 Linux 64bit.**

Check installation of miniconda by typing in terminal where conda is activated `conda install -c bioconda fastp` to install fastp QC tool.

For data QC, most widely used application is FastQC. Try to install it by your own using conda.

***Using internet***
There are numerous online websites providing courses for Bioinformatics and NGS analysis. Many of them expect the user to have basic knowledge of Linux. To get this

basic knowledge you can use Internet, however one should follow few basic guidelines

1. Check the spelling of the command
2. Read the error message and try to understand it
3. While searching for solution over Internet don't copy the entire error message. The message may contain your system id or the folders you have created. So check the general message and use it to search the web.

For example:

```
ls /Users/reddy/*.hgrd
```

results in the following message

```
ls: /Users/reddy/*.hgrd: No such file or directory
```

In the above command I am looking for files with extension ".hgrd" in my home folder. If I copy the entire message "ls /Users/reddy/*.hgrd" to Google search, I will get some results but they may not be specific. Hence it is prudent to search for the generic terms line "ls: No such file or directory".

## Exercise

1. Look for Documents folder in your system and through terminal read that folder.
2. Create a folder named NGS_learning
3. Using text editor (gedit) create a file names "testfile.txt"in the NGS_folder.
4. Search Internet for to know "how to see the contents of a file in terminal".
5. Check Internet to know "how to add some text to a file".
6. Using 'man' learn about 'sed', 'cut' and paste commands
7. Read up about executlable paths in linux

## Little more Linux¶

In this section we will learn using some tools that are used widely in NGS. These are not NGS specific tools, however they are used in changing the data format in the files, filtering, search and replace etc. The tools are

- **cut**
- **paste**
- **sed**
- **grep**

- **awk**

cut

Cut is used to select columns in tsv,csv or any delimited files. It can also be used to remove few characters in all the columns of a text file. For practice purpose we will use iris data. To know more about the data you can check Wikipedia.

Copy the iris.tsv file to your pwd. Using the following command or just by visiting the link.

```
wget https://raw.githubusercontent.com/pavlov99/pmll/master/
pmll/datasets/iris.tsv
```

The file is in tab delimited format and you can check first few lines of the file in your CLI. It has 5 columns and each column has a name. The first line with column names is called header line.

```
head iris.tsv
```

```
Sepal.Length    Sepal.Width     Petal.Length    Petal.Width     Species
5.1     3.5     1.4     0.2     setosa
4.9     3       1.4     0.2     setosa
4.7     3.2     1.3     0.2     setosa
4.6     3.1     1.5     0.2     setosa
5       3.6     1.4     0.2     setosa
5.4     3.9     1.7     0.4     setosa
4.6     3.4     1.4     0.3     setosa
5       3.4     1.5     0.2     setosa
4.4     2.9     1.4     0.2     setosa
```

Now let us retain only first column

```
cut -f1 iris.tsv
```

```
Sepal.Length
5.1
4.9
4.7
4.6
5
5.4
4.6
5
4.4
4.9
5.4
4.8
4.8
4.3
5.8
```

If we want to retain first three columns

```
cut -f1-3 iris.tsv
```

```
Sepal.Length    Sepal.Width     Petal.Length
5.1     3.5     1.4
4.9     3       1.4
4.7     3.2     1.3
4.6     3.1     1.5
5       3.6     1.4
5.4     3.9     1.7
4.6     3.4     1.4
```

Now lets retain columns from 3 onward

```
cut -f3- iris.tsv
```

```
Petal.Length    Petal.Width     Species
1.4     0.2     setosa
1.4     0.2     setosa
1.3     0.2     setosa
1.5     0.2     setosa
1.4     0.2     setosa
1.7     0.4     setosa
1.4     0.3     setosa
1.5     0.2     setosa
1.4     0.2     setosa
```

Lets retain columns 1 to 3 and 5

```
cut -f1-3,5 iris.tsv
```

```
Sepal.Length    Sepal.Width     Petal.Length    Species
5.1     3.5     1.4     setosa
4.9     3       1.4     setosa
4.7     3.2     1.3     setosa
4.6     3.1     1.5     setosa
5       3.6     1.4     setosa
5.4     3.9     1.7     setosa
4.6     3.4     1.4     setosa
5       3.4     1.5     setosa
```

One important thing we need to know is cut assumes the fields in the file are tab de-limited. If we are working with a comma delimited file (csv) or a space delimited file we need to specify the delimiter using -d option. We can see how the -d option works by defining '.' as delimiter.

```
cut -d "." -f1 iris.tsv
```

```
Sepal
5
4
4
4
5       3
5
4
5       3
4
4
5
```

You can see, it has retained the values before first ".". If there is no "." in one of the rows it retained values until it finds a "." in that row. You can try cutting different columns with different delimiters.

**sed**

sed is used to replace file contents.

Lets replace the tabs in iris.tsv to commas.

```
sed 's|\t|,|g' iris.tsv|head
```

```
Sepal.Length,Sepal.Width,Petal.Length,Petal.Width,Species
5.1,3.5,1.4,0.2,setosa
4.9,3,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
4.6,3.1,1.5,0.2,setosa
5,3.6,1.4,0.2,setosa
5.4,3.9,1.7,0.4,setosa
4.6,3.4,1.4,0.3,setosa
5,3.4,1.5,0.2,setosa
4.4,2.9,1.4,0.2,setosa
```

The above options s indicates substitute tab (\t) with "," globally in the file. The out put of sed command was piped to head command using "|". The tabs are replaced with comma. Now using the same command create an iris csv file named iris.csv

```
sed -e $ 's|\t|,|g' iris.tsv > iris.csv
```

Check if iris.csv was created, if yes check first few lines of the iris.csv.

```
head iris.csv
```

```
Sepal.Length,Sepal.Width,Petal.Length,Petal.Width,Species
5.1,3.5,1.4,0.2,setosa
4.9,3,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
4.6,3.1,1.5,0.2,setosa
5,3.6,1.4,0.2,setosa
5.4,3.9,1.7,0.4,setosa
4.6,3.4,1.4,0.3,setosa
5,3.4,1.5,0.2,setosa
4.4,2.9,1.4,0.2,setosa
```

Check what happens if g was not given in 's|\t|,|g' or if we use 's|\t|,|1' and 's|\t|,|2'

```
sed -e $'s|\t|,|' iris.tsv|head
```

```
Sepal.Length,Sepal.Width        Petal.Length    Petal.Width     Species
5.1,3.5 1.4     0.2     setosa
4.9,3   1.4     0.2     setosa
4.7,3.2 1.3     0.2     setosa
4.6,3.1 1.5     0.2     setosa
5,3.6   1.4     0.2     setosa
5.4,3.9 1.7     0.4     setosa
4.6,3.4 1.4     0.3     setosa
5,3.4   1.5     0.2     setosa
4.4,2.9 1.4     0.2     setosa
```

It only replaced first first instance of tab (\t)

```
sed -e $'s|\t|,|1' iris.tsv|head
```

```
Sepal.Length,Sepal.Width        Petal.Length    Petal.Width     Species
5.1,3.5 1.4     0.2     setosa
4.9,3   1.4     0.2     setosa
4.7,3.2 1.3     0.2     setosa
4.6,3.1 1.5     0.2     setosa
5,3.6   1.4     0.2     setosa
5.4,3.9 1.7     0.4     setosa
4.6,3.4 1.4     0.3     setosa
5,3.4   1.5     0.2     setosa
4.4,2.9 1.4     0.2     setosa
```

When we use 's|\t|,|1', sed was replacing first instance of tab with comma. Now change the number 1 to 2 in 's|\t|,|1' and check the result.

**grep**

grep is used to find a pattern in the text file. Lets see how it works

```
grep "seto" iris.tsv
```

```
5.1     3.5     1.4     0.2     setosa
4.9     3       1.4     0.2     setosa
4.7     3.2     1.3     0.2     setosa
4.6     3.1     1.5     0.2     setosa
5       3.6     1.4     0.2     setosa
5.4     3.9     1.7     0.4     setosa
4.6     3.4     1.4     0.3     setosa
5       3.4     1.5     0.2     setosa
4.4     2.9     1.4     0.2     setosa
4.9     3.1     1.5     0.1     setosa
5.4     3.7     1.5     0.2     setosa
4.8     3.4     1.6     0.2     setosa
4.8     3       1.4     0.1     setosa
4.3     3       1.1     0.1     setosa
5.8     4       1.2     0.2     setosa
5.7     4.4     1.5     0.4     setosa
5.4     3.9     1.3     0.4     setosa
5.1     3.5     1.4     0.3     setosa
5.7     3.8     1.7     0.3     setosa
5.1     3.8     1.5     0.3     setosa
5.4     3.4     1.7     0.2     setosa
5.1     3.7     1.5     0.4     setosa
4.6     3.6     1       0.2     setosa
```

grep is case sensitive, if want it to ignore the case use -i and if we want it to consider the entire word then use -w.

```
grep -i "Seto" iris.tsv
```

```
5.1     3.5     1.4     0.2     setosa
4.9     3       1.4     0.2     setosa
4.7     3.2     1.3     0.2     setosa
4.6     3.1     1.5     0.2     setosa
5       3.6     1.4     0.2     setosa
5.4     3.9     1.7     0.4     setosa
4.6     3.4     1.4     0.3     setosa
5       3.4     1.5     0.2     setosa
4.4     2.9     1.4     0.2     setosa
4.9     3.1     1.5     0.1     setosa
5.4     3.7     1.5     0.2     setosa
```

```
4.8     3.4     1.6     0.2     setosa
4.8     3       1.4     0.1     setosa
```

**paste**

It is used to paste two files side bu side. If we want to re arrange columns in iris.tsv we will generate two or more files based on arrangement requirement and paste them in the order we would like.

```
cut -f1,3,4 iris.tsv > iris.cols.134.csv
```

```
cut -f5 iris.tsv > iris.col5.tsv
```

```
cut -f2 iris.tsv > iris.col2.tsv
```

```
paste iris.col5.tsv iris.col2.tsv iris.cols.134.csv > iris.rearrange.tsv
```

```
head iris.rearrange.tsv
```

```
Species Sepal.Width     Sepal.Length    Petal.Length    Petal.Width
setosa  3.5     5.1     1.4     0.2
setosa  3       4.9     1.4     0.2
setosa  3.2     4.7     1.3     0.2
setosa  3.1     4.6     1.5     0.2
setosa  3.6     5       1.4     0.2
setosa  3.9     5.4     1.7     0.4
setosa  3.4     4.6     1.4     0.3
setosa  3.4     5       1.5     0.2
setosa  2.9     4.4     1.4     0.2
```

awk

awk is a scripting language which is used for file manipulation. It can be used to extract and format text from the file. Lets extract few columns from iris.tsv.

```
awk '{FS="\t"; OFS="\t"}{print $5,$1}' iris.tsv
```

```
Species Sepal.Length
setosa  5.1
setosa  4.9
setosa  4.7
setosa  4.6
setosa  5
setosa  5.4
setosa  4.6
setosa  5
```

In the above command FS= field separator OFS= output field separator. $5 and 1$ represent column numbers in the file. Let us filter rows based on values

```
awk '{FS=",";OFS="\t"}{if($1 < 5) print}' iris.csv
```

```
4.9,3,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
4.6,3.1,1.5,0.2,setosa
4.6,3.4,1.4,0.3,setosa
4.4,2.9,1.4,0.2,setosa
4.9,3.1,1.5,0.1,setosa
4.8,3.4,1.6,0.2,setosa
4.8,3,1.4,0.1,setosa
4.3,3,1.1,0.1,setosa
4.6,3.6,1,0.2,setosa
4.8,3.4,1.9,0.2,setosa
```

```
4.7,3.2,1.6,0.2,setosa
4.8,3.1,1.6,0.2,setosa
```

We can use && for and || for or notations in awk if statement. awk is a full fledged programming language and has many functionalities. There are books available for awk programming.

***Download Fastq Data***

For practice we will use the data that consists of two commercially available RNA samples: Universal Human Reference (UHR) and Human Brain Reference (HBR). The UHR is total RNA isolated from a diverse set of 10 cancer cell lines. The HBR is total RNA isolated from the brains of 23 Caucasians, male and female, of varying age but mostly 60-80 years old.

Create a project folder "RNAseqPractice" from terminal/commandline.

```
mkdir RNAseqPractice
```

in this folder download the practice data using wget

```
cd RNAseqPractice
wget http://genomedata.org/rnaseq-tutorial/HBR_UHR_ERCC_ds_5pc.tar
```

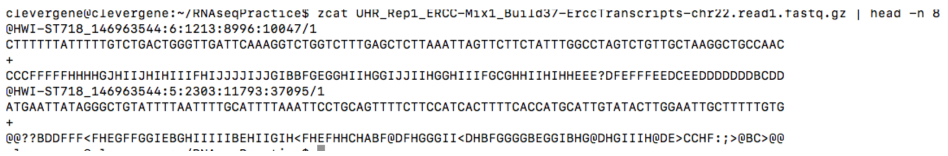
extract the downloaded data using using tar

```
tar -xvf HBR_UHR_ERCC_ds_5pc.tar
```

check the contents of the folder using ls

```
HBR_Rep1_ERCC-Mix2_Build37-ErccTranscripts-chr22.read1.fastq.gz

HBR_Rep1_ERCC-Mix2_Build37-ErccTranscripts-chr22.read2.fastq.gz
HBR_Rep2_ERCC-Mix2_Build37-ErccTranscripts-chr22.read1.fastq.gz
```



```
clevergene@clevergene:~/RNAseqPractice$ zcat UHR_Rep1_ERCC-Mix1_Build37-ErccTranscripts-chr22.read1.fastq.gz | head -n 8
@HWI-ST718_146963544:6:1213:8996:10047/1
CTTTTTATTTTTGTCTGACTGGGTTGATTCAAAGGTCTGGTCTTTGAGCTCTTAAATTAGTTCTTCTATTTGGCCTAGTCTGTTGCTAAGGCTGCCAAC
+
CCCFFFFFHHHHGJHIIJHIHIIIFHIJJJJIJJGIBBFGEGGHIIHGGIJJIIHGGHIIIFGCGHHIIHIHHEEE?DFEFFFEEDCEEDDDDDDDBCDD
@HWI-ST718_146963544:5:2303:11793:37095/1
ATGAATTATAGGGCTGTATTTTAATTTTGCATTTTAAATTCCTGCAGTTTTCTTCCATCACTTTTCACCATGCATTGTATACTTGGAATTGCTTTTTGTG
+
@@??BDDFFF<FHEGFFGGIEBGHIIIIIBEHIIGIH<FHEFHHCHABF@DFHGGGII<DHBFGGGGBEGGIBHG@DHGIIIH@DE>CCHF:;>@BC>@@
```

```
HBR_Rep2_ERCC-
Mix2_Build37-Er-
ccTranscripts-
chr22.read2.fastq.gz
HBR_Rep3_ERCC-Mix2_Build37-ErccTranscripts-chr22.read1.fastq.gz
HBR_Rep3_ERCC-Mix2_Build37-ErccTranscripts-chr22.read2.fastq.gz
UHR_Rep1_ERCC-Mix1_Build37-ErccTranscripts-chr22.read1.fastq.gz
UHR_Rep1_ERCC-Mix1_Build37-ErccTranscripts-chr22.read2.fastq.gz
UHR_Rep2_ERCC-Mix1_Build37-ErccTranscripts-chr22.read1.fastq.gz
UHR_Rep2_ERCC-Mix1_Build37-ErccTranscripts-chr22.read2.fastq.gz
UHR_Rep3_ERCC-Mix1_Build37-ErccTranscripts-chr22.read1.fastq.gz
UHR_Rep3_ERCC-Mix1_Build37-ErccTranscripts-chr22.read2.fastq.gz

HBR_UHR_ERCC_ds_5pc.tar
```

check first 8 line of the files using the following commands

```
zcat UHR_Rep1_ERCC-Mix1_Build37-ErccTranscripts-chr22.read1.fastq.gz | head -n 8
```

zcat will extract lines of text from the gz file and we piped ("|") it to head command to see first 8 lines of the file.

### 1.2.3.2 Data Quality Check

Due to vast amounts of the data, it is practically impossible to check quality of each base present in fastq file(s). Summarized quality parameters like quality value, read length, base, distribution across the reads and presence of adapter sequences and duplicated sequences would provide overall information of data quality. To understand these parameters, we need to understand what they represent.

**Quality value: The logarithmic probability of base calling error ($Q= -\log_{10} P$) [12]. To put it in perspective, Q value 30 means the probability of the base (nucleotide) being wrongly called is 0.001 and Q value 20 means probability of the base being wrongly called 0.01.**

Read length distribution: percent of reads with their respective lengths.

Nucleotide distribution: It visualizes how A, T, G, C are distributed across all the reads at a nucleotide position. If all the reads have same nucleotide at a given position it could be a sequencing artifact. If all the reads have same sequence towards their 3' end it could be adapter sequence.

Read Duplication: Few sequences representing majority of the data indicates presence of rRNA contamination or PCR duplicates.

Since we know data format and which quality parameters to check, let us analyze practice data set. This data is generated by sequencing a paired end stranded library using Illumina platform.

To check the data quality you can use fastqc and mutiqc. First create a conda environment named dataqc and activate it using the following commands.

```
conda create -n dataqc
conda activate dataqc
```

Once created install the fastqc and multiqc tools using the fillowing commands

```
conda install -c bioconda fastqc
conda install -c bioconda multiqc
```

Make a folder named ‘QC’ and analyze the fastq files and store the results in QC folder.

```
mkdir QC
fastqc -o QC -t 2 *.gz
```

The above fastqc command will analyze all the files that are ending with gz and stores the output in QC folder. The -t option utilizes two threads of the processor. Once finished, the analysis results can be checked for each fastq file. You can compile all the sample results in to one single QC report using the following command.

```
cd QC
multiqc .
```

In the above command ‘.’ represents with in the current folder. The command will create multiqc_report.html file and multiqc_data folder.

The QC results can be checked by opening the multiqc_report.html in an internet browser like chrome or firefox.

***Trimming and Filtering***

Although the overall data looks acceptable we need to remove adapter sequences and low quality bases (Q < 20) from the ends of the reads using “fastp”[15]. Create a folder name ‘2_TrimmedData’. Install fastp and quality filter the data by using default parameters. Once the trimming and filtering is complete, check the quality of the trimmed data using FastQC tool. Now our data is ready to be used for further analysis. Save these QC filtered fastq files to an analysis folder.

Install fastp in the dataqc conda environment and change your directory to where the fastq.gz files are locate and trim the paired end data using fastp using the following command.

```
conda install -c bioconda fastp
cd ../ (.. represents a folder above the current folder)
fastp -w 2 -g -x -i HBR_Rep1_ERCC-Mix2_Build37-ErccTranscripts-chr22.read1.fastq.gz -I
HBR_Rep1_ERCC-Mix2_Build37-ErccTranscripts-chr22.read2.fastq.gz -o
./2_TrommedReads/HBRRep1_Trimmed_R1.fastq.gz -O ./2_TrommedReads/HBRRep1_Trimmed_R1.fastq.gz
-h HBRRep1_fastp.html -j HBRRep1_fastp.json
```

repeat the command by changing the input and output files.

in the above command -i for read1 input, -I for read2 input, -o trimmed read1 output, -O trimmed read2 output, -w number of threads to use (you can increase it based on the processors), -x remove single nucleotide repeats like loA, -g to remove poly Gs (required for NovaSeq Data)

```
You can also use a for loop to process samples one by one automatically
```

```
to analyse HBR samples
for i in `ls *.read1.fastq.gz|cut -d "_" -f 1,2`; do fastp -w 2 -g -x -i $i"_ERCC-Mix2_Build37-ErccTranscripts-chr22.read1.fastq.gz" -I $i"_ERCC-Mix2_Build37-ErccTranscripts-chr22.read2.fastq.gz" -o "./2_TrommedReads/"$i"_Trimmed_R1.fastq.gz" -O "./2_TrommedReads/"$i"_Trimmed_R2.fastq.gz" -h $i"_fastp.html" -j $i"_fastp.json"

Try it yourself with UHR samples.
```

### 1.2.3.3 Getting Reference Sequence

Obtaining a genome version with proper annotations of exon, transcript, gene symbol, name, ontology etc is crucial for reference based RNA-Seq data analysis. The annotation information of a genome is stored in a separate file in specific format. There are two widely used file formats for annotation 1. GFF and 2. GTF, most of the applications used in RNA-Seq analysis would accept both of these formats. The Ensembl data base provides well annotated genomes for most of the sequenced organisms but for some organisms, dedicated web resources are available, which are more frequently updated as compared to ensemble database. For example, most widely used Rice genome (*Oryza sativa japonica*) is maintained by MSU Rice Genome Annotation Project team (http://rice.plantbiology.msu.edu/). Similarly PlasmoDB [16] maintains well annotated genomes of different Plasmodium strains. A literature search before downloading a reference genome would help in obtaining a good annotated genome.

*Practice*

Since our data is from Human samples, download the data from the following google drive link and place the downloaded fast and gtf files in your practice folder.
https://drive.google.com/drive/folders/1u4Xhn4rfe_lrLb_lmr9FKfNb9aHsePm9?usp=share_link

### 1.2.3.4 Indexing the Genome

The RNA-Seq reads can be mapped to transcriptome, nonetheless even for well studied species such as Human, Mouse we still may not know all transcripts, hence mapping the reads to the genome enables identification of novel transcripts and estimate their expression levels. Aligning reads to genome means, comparing the reads to the reference genome and finding a best match for each read but, NGS data contains billions of reads and mapping each read to the genome in a convectional manner (BLAST) requires humongous computational power and time. To answer this issue most NGS read mapping algorithms use Burrows–Wheeler transform (BWT) also known as block-sorting compression. In this approach the reference sequence will be transformed into small chunks of quick search compatible format which enable faster alignment of reads to the genome and this is know as “indexing”.

*Practice*

We have our quality filtered raw sequence data and well annotated genome, now we are ready to map (align) our sequence reads to the genome. Unzip the genome and annotation files to “GRCh38.chr22.fasta” and “GRCh38.chr22.gtf” respectively and copy them to the analysis folder (the folder where our QC timmed fastq files were saved). Change the directory to our analysis folder using command `cd`. The read mapping algorithm for RNA-Seq should be selected based on their ability to map reads generated from different splice forms of genes. In this chapter we use HISAT2 [17] due to its speed and low memory requirement.

Deactivate the dataqc environment using

```
conda deactivate
```

Create a conda environment named ‘RnaSeqSlignerrs’, activate it and install hisat2

```
conda create -n RnaSeqAligners
conda activate RnaSeqAligners
conda install -c bioconda hisat2
```

Index the genome by typing following command in terminal

```
hisat2-build –t 2 GRCh38.chr22.fasta GRCh38.chr22.fasta
```

`–t` to specify number of processor cores to be used.

This command will generate multiple files with “.ht2” extension. The indexing process will take considerable time (~1-2 Hrs) when we use complete human genome on a laptop depending on the configuration. We can generate reference index on any of the high end systems and copy it to any other computer. But in this practice we are using a small potion of chromosome 22 hence it will be quick.

### 1.2.3.5 Aligning reads

Once genome is indexed, read mapping is straight forward as most of the algorithms work fine with the default parameters. However, depending on the genome and NGS data, fine-tuning of read mapping by altering few parameters may improve overall results. Some times comparing algorithms would provide better insight into the results.

***Practice***

To start alignment, type the following command in the terminal window

```
hisat2 -p 4 --rna-strandness RF --dta -x GRCh38.chr22.fasta -1 HBRRep1_Trimmed_R1.fastq.gz -2 HBRRep1_Trimmed_R2.fastq.gz -S HBRRep1.sam
```

`-p` is to specify number of processors to use

`--rna-strandness` specifies whether the library is strand specific or not since our data was generated using stranded library we use `RF`.

`--dta` enables reporting of alignments that can be used to identify novel transcripts.

`-x` to specify the base name of index files

`-1` forward (left) reads file

`-2` reverse (right) reads file

`-S` out put alignment file in SAM format.

Once the alignment is complete following message will be shown in the terminal

```
454369 reads; of these:
  454369 (100.00%) were paired; of these:
    5533 (1.22%) aligned concordantly 0 times
    403439 (88.79%) aligned concordantly exactly 1 time
    45397 (9.99%) aligned concordantly >1 times
    ----
    5533 pairs aligned concordantly 0 times; of these:
      452 (8.17%) aligned discordantly 1 time
    ----
     5081 pairs aligned 0 times concordantly or discordantly; of these:
      10162 mates make up the pairs; of these:
        5239 (51.55%) aligned 0 times
        4213 (41.46%) aligned exactly 1 time
        710 (6.99%) aligned >1 times
99.42% overall alignment rate
```

This message denotes, our data contained 454369 reads and all of them were paired. 88.79% of our reads mapped uniquely to the genome and 9.99% of reads mapped at more than one genomic location and 1.22% reads did not map to the genome. The Sequence Alignment/Map (SAM) file contains 11 mandatory fields for essential alignment information such as genome and read name, mapping position, sequence, quality scores etc. for each alignment.

Now try to map other samples reads and save the alignment to respective sam files. The mapping quality can be assed by different parameters like gene body coverage and proportion of reads mapped to features (exons) etc. Use Qualimap2 [18] to assess the mapping quality.

### 1.2.3.6 Remove PCR Duplicates

We need to assess how many of the mapped reads originated from the same RNA molecule (PCR-duplicates) before calculating the expression levels. Computationally, read duplicates are defined by their mapping position, reads with same mapping position and length are considered as duplicates. There is no clear guideline on removing or retaining PCR duplicates [19], however if the PCR duplicates constitutes major fraction of the data, it is always good to compare the results with and without duplicates.

*Practice*

For this practice we use Picard tools [20] to estimate the fraction of PCR duplicates in the mapped reads. We need to sort SAM files by coordinate and convert them to Binary Alignment/Map (BAM) format as Picard takes sorted BAM format file as input. To work with SAM and BAM files we will use sambamba [21] hence install both programs by typing following commands in terminal.

```
conda install sambamba
```

To convert SAM file to BAM use following command

```
sambamba view -f bam -S sam -o HBRRep1.bam HBRRep1.sam
```

`-f` output file format

`-S` input file format

`-o` output file name

input file name

Now sort the bam file by coordinate

```
sambamba sort HBRRep1.bam
```

This command generates sorted bam file and index information of the bam file `HBRRep1.sorted.bam` and `HBRRep1.sorted.bam.bai` respectively. This bam file can be visualized using Integrative Genomics Viewer (IGV) [22].

### 1.2.3.7 Calculate expression values

In RNA-Seq experiments, gene/transcript expressions are measured by counting the reads mapped to its respective position in the genome. The expression values can be presented in different forms like read counts, RPM, RPKM, FPKM, TPM etc. If relative expression of a transcript with respect to other transcripts in a sample is to be measured, the RPM, RPKM, FPKM or TPM are used as the expression needs to be normalized. Whereas, to compare expression of two samples, using read counts is the better option.

One frequently asked question is "Should we use all mapped reads or only uniquely mapped reads?" to estimate the expression levels. Some genomes, specially plants contain high repetitive regions and many of these repetitive regions contain genes or pseudo genes. In such genomes half of the reads may be mapped to multiple locations. In such instances, it is better to use algorithms which can assign counts to both the features. Similarly, if a read is overlapping more than one feature it would be worth while to assign a count to both the features. For better understanding, one can always compare results with and without multimapped and overlapped reads for expression analysis.

*Practice*

For this analysis we will use featureCounts [23] which is a part of subread application and can be installed using conda. Type the following command to get gene level summarized read counts. First change the directory to 2_TrimmedReads

```
conda install -c bioconda subread
featureCounts -p -T 4 -M -O -a GRCh38.chr22.gtf -o gene.expression.txt *.bam
```

`-p` for paired end reads

`-T` number of processors to use

`-M` to consider multimapped reads

`-O` to consider overlapping reads

`-a` genome annotation file

`-o` output filename

*.bam

The output file `gene.expression.txt` is a tab delimited text file which can be opened with a spreadsheet application. Transcript level expression can be calculated by providing an extra option `-g "transcript_id".`

**Identification of novel transcripts**

In the previous practice we used reference annotation to quantify expression of known genes and transcripts. Using stringTie [24] program we can assemble novel transcripts in genome guided or *de novo* mode. Install stringTie and identify novel transcripts in reference guided mode by following steps

```
conda install -c bioconda stringtie
stringtie -G GRCh38.chr22.gtf -o HBRRep1.transcripts.gtf  HBRRep1.sorted.bam
stringtie -G GRCh38.chr22.gtf -o HBRRep2.transcripts.gtf HBRRep2.sorted.bam
```

`-G` annotation file to be used as gude

`-o` output GTF file name

**This command needs to be run for each bam file**

stringTie assigns arbitrary transcript IDs to each assembled transcript, therefore each GTF files may have different set of transcripts. There may be similarities between GTF files but the number of transcripts and their exact structure will differ in the output files for each sample. One solution for this problem is to merge the GTF files and use it for expression quantification using stringTie merge option.

```
stringtie --merge -o meged.gtf -G GRCh38.chr22.gtf *. transcripts.gtf
```

The "merged.gtf" can be used with featureCounts to generate transcript level summarized read counts into "merged.transcripts.txt".

### 5.2.3.8 Differential expression

Various statistical methods are available for differential expression of RNA-Seq; they vary in data normalization and distribution model considerations. Algorithms which use negative binomial distribution like DESeq [25] and edgeR [26] are considered to be sensitive and specific. However, using more than one differential expression analysis method would provide better results in detecting true positives. Most of the DE analysis packages use R programing language as base. To use them knowledge of R is essential. How ever in recent times shiny applications have facilitated the GUI for these DE packages one of such application is “DEBrowser”. We will use for the DE analysis and visualization.

*Practice*

R is an open source language and environment for statistical computing and graphics. Availability of number of packages for different applications makes R one of the widely used statistical program in the field of genomics. Though R is a command line driven program, availability of Integrated Development Environments like RStudio makes use of R relatively easy. So we will install R and RStudio by downloading the installers.
https://cran.r-project.org/
https://www.rstudio.com/products/rstudio/download/
Open RStudio and install DEBrowser by following the instructions in
https://www.bioconductor.org/packages/release/bioc/vignettes/debrowser/inst/doc/DEBrowser.html

**Prepare your expression data for DEBrowser input using a spread sheet application like libre office calc, excel etc.**

**1.2.3.9 Gene Ontology and Pathway analysis**

The Gene Ontology (GO) describes gene function and its classification based on their function. GO information is mainly used for enrichment analysis of gene sets that are up or down regulated under certain conditions. Gene set enrichment analysis will find which GO terms are over-represented (or under-represented). Similarly, pathway enrichment analysis will identify the major pathways in which these genes are involved.

*Practice*

Within R, there are many packages to perform gene set enrichment analysis but, due to ease, we will use Database for Annotation, Visualization and Integrated Discovery (DAVID) [27]. Open https://david.ncifcrf.gov/ in web browser and select functional annotation.

1. Copy and paste ids of over expressed transcripts in upload gene list box.
2. Select the identifier "ENSEMBL_TRANSCRIPT_ID" from drop down menu. Select "Gene list" and submit the list (Figure 2.7).
3. Some of our transcript ids will not be mapped because they were specific to our experiment hence in the next page click on "Continue to Submit the IDs That DAVID Could Map" (Figure 2.8).
4. In next page (Figure 2.9) select functional annotation chart.
5. Clear all default selections in next page and in Gene Ontology select GOTERM_BP_DIRECT, GOTERM_CC_DIRECT and GOTERM_CC_DIRECT. Similarly select KEGG_PATHWAY in Pathways and click on Functional Annotation Chart button.
6. Results will open in a separate browser window and clicking on download file will open a text file in web browser. We can copy and paste the data in a text editor or spreadsheet application.

You may try performing gene set enrichment analysis for under expressed transcripts as well.

### 1.2.4 *de novo* Transcriptome Analysis

One of the major advantages of RNA-Seq is its capability to study expression profile of organism for which reference genome or transcriptome are not available. Analyzing such RNA-Seq data contain two extra steps compared to reference based analysis afore mentioned. 1. Assembling Transcriptome and 2. Annotating assembled transcripts. Once an annotated transcriptome is available, one can follow the reference based approach for differential expression analysis.

There are various *de novo* transcriptome assemblers [28–30], most of them work on the same principle "De Bruijn graph" [31]. According this method method, the reads are broken into small K-mers and the overlapping K-mers are collapsed to make contigs. It is important to select the right K-mer length to get optimal assembly, hence generating assemblies with different K-mer length and comparing them is widely practiced [32]. The *de novo* transcriptome needs to be validated before using it as reference for differential expression analysis. Number of contigs, distribution of contig length and how many of them are annotated will provide basic impression of how well the transcriptome is assembled [33]. In addition, what proportion of raw reads get mapped onto the transcriptome also provides an idea of assembled transcriptome. The contigs are scaffolded with the paired end read information, as one of the paired reads are mapped to one contig and the other to another contig, these two contigs are stitched together. The scaffolds are annotated by BLASTing them against protein and non coding RNA databases like Uniref90, NCBI non redundant (nr) protein database using BLAST or similar tools and the annotated contigs are termed as Unigenes.

*Practice*

This would require a system with high configuration so do not do it in your laptop. Using brew install "trinity" *de novo* transcriptome assembler using conda and generate *de novo* transcriptome from our practice data by default parameters. Scaffold the trinity contigs using SSPACE [34] default parameters. Check how many reads are mapping onto the scaffolds using HISAT2. Annotate scaffolds using Blast2GO [35] and export the annotation information into a tab delimited text file. Create a GTF file using the Blast2GO output and use it as reference annotation for scaffolds to perform differential expression analysis.

In conclusion, RNA-Seq is by far the best method available to study gene expression in model and non model organism. Success of any study depends on the experiment design and right amount of data. Careful selection of right tools from the library preparation to differential expression analysis would provide great insights into gene expression and functional profile of the study organism.

**References**

1. Mardis, E. R. Next-generation DNA sequencing methods. *Annu. Rev. Genomics Hum. Genet.* **9,** 387–402 (2008).
2. Buermans, H. P. J. & den Dunnen, J. T. Next generation sequencing technology: Advances and applications. *Biochim. Biophys. Acta BBA - Mol. Basis Dis.* **1842,** 1932–1941 (2014).
3. Koboldt, D. C., Steinberg, K. M., Larson, D. E., Wilson, R. K. & Mardis, E. The Next-Generation Sequencing Revolution and Its Impact on Genomics. *Cell* **155,** 27–38 (2013).

4. Mutz, K.-O., Heilkenbrinker, A., Lönne, M., Walter, J.-G. & Stahl, F. Transcriptome analysis using next-generation sequencing. *Curr. Opin. Biotechnol.* **24,** 22–30 (2013).
5. Mardis, E. R. Next-generation sequencing platforms. *Annu. Rev. Anal. Chem. Palo Alto Calif* **6,** 287–303 (2013).
6. Manga, P. *et al.* Replicates, Read Numbers, and Other Important Experimental Design Considerations for Microbial RNA-seq Identified Using Bacillus thuringiensis Datasets. *Front. Microbiol.* **7,** (2016).
7. Schurch, N. J. *et al.* How many biological replicates are needed in an RNA-seq experiment and which differential expression tool should you use? *RNA* **22,** 839–851 (2016).
8. Rosenbloom, K. R. *et al.* ENCODE data in the UCSC Genome Browser: year 5 update. *Nucleic Acids Res.* **41,** D56-63 (2013).
9. Sims, D., Sudbery, I., Ilott, N. E., Heger, A. & Ponting, C. P. Sequencing depth and coverage: key considerations in genomic analyses. *Nat. Rev. Genet.* **15,** 121–132 (2014).
10. Conesa, A. *et al.* A survey of best practices for RNA-seq data analysis. *Genome Biol.* **17,** 13 (2016).
11. Afgan, E. *et al.* The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. *Nucleic Acids Res.* **44,** W3–W10 (2016).
12. Ewing, B. & Green, P. Base-calling of automated sequencer traces using phred. II. Error probabilities. *Genome Res.* **8,** 186–194 (1998).
13. Field, D. *et al.* Open software for biologists: from famine to feast. *Nat. Biotechnol.* **24,** 801–803 (2006).
14. Andrews, S. FastQC A Quality Control tool for High Throughput Sequence Data. *http://www.bioinformatics.babraham.ac.uk/projects/fastqc/* Available at:

http://www.bioinformatics.babraham.ac.uk/projects/fastqc/. (Accessed: 29th June 2016)

15. Shifu Chen, Yanqing Zhou, Yaru Chen, Jia Gu; fastp: an ultra-fast all-in-one FASTQ preprocessor, Bioinformatics, Volume 34, Issue 17, 1 September 2018, Pages i884–i890, https://doi.org/10.1093/bioinformatics/bty560
16. Bahl, A. *et al.* PlasmoDB: the Plasmodium genome resource. A database integrating experimental and computational data. *Nucleic Acids Res.* **31,** 212–215 (2003).
17. Kim, D., Langmead, B. & Salzberg, S. L. HISAT: a fast spliced aligner with low memory requirements. *Nat. Methods* **12,** 357–360 (2015).
18. Okonechnikov, K., Conesa, A. & García-Alcalde, F. Qualimap 2: advanced multi-sample quality control for high-throughput sequencing data. *Bioinforma. Oxf. Engl.* **32,** 292–294 (2016).
19. Parekh, S., Ziegenhain, C., Vieth, B., Enard, W. & Hellmann, I. The impact of amplification on differential expression analyses by RNA-seq. *Sci. Rep.* **6,** (2016).
20. Picard Tools - By Broad Institute. Available at: http://broadinstitute.github.io/picard/. (Accessed: 31st January 2017)
21. Tarasov, A., Vilella, A. J., Cuppen, E., Nijman, I. J. & Prins, P. Sambamba: fast processing of NGS alignment formats. *Bioinformatics* **31,** 2032–2034 (2015).
22. Thorvaldsdóttir, H., Robinson, J. T. & Mesirov, J. P. Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. *Brief. Bioinform.* **14,** 178–192 (2013).
23. Liao, Y., Smyth, G. K. & Shi, W. featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. *Bioinforma. Oxf. Engl.* **30,** 923–930 (2014).

24. Pertea, M. *et al.* StringTie enables improved reconstruction of a transcriptome from RNA-seq reads. *Nat. Biotechnol.* **33,** 290–295 (2015).

25. Anders, S. & Huber, W. Differential expression analysis for sequence count data. *Genome Biol.* **11,** 1–12 (2010).

26. Robinson, M. D., McCarthy, D. J. & Smyth, G. K. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* **26,** 139–140 (2010).

27. Huang, D. W., Sherman, B. T. & Lempicki, R. A. Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. *Nat. Protoc.* **4,** 44–57 (2009).

28. Grabherr, M. G. *et al.* Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nat. Biotechnol.* **29,** 644–652 (2011).

29. Xie, Y. *et al.* SOAPdenovo-Trans: de novo transcriptome assembly with short RNA-Seq reads. *Bioinformatics* **30,** 1660–1666 (2014).

30. Liu, J. *et al.* BinPacker: Packing-Based De Novo Transcriptome Assembly from RNA-seq Data. *PLOS Comput. Biol.* **12,** e1004772 (2016).

31. Clarke, K., Yang, Y., Marsh, R., Xie, L. & Zhang, K. K. Comparative analysis of de novo transcriptome assembly. *Sci. China Life Sci.* **56,** 156–162 (2013).

32. Durai, D. A. & Schulz, M. H. Informed kmer selection for de novo transcriptome assembly. *Bioinforma. Oxf. Engl.* **32,** 1670–1677 (2016).

33. Smith-Unna, R., Boursnell, C., Patro, R., Hibberd, J. M. & Kelly, S. TransRate: reference-free quality assessment of de novo transcriptome assemblies. *Genome Res.* **26,** 1134–1144 (2016).

34. Boetzer, M., Henkel, C. V., Jansen, H. J., Butler, D. & Pirovano, W. Scaffolding pre-assembled contigs using SSPACE. *Bioinformatics* **27,** 578–579 (2011).

35. Conesa, A. *et al.* Blast2GO: a universal tool for annotation, visualization and analysis in functional genomics research. *Bioinformatics* **21,** 3674–3676 (2005).