



Comparative Gene Expression Study in the *Arabidopsis thaliana* root apex using RNA-seq and microarray transcriptome profiles [RNA-seq]

24.05.2021

Nucleotide sequence analysis

Doroteja Vujinović & Tanya Deniz Toluay

Part 1

Info about research and sequencing dataset

1. What was the aim of the study, described in the scientific article?

The article "Comparing RNA-Seq and microarray gene expression data in two zones of the *Arabidopsis* root apex relevant to spaceflight" aims to examine tiny root sections for transcriptome analysis to compare the RNA sequencing (RNA-Seq) to microarrays in characterizing genes that are relevant to spaceflight.

2. Provide some info about the dataset you will work with and which sequencing technology was used.

Our group is working on the dataset SRX4190906 for GSM3182000. It is from the organism *Arabidopsis thaliana*, root tissue sampled from plants that were grown for 8 days. Total RNA was extracted from these frozen samples using Picopure RNA isolation kit (Arcturus®, Life technologies). RNA concentration was determined on a Qubit (Thermo Fisher/Life Technologies, Inc) and sample quality was assessed using the Agilent 2100 Bioanalyzer (Agilent Technologies, Inc.).

3. Which kit was used for sequencing library preparation? Does this kit preserve strand information (stranded library) or not?

Five ng of total RNA was used for cDNA library construction using Clontech SMART-Seq V3 ultra-low input RNA kit for sequencing (Clontech Laboratories, Inc.) according to manufacturer's protocol. The reverse transcriptase then switches templates and continues transcribing to the end of the oligonucleotide, resulting in full-length cDNA that contains an anchor sequence that serves as a universal priming site for second-strand synthesis. cDNA was amplified with primer II A for 10 PCR cycles. Then Illumina sequencing libraries were generated with 120 pg of cDNA using Illumina Nextera DNA Sample Preparation Kit per manufacturer's instructions. Libraries were quantified by Bioanalyzer and qPCR. Finally, the libraries were pooled by equal molar concentration and sequenced by Illumina 2X150 NextSeq 500. In preparation for sequencing, barcoded libraries were sized on the bioanalyzer, quantitated

by QUBIT and qPCR. This “working pool” was used as input in the NextSeq500 instrument sample preparation protocol. Typically, a 1.3 pM library concentration resulted in optimum clustering density in our instrument. Samples were sequenced on a single flowcell, using a 2x150 cycles (paired-end) configuration.

4. What is the advantage of stranded mRNA library preparation compared to a non-stranded library?

Stranded mRNA library preparation provides more accurate estimates compared to the non-stranded mRNA library preparation. That is why it is also more recommended to use a stranded mRNA approach. The reason stranded is better than non-stranded is that we can tell which strand our RNA is being transcribed from. When using the non-stranded protocol, it's not that we discard 50% of our data - it's that the sequence we see in the read may correspond to either the 1st or 2nd cDNA strand and we have no idea which one it is.

Downloading the data

After downloading the toolkit, we have set configurations with the command:

```
vdb-config -i
prefetch SRR7287381
fastq-dump --split-3 SRR7287381
fastq-dump --split-3 SRR7287381 --origfmt --skip-technical
--read-filter pass --clip -M 25 SRR7
```

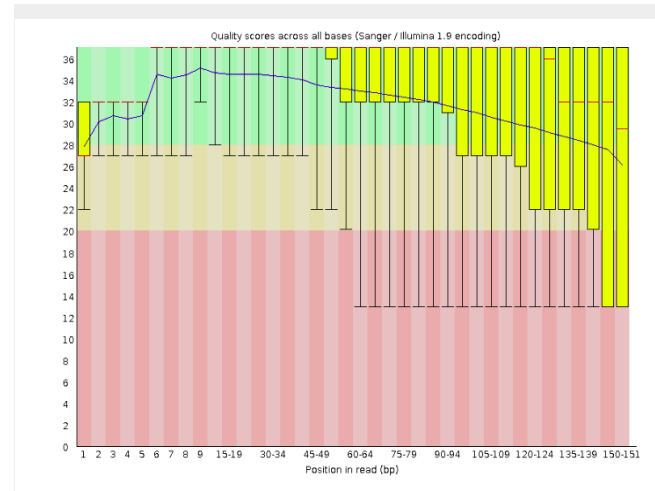
`--split-files` creates a file for every read, meaning the output will have a lot of files, whereas `--split-3` is for paired-end reads, so it will produce 2 fastq files only

Checking the quality of the sequences

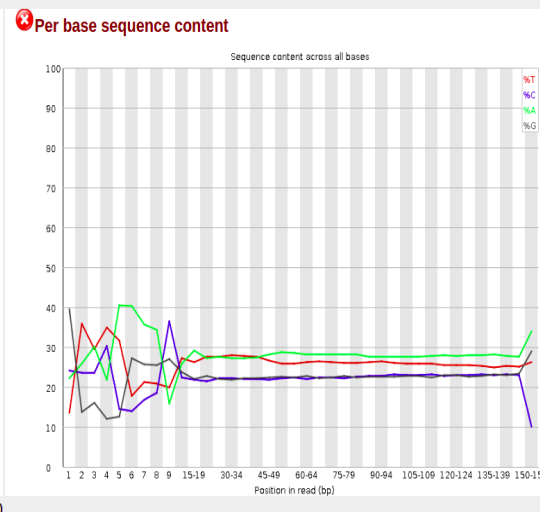
Run FastQC over your dataset. Explain the results.

Basic Statistics

Measure	Value
Filename	SRR7287381_1.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	5047876
Sequences flagged as poor quality	0
Sequence length	35-151
%GC	45



Given the quality score of a base is greater than 20, for most of our sequences, and lie in green and orange parts where are the sequences of very good and reasonable quality, respectively, we can say that our data is good regarding the base quality values. A value of 20 gives us the probability of 99% that our base call was made correctly. The quality of calls on most platforms will degrade as the run progresses, so the quality of base calls is falling below 20. This is quite common because chemicals used for sequencing are being used up.



There is a failure in per base sequence content. This module will fail if the difference between A and T, or G and C is greater than 20% in any position. This can be due to insertion bias from the Nextera kit used to prepare sequencing libraries.

We have a warning in “Per sequence GC content”. This module measures the GC content across the whole length of each sequence in a file and compares it to a modeled normal distribution of GC content. A warning is raised if the sum of the deviations from the normal distribution represents more than 15% of the reads.

Warnings in this module usually indicate a problem with the library. In our case, we have sharp peaks on an otherwise smooth distribution, and they are normally the result of a specific contaminant (adapter dimers for example), which may well be picked up by the overrepresented sequences module.

We also have a warning for “Sequence length distribution”, but this module will raise a warning if all sequences are not the same length. In our case, it is entirely normal to have different read lengths so warnings here can be ignored.

Another warning is “Sequence duplication levels”, which issues a warning if non-unique sequences make up more than 20% of the total. The underlying assumption of this module is of a diverse unenriched library. Any deviation from this assumption will naturally generate duplicates and can lead to warnings or errors from this module.

The last warning we had is in “Overrepresented sequences”. This module will issue a warning if any sequence is found to represent more than 0.1% of the total. In our case, the over-represented sequences might be adapters.

Despite these warnings, our dataset can be considered good.

Part 2

Filtering based on quality parameters, trimming, etc.

Trimmomatic:

```
$ trimmomatic PE SRR7287381_1.fastq SRR7287381_2.fastq
SRR7287381_1_paired.fastq SRR7287381_1_unpaired.fastq
SRR7287381_2_paired.fastq SRR7287381_2_unpaired.fastq
ILLUMINACLIP:NexteraPE-PE.fa:2:30:10:2:keepBothReads
LEADING:3 TRAILING:3 MINLEN:36 SLIDINGWINDOW:4:15
```

Measure	Value
Filename	SRR7287381_1_paired.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	4985238
Sequences flagged as poor quality	0
Sequence length	1-151
%GC	46

We can see from basic statistics, that number of total sequences decreased. Also, some bases were removed, the end of the read was improved due to a sliding window. Using Nextera with adaptor sequences we manage to remove overrepresented parts of the

sequences.

Alignment using STAR

1. Examine GTF/GFF file. Which information can be found in these files?

The GFF (General Feature Format) format consists of one line per feature, each containing 9 columns of data, plus optional track definition lines. The GTF (General Transfer Format) is identical to GFF version 2. Fields must be tab-separated. Also, all but the final field in each feature line must contain a value; "empty" columns are denoted with a '.'

1. seqname - the name of the chromosome or scaffold;
2. source - the name of the program that generated this feature, or the data source (database or project name)
3. feature - feature type
4. start - Start position* of the feature, with sequence numbering starting at 1.
5. end - End position* of the feature, with sequence numbering starting at 1
6. score - A floating-point value
7. strand - defined as + (forward) or - (reverse)
8. frame - One of '0', '1' or '2'. '0' indicates that the first base of the feature is the first base of a codon, '1' that the second base is the first base of a codon, and so on
9. attribute - A semicolon-separated list of tag-value pairs, providing additional information about each feature.

```

1 #!genome-build TAIR10
2 #!genome-version TAIR10
3 #!genome-date 2008-04
4 #!genome-build-accession GCA_000001735.1
5 #!genebuild-last-updated 2010-09
6 1 araport11 gene 10942648 10944727 . - . gene_id "AT1G30814"; gene_source "araport11";
  gene_biotype "protein_coding";
7 1 araport11 transcript 10942648 10944727 . - . gene_id "AT1G30814"; transcript_id
  "AT1G30814.1"; gene_source "araport11"; gene_biotype "protein_coding"; transcript_source "araport11"; transcript_biotype
  "protein_coding";
8 1 araport11 exon 10944317 10944727 . - . gene_id "AT1G30814"; transcript_id "AT1G30814.1";
  exon_number "1"; gene_source "araport11"; gene_biotype "protein_coding"; transcript_source "araport11"; transcript_biotype
  "protein_coding"; exon_id "AT1G30814.1.exon1";
9 1 araport11 exon 10944078 10944229 . - . gene_id "AT1G30814"; transcript_id "AT1G30814.1";
  exon_number "2"; gene_source "araport11"; gene_biotype "protein_coding"; transcript_source "araport11"; transcript_biotype
  "protein_coding"; exon_id "AT1G30814.1.exon2";

```

2. What is the difference between GTF/GFF2/GFF3 file formats?

The GFF and GTF formats are used for annotating genomic intervals (an interval with begin/end position on a contig/chromosome). The GFF (General Feature Format) format consists of one line per feature, each containing 9 columns of data, plus optional track definition lines. The GTF (General Transfer Format) is identical to GFF version 2. GFF2 has several shortcomings compared to GFF3. GFF2 can only represent 2 level feature hierarchies, while GFF3 can support arbitrary levels. GFF2 also does not require that column 3, the feature type, be part of the sequence ontology. It can be any string.

3. Perform 2-pass mapping

Before using STAR, a reference genome must be built using STAR's `genomeGenerate` mode. This requires a genome fasta file and GTF/GFF reference annotation. This can be achieved with the following command:

```

$ STAR --runThreadN 4 --runMode genomeGenerate --genomeDir .
--genomeFastaFiles Arabidopsis_thaliana.TAIR10.dna.toplevel.fa
--sjdbGTFfile Arabidopsis_thaliana.TAIR10.51.gtf --sjdbOverhang 149
--genomeSAindexNbases 12

```

After we have built a genome for STAR, we can proceed to align:

```

$ STAR --runThreadN 4 --genomeDir . --readFilesIn
SRR7287381_1_paired.fastq SRR7287381_2_paired.fastq
--outSAMattrRGline ID:SRR7287381 --outFileNamePrefix STAR_mapped_

```

Part 3

Converting SAM to BAM format and sorting sequences according to the genomic coordinated

```

$ STAR --runThreadN 4 --genomeDir . --readFilesIn
SRR7287381_1_paired.fastq SRR7287381_2_paired.fastq

```

```
--outSAMattrRGline ID:SRR7287381 --outFileNamePrefix
STAR_mapped_SRR7287381_ --outSAMtype BAM SortedByCoordinate --outStd
Log
```

We can also convert SAM to BAM file format using Samtools with command like this:

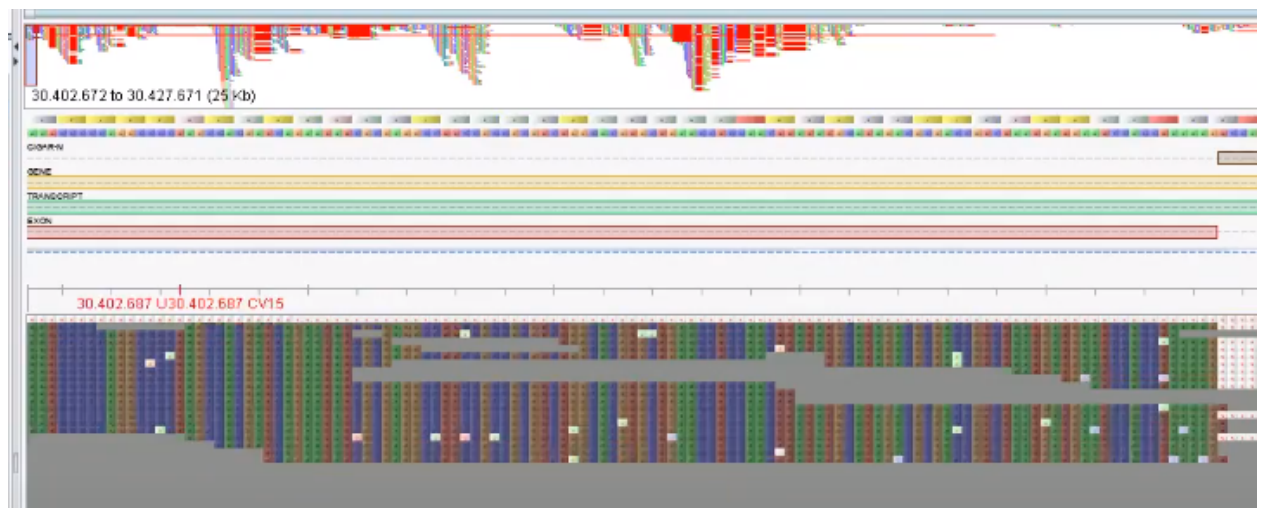
```
$ samtools view -bS file.sam > file.bam
```

Check the alignment with tablet or IGV

In addition to BAM and FASTA files to examine the alignment, we also need an indexed BAM file (BAI). It can be obtained using the following command:

```
$ samtools index STAR_mapped_SRR7287381_Aligned.sortedByCoord.bam
```

We have 5 chromosomes, mitochondrial and chloroplast DNA on our reference genome, but the reads are only present on the first chromosome, which is shown in the picture below. We can select each read and see the sequence ID, length, cigar - there we can see how the read mapped reference genome. All reads align in exon regions.



Part 4

Count the number of reads per gene using FeatureCounts

1. Checking the data qualification for strand-specific property

This can be checked through manuals of the dataset and also through coding with bash. For this, we will need a .bed file and a .bam file. We already have a .bam file

and we also need to produce a .bed file. We can do this through a software gtf2bed:

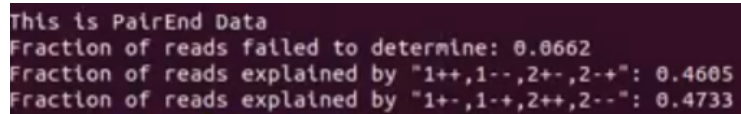
The needed code to obtain a .bed file:

After downloading the scripts with wget command, we obtained two scripts: genePredToBed and gtfToGenePred. We activate them with the given commands:

```
$ chmod +x genePredToBed
$ chmod +x gtfToGenePred
$ ./gtfToGenePred Arabidopsis_thaliana.TAIR10.51.gtf
Arabidopsis_thaliana.TAIR10.51.genePred
$ ./genePredToBed Arabidopsis_thaliana.TAIR10.51.genePred
Arabidopsis_thaliana.TAIR10.51.bed12
```

As our group has the dataset of SRR7287381 the code is:

```
$ Infer_experiment.py -r
../genome/Arabidopsis_thaliana/TAIR10.51.bed12 -i
STAR_mapped_SRR7287381_Aligned.sortedByCoord.out.bam
```



```
This is PairEnd Data
Fraction of reads failed to determine: 0.0662
Fraction of reads explained by "1++,1--,2+-,2-+": 0.4605
Fraction of reads explained by "1+-,1-+,2++,2--": 0.4733
```

The results indicate that our dataset is non-strand specific read.

2. Using the package FeatureCounts

```
$ conda install subread
```

In the featureCounts, we specify the strandness property of the dataset via command -s. The default parameter is 0 as unstranded in the script but if the dataset is different, other parameters can be used such as; 1 for stranded and 2 for reverse stranded. Since our data is a non-strand specific read we need to provide a parameter to indicate that, which is 0. Therefore the code for our dataset is:

```
$ featureCounts -a
../genome/Arabidopsis_thaliana.TAIR10.51.gtf -o
STAR_mapped_SRR7287381_Aligned.sortedByCoord
../alignment/STAR_mapped_SRR7287381_Aligned.sortedByCoord.out.
bam

$ featureCounts -a Arabidopsis_thaliana.TAIR10.51.gtf -s 0 -o
STAR_mapped_SRR7287381_Aligned.sortedByCoord_0
STAR_mapped_SRR7287381_Aligned.sortedByCoord.out.bam
```

Results indicate that there is 5931528 assigned, 3302229 unassigned multimapping, 849997 unassigned with no features and 319119 unassigned ambiguity.

```
$ featureCounts -aArabidopsis_thaliana.TAIR10.51.gtf -s 0 -O
-p -o STAR_mapped_SRR7287381_Aligned.sortedByCoord_0
STAR_mapped_SRR7287381_Aligned.sortedByCoord.out.bam
# in the exon level (successfully assigned alignments: 59.7%)
$ featureCounts -a
../genome/Arabidopsis_thaliana.TAIR10.51.gtf -s 0 -O -p -f -o
STAR_mapped_SRR7287381_Aligned.sortedByCoord_0_exon
STAR_mapped_SRR7287381_Aligned.sortedByCoord.out.bam
```

Part 5

Variant discovery with GATK

1. Download GATK as Docker image

```
$ sudo docker pull broadinstitute/gatk
```

2. Data cleanup > Variant Discovery > Evaluation

```
$ samtools view -h STAR_mapped_SRR7287381_Aligned.out.bam |
less
```

```
$ sudo docker run --rm -name GATK_1 -v
/home/tanya/nsa:/gatk/my_data -it e1a358670b4f bash
```

The function AddOrReplaceReadGroups assigns all the reads in a file to a single new read-group. We use this function to add a read group. This way we can assign all the reads in the input file to a single new read-group.

```
$ ./gatk AddOrReplaceReadGroups -I
my_data/alignment/2ndPass/STAR_mapped_SRR7287381_withoutRG_
Aligned.out.sorted_RGadded.bam -LB Arabidopsis_roots -PL
Illumina -PU unit1 -SM SRR7287381
STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted.RG.bam
STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted.bam

$ ./gatk AddOrReplaceReadGroups -I
my_data/alignment/2ndPass/STAR_mapped_SRR7287381_withoutRG_
```

```
Aligned.out.sorted_RGadded.bam -LB Arabidopsis_roots -PL
Illumina -PU unit1 -SM SRR7287381
```

```
$ samtools view -h
GATK_practicals/STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted_RGadded.bam | less
```

RG created with ID: 1, LB (library): Illumina, SM: SRR7287381 and PU: unit1

```
$ ./gatk MarkDuplicates -I
my_data/alignment/2ndPass/GATK_practicals/STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted_RGadded.bam -M
my_data/alignment/2ndPass/GATK_practicals/STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted_RGadded.MD.txt -O
my_data/alignment/2ndPass/GATK_practicals/STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted_RGadded.MD.bam
--READ_NAME_REGEX null
```

The function `SplitNCigarReads` takes a .bam file and outputs a .bam file with reads split at N CIGAR elements and strings updated. It reassigns mapping qualities for well made alignments to match as DNA conventions, since RNA alignments have different conventions than DNA aligners.

```
$ ./gatk SplitNCigarReads -I
my_data/alignment/2ndPass/GATK_practicals/STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted_RGadded.MD.bam -O
my_data/alignment/2ndPass/GATK_practicals/STAR_mapped_SRR7287381_withoutRG_Aligned.out.sorted_RGadded.MD.split.bam -R
my_data/genome/Arabidopsis_thaliana.TAIR10.dna.toplevel.fa -L 1
```

```
$ ./gatk CreateSequenceDictionary -R
my_data/genome/Arabidopsis_thaliana.TAIR10.dna.toplevel.fa -O
my_data/genome/Arabidopsis_thaliana.TAIR10.dna.toplevel.dict
```

```
$ cat arabidopsis_thaliana.vcf | sed 's/The 1001 Genomes Project_2016/The_1001_Genomes_Project_2016/g' >
arabidopsis_thaliana_corrected_spaces.vcf
```

```
$ awk '{if(NR!=1687355){print $0}}'
arabidopsis_thaliana_corrected_spaces.vcf >
arabidopsis_thaliana_corrected_spaces_rmW.vcf
```

```
$ ./gatk IndexFeatureFile -I
my_data/genome/Arabidopsis_thaliana_rmW.vcf
```

```
$ samtools index
gatk/my_data/2ndPass/GATK_practicals/STAR_mapped_SRR7287381
_Aligned.out.sorted.RG.rmdup.split.BQSR.bam
```

The function HaplotypeCaller takes a .bam file and calls SNPs and indels; when the dataset shows regions with variation, the function discards the mapping information and reassembles the reads in that region.

```
$ ./gatk HaplotypeCaller -R
my_data/genome/Arabidopsis_thaliana_TAIR10.dna.toplevel.fa
-I
my_data/alignment/2ndPass/GATK_practicals/STAR_mapped_SRR72
87381_Aligned.out.sorted.RG.rmdup.split.BQSR.bam -O
my_data/alignment/2ndPass/GATK_practicals/STAR_mapped_SRR72
87381_Aligned.out.sorted.RG.rmdup.split.BQSR.g.vcf.gz -ERC
GVCF -L 1
```

The function GenotypeGVCFs outputs a .vcf file with all the samples having been jointly genotyped. It is a very useful tool for joint genotyping on a single input. The input file can contain one or more samples.

```
$ ./gatk GenotypeGVCFs -R
my_data/genome/Arabidopsis_thaliana.TAIR10.dna.toplevel.fa
-V
my_data/alignment/2ndPass/GATK_practicals/STAR_mapped_SRR72
87381_Aligned.out.sorted.RG.rmdup.split.BQSR.g.vcf.gz -O
output.vcf.gz -L 1
```

Selecting SNP variants via function SelectVariants: the function is selecting a subset of variants with given criteria.

```
$ ./gatk SelectVariants -V
my_data/alignment/2ndPass/GATK_practicals/output.vcf.gz
-select-type SNP -O
my_data/alignment/2ndPass/GATK_practicals/output_SNP.vcf.gz
```

Filtering the most trustable variants via function VariantFiltration: the function is filtering the dataset with given criteria.

```
$ ./gatk VariantFiltration -V
my_data/alignment/2ndPass/GATK_practicals/output_SNP.vcf.gz
-filter "OD < 2.0" --filter-name "QD2" -filter "QUAL <
30.0" --filter-name "QUAL30" -filter "SOR > 3.0"
--filter-name "SOR3" -filter "FS > 60.0" --filter-name
"FS60" -filter "MQ < 40.0" --filter-name "MQ40" -filter
"MQRankSum < -12.5" --filter-name "MQRankSum-12.5" -filter
```

```
"ReadPosRankSum < -8.0" --filter-name "ReadPosRankSum-8" -O
my_data/alignment/2ndPass/GATK_practicals/output_SNP_filtered.
vcf.gz
```

With SnpEff, we are checking if the database is available. The snpEff helps to filter and manipulate .vcf (genomic annotated files) and annotates the effects of variants on genes. The end file stores a big text file with our dataset gene sequence variations.

```
$ snpEff databases | grep -i arabidopsis

$ snpEff Arabidopsis_thaliana

$ snpEff Arabidopsis_thaliana output_SNP_filtered.vcf.gz >
snps_filtered.ann.vcf
```

Part 6

EDirect

1. Select a gene from an organism according to your preferences and write a command to get protein sequence from RefSeq database with EDirect tools in fasta format (using esearch tool).

```
$ esearch -db nuccore -query "(lycopene[All Fields] AND
cyclase[All Fields]) AND Solanum lycopersicum[Organism] AND
(100[SLEN] : 5000[SLEN])" | efetch -format fasta >
solanum_lycopersicum.fasta
```

2. Create a new .txt file and write the NCBI accession number of protein sequences in it and repeat the previous exercise with the epost tool.

```
$ esearch -db nuccore -query "(lycopene[All Fields] AND
cyclase[All Fields]) AND Solanum lycopersicum[Organism] AND
(100[SLEN] : 5000[SLEN])" | efetch -format gbk | xtract
-pattern INSDSeq -element INSDSeq_primary-accession >
acc.numbers.txt

$ cat acc.numbers.txt | epost -db nuccore -format acc |
efetch -format fasta > solanum_lycopersicum_epost.fasta
```

3. Use elink to get corresponding mRNA from RefSeq and download records in GenBank format.

```
$ esearch -db nuccore -query NM_001316759.1 | elink -target
protein -name nuccore_protein | efetch -format gp
```

4. Check how many sequences are linked to this in the nuccore database. With xtract tool extract accession number, length, and definition (hint: download the results in gbc format).

```
$ esearch -db nuccore -query NM_001316759.1 | elink -target
protein -name nuccore_protein | efetch -format gbc |xtract
-pattern INSDSeq -element INSDSeq_primary-accession
INSDSeq_length INSDSeq_definition

NP_001303688      500  lycopene beta cyclase, chloroplastic
[Solanum lycopersicum]
```

Standalone BLAST

1. Use a fasta sequence of a protein, which you have obtained in the previous task and write a command to search this sequence against the RNA RefSeq database and limit the search space to some other species and check if the transcript for this protein is available.

```
$blastx -db refseq_protein -query
solanum_lycopersicum_epost.fasta -out results.txt -outfmt 0
-entrez_query "solanum lycopersicum[ORGN]" -remote
```

2. Download all mRNA sequences of the previously used organism and create a local blast database. Repeat the analysis with the created database.

```
$ makeblastdb -in Arabidopsis_mRNA_RefSeq.fasta -input_type
fasta -dbtype "nucl" -parse_seqids

$ blastn -db Arabidopsis_mRNA_RefSeq.fasta -query
solanum_lycopersicum_epost.fasta -out results.txt -outfmt 0
```