Brief Guide for NGS Transcriptomics: From gene expression to genetics.

by
Aureliano Bombarely

Lectures:

1. **Basics of the Next Generation Sequencing (NGS).**
   1.1. The sequencing revolutions.
   1.2. Strengths and weaknesses of the different technologies.
   1.3. Inputs and outputs.
2. **RNAseq experiment design.**
   2.1. Reference vs Non-reference.
   2.2. High heterozygosity and polyploid polyploid problem.
   2.3. Tissue selection and treatments.
   2.4. Sequencing technology.
3. **RNAseq expression analysis.**
   3.1. Reference preparation and read mapping.
   3.2. Gene expression.
   3.3. Analysis and visualization.
4. **Use of RNAseq reads for phylogeny and genetics.**
   4.1. Recovering full length mRNA: Reference guided assembly.
   4.2. Phylogeny though RNAseq: From gene tree to species tree.
   4.3. From reads to markers: SNP calling.
   4.4. Population genetics and NGS.

Exercises:

1. **Basic Linux commands.**
2. **Sequencing evaluation.**
3. **Simple read mapping.**
4. **Simple de-novo assembly.**
5. **Basic R commands**
6. **Functional annotation.**
7. **Differential gene expression.**
8. **Cluster analysis for gene expression.**
9. **Selecting genes for phylogeny.**
10. **SNP calling and filtering.**
11. **Analysis of the population structure.**

Exercises:

1. **Basic Linux commands.**
2. **Sequencing evaluation.**
3. **Simple read mapping.**
4. **Simple de-novo assembly.**
5. **Basic R commands**
6. **Functional annotation.**
7. **Differential gene expression.**
8. **Cluster analysis for gene expression.**
9. **Selecting genes for phylogeny.**
10. **SNP calling and filtering.**
11. **Analysis of the population structure.**

1. **Basic Linux commands.**

Goal: Learn basic linux commands to move through the system, manipulate files and run programs. Use command combinations to retrieve basic information of bioinformatic files.

Supporting information: http://www.slideshare.net/aubombarely/basiclinux

Exercise List:

1. Login to the system and open a terminal.
2. Print the working directory.
3. List the files and the directories from the working directory.
4. Change working directory to /Data/rice
5. List the list of files with its size in an human readable form.
6. Create a directory in /Data/rice with the name 'exercise1'
7. Change name of the directory "Refereence" to "Reference"

1. **Basic Linux commands.**

Exercise List:

8. Decompress the files with gunzip or tar -zxvf and print the size of each file.

9. Print the first 10 lines from the file IRGSP-1.0_cds_2012-10-02.fasta

10. Print the last 20 lines from the file IRGSP-1.0_cds_2012-10-02.fasta

11. Count the number of sequences from the IRGSP-1.0_cds_2012-10-02.fasta file

12. Count the number of sequences from the IRGSP-1.0_genome.fasta

13. Print the size of the rice genome.

14. Add the folder /home/bioinfo/Programs/GenoToolBox/bin to the PATH environmental variable and print it

15. Print the help for FastaSeqStats script. Using this script calculate the stats for the rice genome and transcriptome (*Note: Install bioperl with "sudo apt-get install bioperl")

16. Using grep command get a list of all the rice chromosome 2 genes. Using sed command remove the sign of starting ID (>).

1. **Basic Linux commands.**

Exercise List:

17. Print the help for FastaExtract script. Using this script and the list of cds from the chromosome 2, extract the sequence of those genes into a file with the name: 'rice_cds_chr02.fasta'.

18. Move this file to the 'exercise1' dir.

19. Using a combination of grep, sed, sort and uniq print the number of CDS per chromosome.

20. Using a combination of cut, sed, sort, uniq and head print the list of the 10 most represented function in the locus.gff file

1. **Basic Linux commands.**

Exercise List:

1. Login to the system and open a terminal.

   Password: bioinfo

2. Print the working directory.

   pwd

3. List the files and the directories from the working directory.

   ls

4. Change working directory to /Data/rice

   cd /Data/rice

5. List the list of files with its size in an human readable form.

   ls -lh

1. **Basic Linux commands.**

Exercise List:

6. Create a directory in /Data/rice with the name ‘exercise1’

```
mkdir exercise1
```

7. Change name of the directory “Refereence” to “Reference”

```
mv Refereence Reference
```

8. Decompress the files with gunzip or tar -zxvf and print the size of each file.

```
gunzip IRGSP-1.0_cds_2012-10-02.fasta.gz

gunzip IRGSP-1.0_genome.fasta.gz

gunzip IRGSP-1.0_protein_2012-10-02.fasta.gz

gunzip IRGSP-1.0_transcripts_2012-10-02.fasta.gz

tar -zxvf IRGSP-1.0_representative_2012-10-17.tar.gz
```

1. **Basic Linux commands.**

Exercise List:

9. Print the first 10 lines from the file IRGSP-1.0_cds_2012-10-02.fasta

```
head IRGSP-1.0_cds_2012-10-02.fasta
```

10. Print the last 20 lines from the file IRGSP-1.0_cds_2012-10-02.fasta

```
tail -n 20 IRGSP-1.0_cds_2012-10-02.fasta
```

11. Count the number of sequences from the IRGSP-1.0_cds_2012-10-02.fasta file

```
grep -c '>' IRGSP-1.0_cds_2012-10-02.fasta
```

12. Count the number of sequences from the IRGSP-1.0_genome.fasta

```
grep -c '>' IRGSP-1.0_genome.fasta
```

13. Print the size of the rice genome.

```
grep -v '>' IRGSP-1.0_genome.fasta | wc -c
```

1. **Basic Linux commands.**

Exercise List:

14. Add the folder /home/bioinfo/Programs/GenoToolBox/bin to the PATH environmental variable and print it

```
export PATH=/home/bioinfo/Programs/GenoToolBox/bin:$PATH

echo $PATH
```

15. Print the help for FastaSeqStats script. Using this script calculate the stats for the rice genome and transcriptome.

```
FastaSeqStats -h

FastaSeqStats -i IRGSP-1.0_genome.fasta
```

16. Using grep command get a list of all the rice chromosome 2 genes. Using sed command remove the sign of starting ID (>).

```
grep '>Os02' IRGSP-1.0_cds_2012-10-02.fasta | sed -r 's/>//' > rice_cds_chr02.list.txt
```

1. **Basic Linux commands.**

Exercise List:

17. Print the help for FastaExtract script. Using this script and the list of cds from the chromosome 2, extract the sequence of those genes into a file with the name: 'rice_cds_chr02.fasta'.

```
FastaExtract -h

FastaExtract -f IRGSP-1.0_cds_2012-10-02.fasta -i rice_cds_chr02.list.txt -o rice_cds_chr02.fasta
```

18. Move this file to the 'exercise1' dir.

```
mv rice_cds_chr02.fasta /home/bioinfo/Data/rice/exercise1
```

19. Using a combination of grep, sed, sort and uniq print the number of CDS per chromosome.

```
grep '>' IRGSP-1.0_genome.fasta | sed -r 's/t.+//' | sort | uniq -c
```

20. Using a combination of cut, sed, sort, uniq and head print the list of the 10 most represented function in the locus.gff file

```
cut -f9 locus.gff | sed -r 's/ID=.+Note=//' | sed -r 's/\s+\(Os.+//' | sort | uniq -c | sort -nr | head
```

Exercises:

1. **Basic Linux commands.**
2. **Sequencing evaluation.**
3. **Simple read mapping.**
4. **Simple de-novo assembly.**
5. **Basic R commands**
6. **Functional annotation.**
7. **Differential gene expression.**
8. **Cluster analysis for gene expression.**
9. **Selecting genes for phylogeny.**
10. **SNP calling and filtering.**
11. **Analysis of the population structure.**

2. **Sequence Evaluation**

Goal: Evaluate the quality of Illumina reads.

Supporting information: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/

Example good Illumina dataset:

http://www.bioinformatics.babraham.ac.uk/projects/fastqc/good_sequence_short_fastqc/fastqc_report.html

Example poor Illumina dataset:

http://www.bioinformatics.babraham.ac.uk/projects/fastqc/bad_sequence_fastqc/fastqc_report.html

Dataset origin: SRA database, study SRP004651

http://trace.ncbi.nlm.nih.gov/Traces/sra?study=SRP004651

Shen Y et al. 2011, "Transcriptome dynamics through alternative polyadenylation in developmental and environmental responses in plants revealed by deep sequencing" Genome Res. 21:1478-86

2. **Sequence Evaluation**

Exercise List:

1. Type in the terminal ‘fastqc’ to start the program.

2. Load the fastq files from /home/bioinfo/Data/rice/RNAseq and run the analysis.

3. Explore the different sections.

2. **Sequence Evaluation**

Exercise List:

1. Type in the terminal ‘fastqc’ to start the program.

2. Load the fastq files from /home/bioinfo/Data/rice/RNAseq and run the analysis.

3. Explore the different sections.

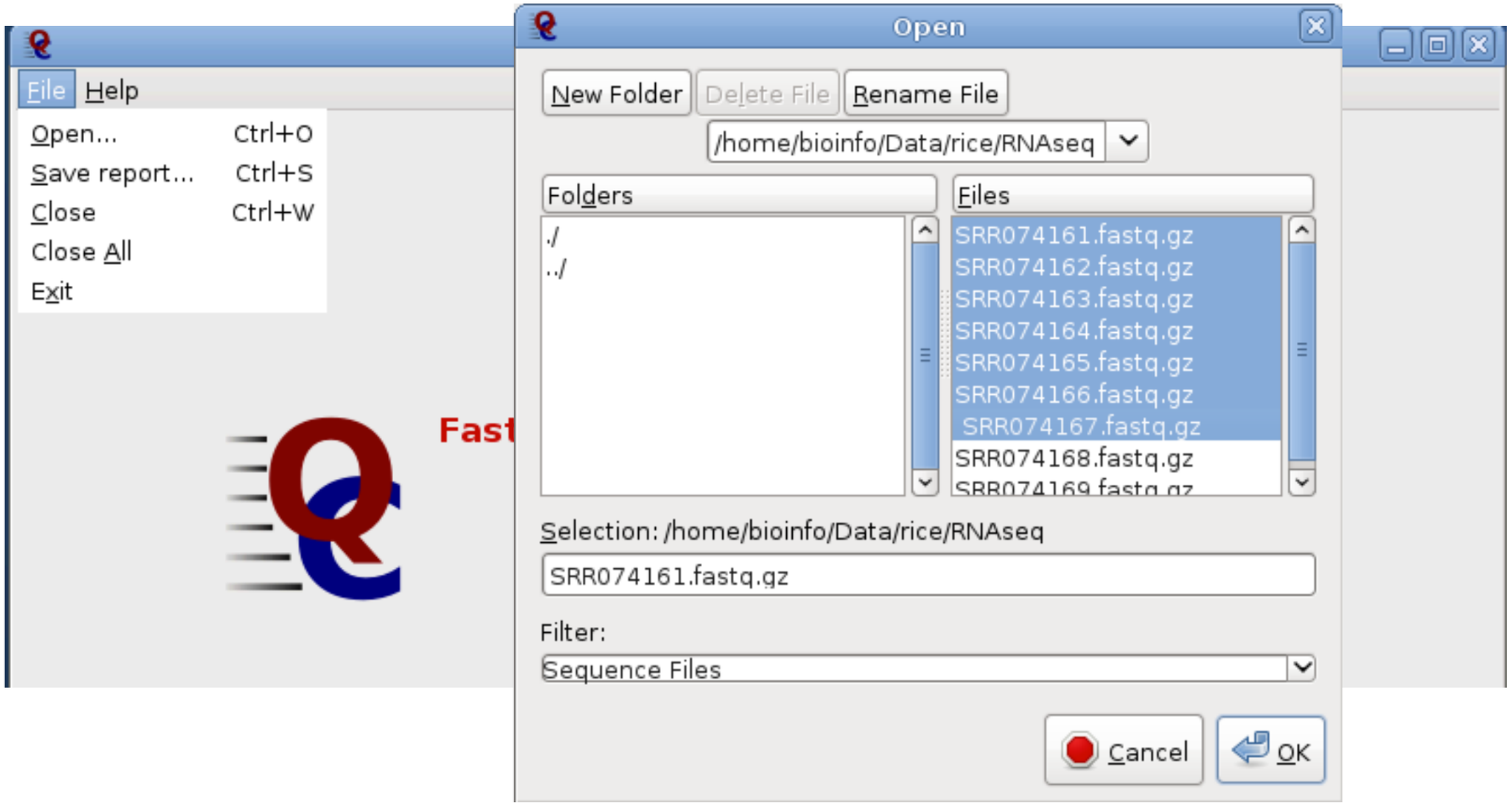
2. **Sequence Evaluation**

Exercise List:

1. Type in the terminal ‘fastqc’ to start the program.

fastqc

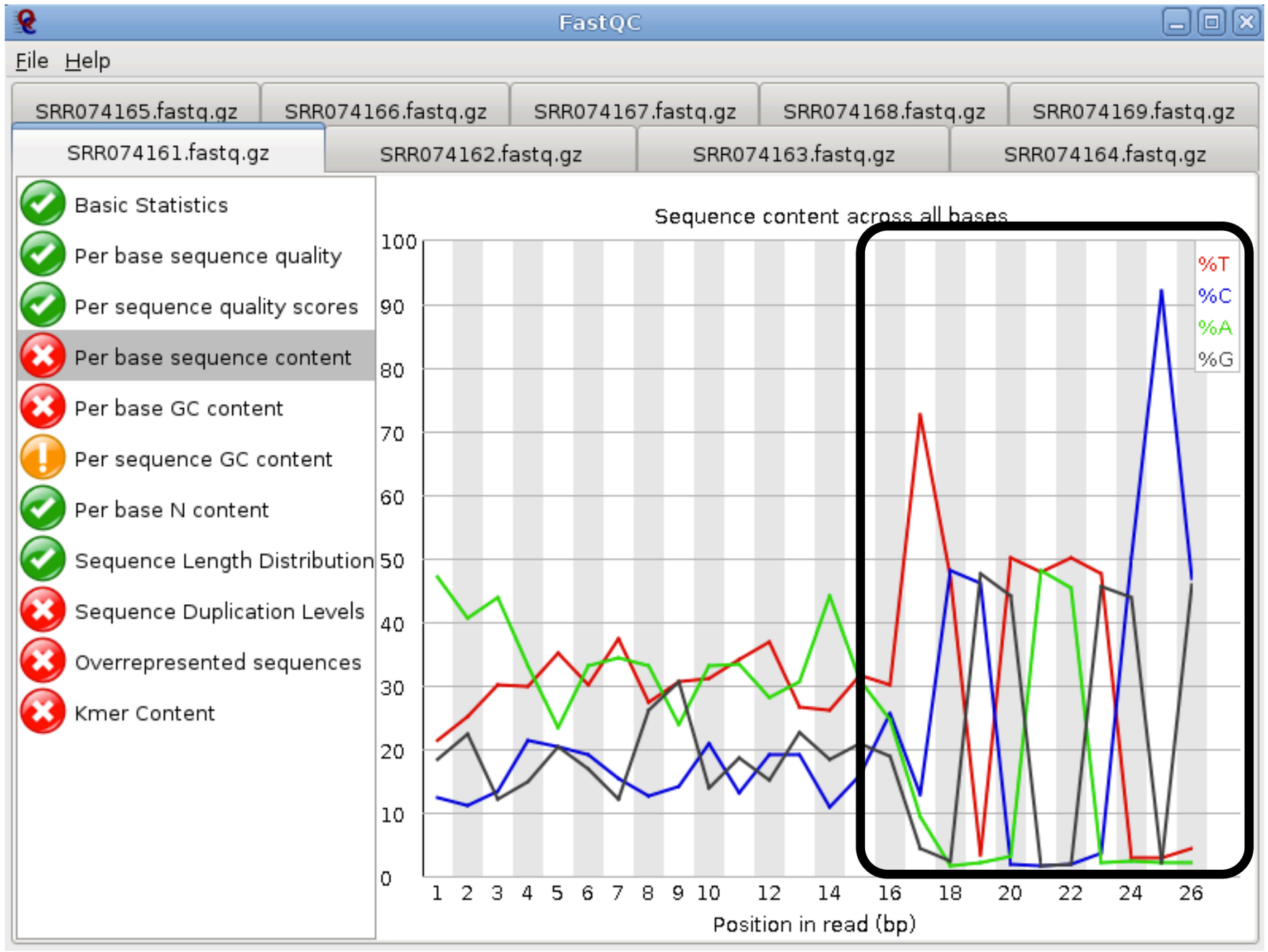
2. Load the fastq files from /home/bioinfo/Data/rice/RNAseq and run the analysis.



2. **Sequence Evaluation**

Exercise List:

3. Explore the different sections.



This profile may be an indicative of an adapter

Exercise List:

4. Download the Illumina adapter sequence from http://intron.ccam.uchc.edu/groups/tgcore/wiki/013c0/Solexa_Library_Primer_Sequences.html and convert them in a fasta file (copy, paste and manual editing)

5. Process the sequences using fastq-mcf with Q20 L20

6. Rerun fastqc to see the quality after the read processing

Good Practices Note:

- Get the fastq-mcf output during the read processing to create a report file or table.
- Move the read result to a new folder, for example: 'processed'. Keep the old sequence in another folder, for example: 'raw'

2. **Sequence Evaluation**

Exercise List:

4. Download the Illumina adapter sequence from http://intron.ccam.uchc.edu/groups/tgcore/wiki/013c0/Solexa_Library_Primer_Sequences.html and convert them in a fasta file (copy, paste and manual editing)

5. Process the sequences using fastq-mcf with Q20 L20

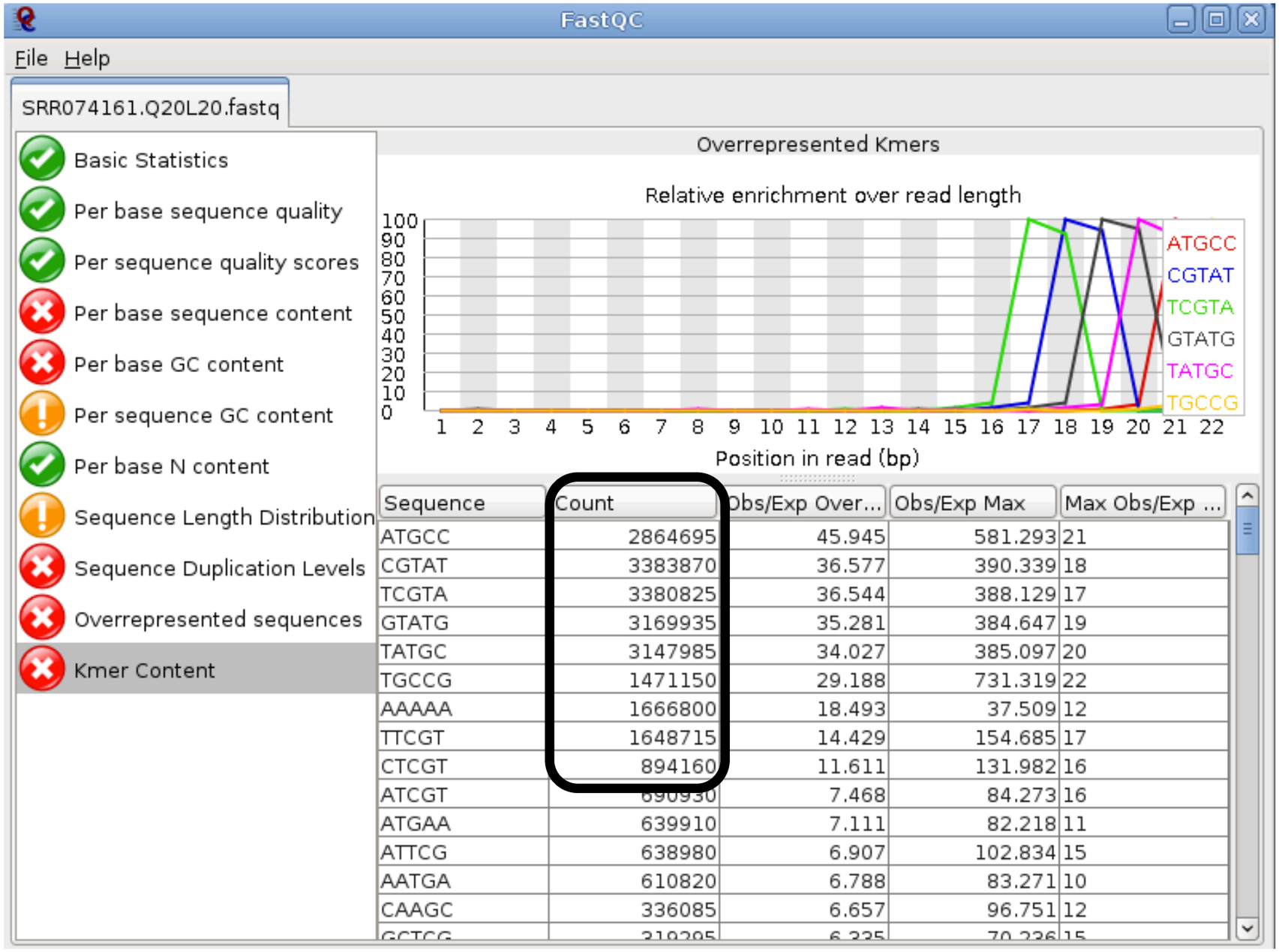
```
fastq-mcf -o SRR074161.Q20L20.fastq -q 20 -l 20 adapters.fa SRR074161.fastq
fastq-mcf -o SRR074162.Q20L20.fastq -q 20 -l 20 adapters.fa SRR074162.fastq
fastq-mcf -o SRR074163.Q20L20.fastq -q 20 -l 20 adapters.fa SRR074163.fastq
fastq-mcf -o SRR074164.Q20L20.fastq -q 20 -l 20 adapters.fa SRR074164.fastq
fastq-mcf -o SRR074165.Q20L20.fastq -q 20 -l 20 adapters.fa SRR074165.fastq
fastq-mcf -o SRR074166.Q20L20.fastq -q 20 -l 20 adapters.fa SRR074166.fastq
fastq-mcf -o SRR074167.Q20L20.fastq -q 20 -l 20 adapters.fa SRR074167.fastq
fastq-mcf -o SRR074168.Q20L20.fastq -q 20 -l 20 adapters.fa SRR074168.fastq
fastq-mcf -o SRR074169.Q20L20.fastq -q 20 -l 20 adapters.fa SRR074169.fastq
```

6. Rerun fastqc to see the quality after the read processing

2. **Sequence Evaluation**

Exercise List:

6. Rerun fastqc to see the quality after the read processing



Sequence	Count	Obs/Exp Over...	Obs/Exp Max	Max Obs/Exp ...
ATGCC	2864695	45.945	581.293	21
CGTAT	3383870	36.577	390.339	18
TCGTA	3380825	36.544	388.129	17
GTATG	3169935	35.281	384.647	19
TATGC	3147985	34.027	385.097	20
TGCCG	1471150	29.188	731.319	22
AAAAA	1666800	18.493	37.509	12
TTCGT	1648715	14.429	154.685	17
CTCGT	894160	11.611	131.982	16
ATCGT	690930	7.468	84.273	16
ATGAA	639910	7.111	82.218	11
ATTCG	638980	6.907	102.834	15
AATGA	610820	6.788	83.271	10
CAAGC	336085	6.657	96.751	12
GCTCG	319295	6.335	70.236	15

It shows Kmers in a high percentage of the reads

2. **Sequence Evaluation**

Exercise List:

7. Based in the Kmer Content section of the FastQC report, create a new adapter sequence and add to the adapter file.

8. Rerun fastq-mcf with the adjusted L to process the reads and fastqc to evaluate them.

Good Practices Note:

- Adjust the L value based in the start coordinate of the kmer content

Exercise List:

7. Based in the Kmer Content section of the FastQC report, create a new adapter sequence and add to the adapter file.

```
TCGTATGCC
```

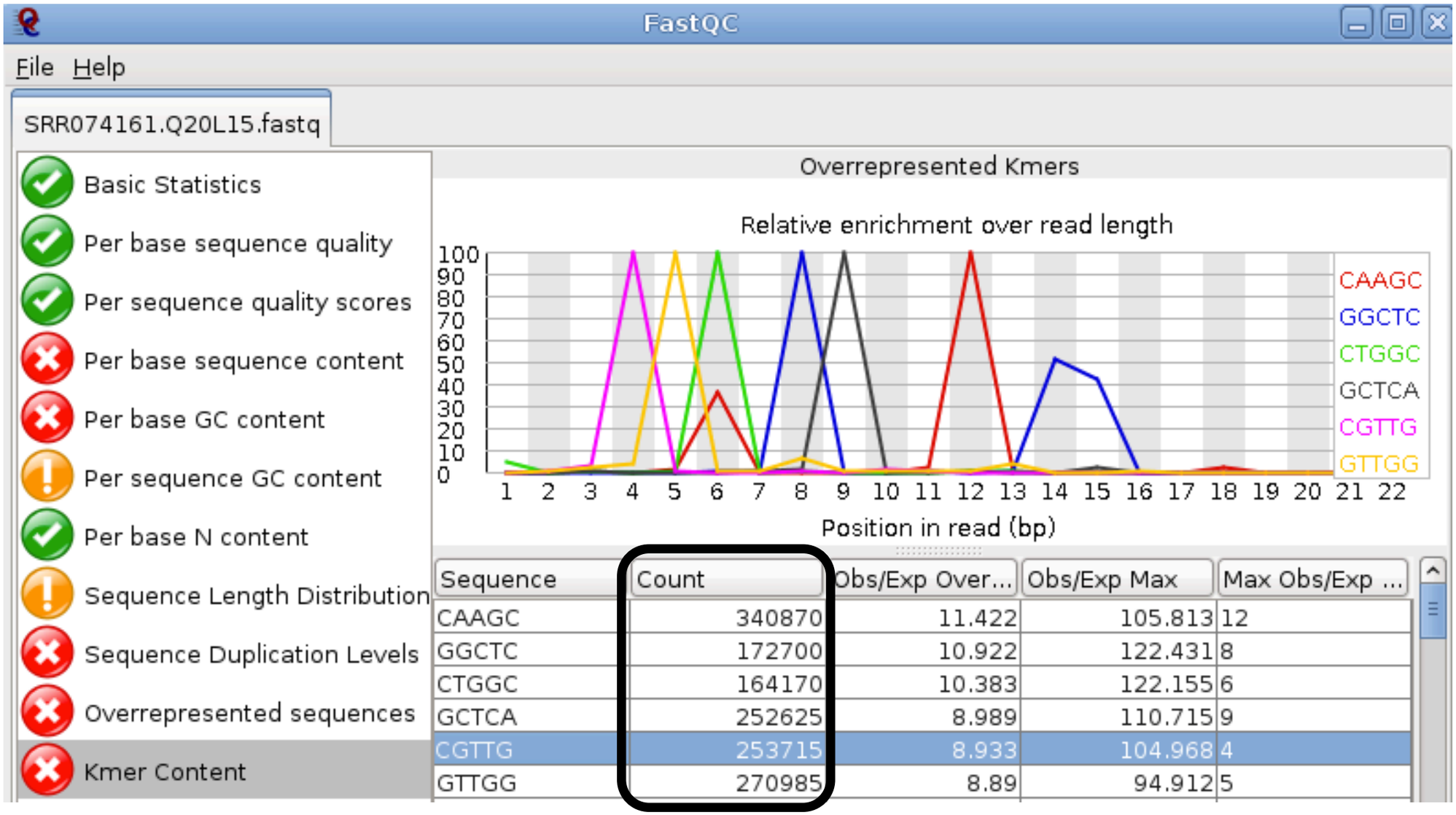
8. Rerun fastq-mcf to process the reads and fastqc to evaluate them.

```
fastq-mcf -o SRR074161.Q20L15.fastq -q 20 -l 15 adapters.fa SRR074161.fastq
fastq-mcf -o SRR074162.Q20L15.fastq -q 20 -l 15 adapters.fa SRR074162.fastq
fastq-mcf -o SRR074163.Q20L15.fastq -q 20 -l 15 adapters.fa SRR074163.fastq
fastq-mcf -o SRR074164.Q20L15.fastq -q 20 -l 15 adapters.fa SRR074164.fastq
fastq-mcf -o SRR074165.Q20L15.fastq -q 20 -l 15 adapters.fa SRR074165.fastq
fastq-mcf -o SRR074166.Q20L15.fastq -q 20 -l 15 adapters.fa SRR074166.fastq
fastq-mcf -o SRR074167.Q20L15.fastq -q 20 -l 15 adapters.fa SRR074167.fastq
fastq-mcf -o SRR074168.Q20L15.fastq -q 20 -l 15 adapters.fa SRR074168.fastq
fastq-mcf -o SRR074169.Q20L15.fastq -q 20 -l 15 adapters.fa SRR074169.fastq
```

2. **Sequence Evaluation**

Exercise List:

8. Rerun fastq-mcf to process the reads and fastqc to evaluate them.



Sequence	Count	Obs/Exp Over...	Obs/Exp Max	Max Obs/Exp ...
CAAGC	340870	11.422	105.813	12
GGCTC	172700	10.922	122.431	8
CTGGC	164170	10.383	122.155	6
GCTCA	252625	8.989	110.715	9
CGTTG	253715	8.933	104.968	4
GTTGG	270985	8.89	94.912	5

It still shows Kmers, but the count is much lower (<10% of the total reads)

Exercises:

1. **Basic Linux commands.**
2. **Sequencing evaluation.**
3. **Simple read mapping.**
4. **Simple de-novo assembly.**
5. **Basic R commands**
6. **Functional annotation.**
7. **Differential gene expression.**
8. **Cluster analysis for gene expression.**
9. **Selecting genes for phylogeny.**
10. **SNP calling and filtering.**
11. **Analysis of the population structure.**

3. **Simple Read Mapping**

Goal: Map processed reads to a reference using Bowtie2

Supporting information: http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml

How to align reads... quick guide:

1. Build the reference index for the alignment tool.

2. Run the alignment.

3. Evaluate how many reads have aligned and rerun with other conditions if it is not a satisfactory result.

Note about sam files and cufflinks:

Cufflinks uses a sam header slightly different from the standard head.

Instead @HD, @SG, @PD, Cufflinks expect @HD, @PD, @SG (alphabetical order)

3. **Simple Read Mapping**

Exercise List:

1. Build the rice genome index using bowtie2-build

2. Run an alignment with the default parameters for the read dataset SRR074161.

3. Using samtools view count how many reads have been mapped.

4. Using samtools view, select the reads that have been mapped. Using command line combination (grep, sed, awk) evaluate how many matches map in more than one place with the same score (AS primary match score, XS secondary match score).

5. Select the matches where the main map score (AS) is higher than the secondary map score (XS) using the following command.

```
perl -ne 'chomp($_); @a = split(/\t/, $_); if ($a[11] ne $a[12]) { print "$_\n"}'
SRR074161.map.sam > SRR071161.map.uniq.sam
```

6. Convert the SRR071161.map.uniq.sam to bam format.

7. Extract the non mapped reads into a sam file and pipe it selecting the columns 1 and 10 to create a fasta file.

Exercise List:

8. Delete the sam files.

9. Repeat the same operation with the rest of the reads. Write a bash script to do it.

3. **Simple Read Mapping**

Exercise List:

1. Build the rice genome index using bowtie2-build

```
bowtie2-build IRGSP-1.0_genome.fasta rice_genome
```

2. Run an alignment with the default parameters for the read dataset SRR074161

```
bowtie2 -t -x rice_genome -U SRR074161.Q20L15.fastq -S SRR074161.sam
```

3. Using samtools view count how many reads have been mapped.

```
/home/bioinfo/Programs/samtools-0.1.18/samtools view -c -F 4 -S SRR074161.sam
```

4. Using samtools view, select the reads that have been mapped.

```
samtools view -F 4 -S SRR074161.sam > SRR074161.map.sam
```

5. Using command line combination (grep, sed, awk) evaluate how many matches map in more than one place with the same score (AS primary match score, XS secondary match score).

```
grep -v '^@' SRR074161.map.sam | cut -f12,13 | awk '{ if ($1 == $2) print $0}' | wc -l
```

Exercise List:

6. Convert the SRR071161.map.uniq.sam to bam format.

```
samtools view -SH SRR071161.sam | sort > SRR071161.head
cat SRR071161.sam SRR071161.map.uniq.sam > SRR071161.map.uniq.head.sam
samtools view -Sb -o SRR07161.bam SRR071161.map.uniq.head.sam
```

7. Extract the non mapped reads into a sam file and pipe it selecting the columns 1 and 10 to create a fasta file.

```
samtools view -f 4 -S SRR071161.sam | cut -f1,10 | sed -r 's/SRR/>SRR/' | sed -r 's/\t/
\n/' > SRR071161.nomap.fasta
```

8. Delete the sam files.

```
rm *.sam
```

9. Repeat the same operation with the rest of the reads. Write a bash script to do it.

3. **Simple Read Mapping**

```
#!/bin/bash

bowtie2 -t -x rice_genome -U ../processed/SRR074162.Q20L15.fastq -S SRR074162.sam
grep -v '^@' SRR074162.sam | cut -f12,13 | sed -r 's/AS:i://' | sed -r 's/XS:i://' | awk '{ if ($1 =! $2) print $0}' | wc -l
samtools view -F 4 -S SRR074162.sam > SRR074162.map.sam
perl -ne 'chomp($_); @a = split(/\t/, $_); if ($a[11] ne $a[12]) { print "$_\n"}' SRR074162.map.sam > SRR074162.map.uniq.sam
samtools view -SH SRR074162.sam | sort > SRR074162.header
cat SRR074162.header SRR074162.map.uniq.sam > SRR074162.map.uniq.head.sam
samtools view -Sb -o SRR074162.bam SRR074162.map.uniq.head.sam
rm *.sam
rm SRR074162.header

bowtie2 -t -x rice_genome -U ../processed/SRR074163.Q20L15.fastq -S SRR074163.sam
grep -v '^@' SRR074163.sam | cut -f12,13 | sed -r 's/AS:i://' | sed -r 's/XS:i://' | awk '{ if ($1 =! $2) print $0}' | wc -l
samtools view -F 4 -S SRR074163.sam > SRR074163.map.sam
perl -ne 'chomp($_); @a = split(/\t/, $_); if ($a[11] ne $a[12]) { print "$_\n"}' SRR074163.map.sam > SRR074163.map.uniq.sam
samtools view -SH SRR074163.sam | sort > SRR074163.header
cat SRR074163.header SRR074163.map.uniq.sam > SRR074163.map.uniq.head.sam
samtools view -Sb -o SRR074163.bam SRR074163.map.uniq.head.sam
rm *.sam
rm SRR074163.header

bowtie2 -t -x rice_genome -U ../processed/SRR074164.Q20L15.fastq -S SRR074164.sam
grep -v '^@' SRR074164.sam | cut -f12,13 | sed -r 's/AS:i://' | sed -r 's/XS:i://' | awk '{ if ($1 =! $2) print $0}' | wc -l
samtools view -F 4 -S SRR074164.sam > SRR074164.map.sam
perl -ne 'chomp($_); @a = split(/\t/, $_); if ($a[11] ne $a[12]) { print "$_\n"}' SRR074164.map.sam > SRR074164.map.uniq.sam
samtools view -SH SRR074164.sam | sort > SRR074164.header
cat SRR074164.header SRR074164.map.uniq.sam > SRR074164.map.uniq.head.sam
samtools view -Sb -o SRR074164.bam SRR074164.map.uniq.head.sam
rm *.sam
rm SRR074164.header

bowtie2 -t -x rice_genome -U ../processed/SRR074165.Q20L15.fastq -S SRR074165.sam
grep -v '^@' SRR074165.sam | cut -f12,13 | sed -r 's/AS:i://' | sed -r 's/XS:i://' | awk '{ if ($1 =! $2) print $0}' | wc -l
samtools view -F 4 -S SRR074165.sam > SRR074165.map.sam
perl -ne 'chomp($_); @a = split(/\t/, $_); if ($a[11] ne $a[12]) { print "$_\n"}' SRR074165.map.sam > SRR074165.map.uniq.sam
samtools view -SH SRR074165.sam | sort > SRR074165.header
cat SRR074165.header SRR074165.map.uniq.sam > SRR074165.map.uniq.head.sam
samtools view -Sb -o SRR074165.bam SRR074165.map.uniq.head.sam
rm *.sam
rm SRR074165.header
```

3. **Simple Read Mapping**

```
bowtie2 -t -x rice_genome -U ../processed/SRR074166.Q20L15.fastq -S SRR074166.sam
grep -v '^@' SRR074166.sam | cut -f12,13 | sed -r 's/AS:i://' | sed -r 's/XS:i://' | awk '{ if ($1 =! $2) print $0}' | wc -l
samtools view -F 4 -S SRR074166.sam > SRR074166.map.sam
perl -ne 'chomp($_); @a = split(/\t/, $_); if ($a[11] ne $a[12]) { print "$_\n"}' SRR074166.map.sam > SRR074166.map.uniq.sam
samtools view -SH SRR074166.sam | sort > SRR074166.header
cat SRR074166.header SRR074166.map.uniq.sam > SRR074166.map.uniq.head.sam
samtools view -Sb -o SRR074166.bam SRR074166.map.uniq.head.sam
rm *.sam
rm SRR074166.header

bowtie2 -t -x rice_genome -U ../processed/SRR074167.Q20L15.fastq -S SRR074167.sam
grep -v '^@' SRR074167.sam | cut -f12,13 | sed -r 's/AS:i://' | sed -r 's/XS:i://' | awk '{ if ($1 =! $2) print $0}' | wc -l
samtools view -F 4 -S SRR074167.sam > SRR074167.map.sam
perl -ne 'chomp($_); @a = split(/\t/, $_); if ($a[11] ne $a[12]) { print "$_\n"}' SRR074167.map.sam > SRR074167.map.uniq.sam
samtools view -SH SRR074167.sam | sort > SRR074167.header
cat SRR074167.header SRR074167.map.uniq.sam > SRR074167.map.uniq.head.sam
samtools view -Sb -o SRR074167.bam SRR074167.map.uniq.head.sam
rm *.sam
rm SRR074167.header

bowtie2 -t -x rice_genome -U ../processed/SRR074168.Q20L15.fastq -S SRR074168.sam
grep -v '^@' SRR074168.sam | cut -f12,13 | sed -r 's/AS:i://' | sed -r 's/XS:i://' | awk '{ if ($1 =! $2) print $0}' | wc -l
samtools view -F 4 -S SRR074168.sam > SRR074168.map.sam
perl -ne 'chomp($_); @a = split(/\t/, $_); if ($a[11] ne $a[12]) { print "$_\n"}' SRR074168.map.sam > SRR074168.map.uniq.sam
samtools view -SH SRR074168.sam | sort > SRR074168.header
cat SRR074168.header SRR074168.map.uniq.sam > SRR074168.map.uniq.head.sam
samtools view -Sb -o SRR074168.bam SRR074168.map.uniq.head.sam
rm *.sam
rm SRR074168.header

bowtie2 -t -x rice_genome -U ../processed/SRR074169.Q20L15.fastq -S SRR074169.sam
grep -v '^@' SRR074169.sam | cut -f12,13 | sed -r 's/AS:i://' | sed -r 's/XS:i://' | awk '{ if ($1 =! $2) print $0}' | wc -l
samtools view -F 4 -S SRR074169.sam > SRR074169.map.sam
perl -ne 'chomp($_); @a = split(/\t/, $_); if ($a[11] ne $a[12]) { print "$_\n"}' SRR074169.map.sam > SRR074169.map.uniq.sam
samtools view -SH SRR074169.sam | sort > SRR074169.header
cat SRR074169.header SRR074169.map.uniq.sam > SRR074169.map.uniq.head.sam
samtools view -Sb -o SRR074169.bam SRR074169.map.uniq.head.sam
rm *.sam
rm SRR074169.header
```

Exercises:

1. **Basic Linux commands.**
2. **Sequencing evaluation.**
3. **Simple read mapping.**
4. **Simple de-novo assembly.**
5. **Basic R commands**
6. **Functional annotation.**
7. **Differential gene expression.**
8. **Cluster analysis for gene expression.**
9. **Selecting genes for phylogeny.**
10. **SNP calling and filtering.**
11. **Analysis of the population structure.**

4. **Simple de-novo assembly**

Goal: Assembly a transcriptomic dataset using SOAPdenovo-trans

Supporting information: http://soap.genomics.org.cn/SOAPdenovo-Trans.html

How to assembly reads with SOAP... quick guide:

1. Create the configuration file with the file names.

2. Run the assembly.

4. **Simple de-novo assembly**

Exercise List:

1. Prepare the SOAPdenovo configuration file for the reads SRR071161.nomap.fasta

2. Run SOAPdenovo with Kmer=15

3. Analyze the results with FastaSeqStats script

4. **Simple de-novo assembly**

Exercise List:

1. Prepare the SOAPdenovo configuration file for the reads SRR071161.nomap.fasta

```
max_rd_len=50
[LIB]
asm_flags=1
f=/home/bioinfo/Data/rice/exercise1/mapping/SRR071161.nomap.fasta
```

2. Run SOAPdenovo with Kmer=15

3. Analyze the results with FastaSeqStats script

Exercises:

1. **Basic Linux commands.**
2. **Sequencing evaluation.**
3. **Simple read mapping.**
4. **Simple de-novo assembly.**
5. **Basic R commands**
6. **Functional annotation.**
7. **Differential gene expression.**
8. **Cluster analysis for gene expression.**
9. **Selecting genes for phylogeny.**
10. **SNP calling and filtering.**
11. **Analysis of the population structure.**

5. **Basic R commands.**

Goal: Learn basic R commands.

Supporting information: http://www.slideshare.net/aubombarely/introduction2r

Exercise List:

1. Open the program R-Studio.
2. Sum 2 + 7 and assign the result to the object 'a'
3. Create the vector 'b' with consecutive elements from 1 to 100
4. Create the vector 'c' with consecutive elements from 1 to 100 in reverse order
5. Create the matrix 'm' of 2 rows and 100 columns with 'b' and 'c'
6. Create the transpose of the matrix 'm'.
7. Prepare a datatse with the locus.gff file: cut -f1,4,5 locus.gff > locus_coord.txt

5. **Basic R commands.**

Exercise List:

8. Load the locus_coord.txt file in R using read.delim() function

9. Summarize the dataframe created with the locus_coord.txt file.

10. Create a table with the frequencies of the first column (chromosomes) of that dataframe.

11. Create a graph (barplot) with the number of genes per chromosome

Exercises:

1. **Basic Linux commands.**
2. **Sequencing evaluation.**
3. **Simple read mapping.**
4. **Simple de-novo assembly.**
5. **Basic R commands**
6. **Functional annotation.**
7. **Differential gene expression.**
8. **Cluster analysis for gene expression.**
9. **Selecting genes for phylogeny.**
10. **SNP calling and filtering.**
11. **Analysis of the population structure.**

6. **Functional Annotation.**

Goal: Assign function to a dataset using Blast2GO

Supporting information: http://www.blast2go.com/b2glaunch/resources

Exercise List:

1. Using FastaExtract create a file with all the contigs from the SOAPdenovo-trans assembly with a length > 25.

2. Open Blast2GO and load the previous fasta file.

3. Run the Blast annotation to identify those transcripts.

Exercises:

1. **Basic Linux commands.**
2. **Sequencing evaluation.**
3. **Simple read mapping.**
4. **Simple de-novo assembly.**
5. **Basic R commands**
6. **Functional annotation.**
7. **Differential gene expression.**
8. **Cluster analysis for gene expression.**
9. **Selecting genes for phylogeny.**
10. **SNP calling and filtering.**
11. **Analysis of the population structure.**

7. **Differential Gene Expression.**

Goal: Analyze the differential gene expression of two datasets using TopHat/Cufflinks

Supporting information:

http://www.nature.com/nprot/journal/v7/n3/full/nprot.2012.016.html
http://tophat.cbcb.umd.edu/manual.html
http://cufflinks.cbcb.umd.edu/manual.html

Sample Source:

SRR074161: Cypress-HighMilling-Developing seeds-6days-Rep1
SRR074162: Ilpumbyeo-HighTaste-Developing seeds-6days-Rep1
SRR074163: Ilpumbyeo-HighTaste-Developing seeds-6days-Rep2
SRR074164: LaGrue-LowMilling-Developing seeds-6days-Rep1
SRR074165: LaGrue-LowMilling-Developing seeds-6days-Rep2
SRR074166: Nipponbare-Control-Developing seeds-6days-Rep1
SRR074167: Nipponbare-Control-Developing seeds-6days-Rep2
SRR074168: YR15965-LowTaste-Developing seeds-6days-Rep1
SRR074169: YR15965-LowTaste-Developing seeds-6days-Rep2

7. **Differential Gene Expression.**

Exercise List:

1. Run Cufflinks (When the analysis doesn't have replicates, Cufflinks assumes that most of the genes doesn't change the expression between conditions, so it can be used to calculate de variance) over the samples. Remember that the SAM file need to be ordered and the header need to have the order @HD, @PD, @SG.

2. Run Cuffmerge with the results from the last step.

3. Run Cuffdiff to analyze the differential expression of the different samples.