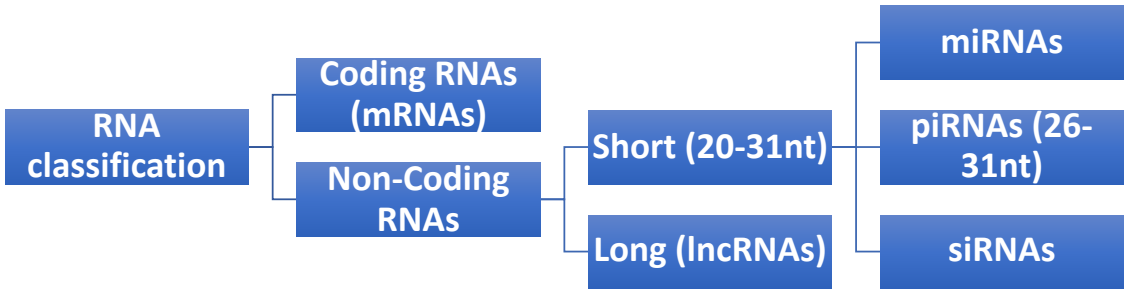
Toolkit for piRNA identification in small RNA-seq libraries

Program name: “piRNA_smallRNAseq_analysis.py”

UNIVERSITAT DE VIC
UNIVERSITAT CENTRAL
DE CATALUNYA

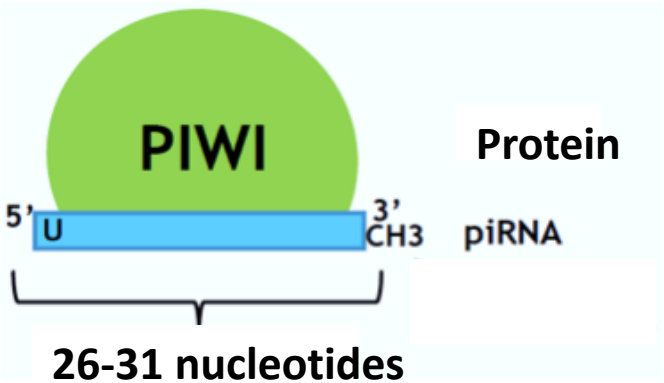
Natalia Llonga Tejedor

Piwi-Interacting RNA (piRNA)



Functions:

- Silencing Transposable elements
- Silencing some mRNAs



Species	Number of piRNAs annotated
Drosophila melanogaster	21.027.419
Bombyx mori	1.174.963
Homo sapiens	32.826

Data analysis

Problems that we can have:

- Big libraries (around 1-2GB of space disk)
- Spend a lot of time between each step
- The high number of libraries

How to solve it?

Toolkit for piRNA identification in small RNA-seq libraries

- Facilitate the work
- Reduce the time used in each step

piRNA_smallRNAseq_analysis.py

How the script Works

Version: Python 3.6.4 under macOS 10.13.4

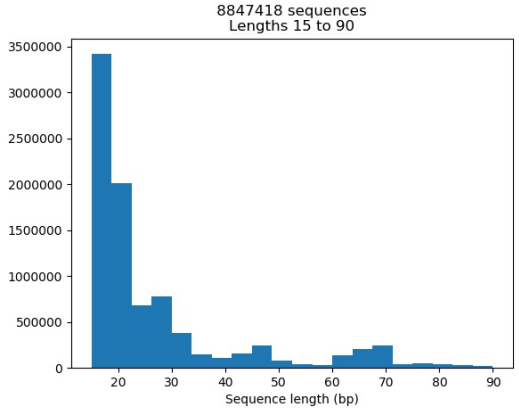
Modules used:

```
import glob, os, sys
from Bio import SeqIO
import pylab
```

Dependences:

-Bowtie
-Bowtie2
- Samtools
- List of miRNAs in fasta format
- Reference genome
- All libraries of small RNA-seq in fastq format

Data visualization



Command to use: python piRNA_smallRNAseq_analysis.py

Examples of functions used:

```
def fastq_to_fasta(file):
    ''' This function converts the fastq files to fasta files
    fastq > fasta
    '''
    fileout=str(file)+".fasta"
    SeqIO.convert(file, "fastq", fileout, "fasta")
    print("Fastq to Fasta Done!")
```

```
for file in glob.glob("*.fastq"):
    print (file)
    fastq_to_fasta(file)
```

```
def fasta_select_seqs(file2, min_length, max_length):
    ''' This function is used to select reads by their length, from a fasta
    and returns other fasta file with the sequences
    fasta > fasta
    '''
    fileout=str(file2)+"26-31.fasta"
    selected_sequences=[]
    for record in SeqIO.parse(file2, "fasta"):
        if len(record.seq)>= min_length and len(record.seq)<= max_length:
            selected_sequences.append(record)
    SeqIO.write(selected_sequences, fileout, "fasta")
    print("Select reads by lenght Done!")
```

```
for file2 in glob.glob("*fastq.fasta"):
    fasta_select_seqs(file2, 26, 31)
```

```
def plot_selected_seqs(file3):
    '''This function is used to show the sequence length distribution
    from a fasta file
    fasta > jpeg
    '''
    fileout=str(file3)+".jpeg"
    sizes = [len(rec) for rec in SeqIO.parse(file3, "fasta")]
    pylab.hist(sizes, bins=20)
    pylab.title(" \n %i sequences \nLengths %i to %i" \
        % (len(sizes), min(sizes), max(sizes)))
    pylab.xlabel("Sequence length (bp)")
    pylab.ylabel("Count")
    #pylab.show()
    pylab.savefig(fileout)

    print("Histograms done!")
```

```
def remove_miRNAs(file3):
    '''
    This function is used to remove miRNA reads from multifasta file, using Bowtie2
    fasta > SAM > BAM > Fastq '''
    COMMAND = ("bowtie2 -L 18 -N 0 -p 2 -x miRNAs_bowtie2 -f "+str(file3)+" -S "+str(file3)+".SAM")
    print(COMMAND)
    COMMAND2 = ("samtools view -S "+str(file3)+".SAM -f4 > "+str(file3)+"nomiRNA.SAM")
    COMMAND3 = ("samtools view -bS "+str(file3)+"nomiRNA.SAM > "+str(file3)+"nomiRNA.BAM")
    COMMAND4 = ("samtools bam2fq "+str(file3)+"nomiRNA.BAM > "+str(file3)+"nomiRNA.fastq")
    os.system(COMMAND)
    os.system(COMMAND2)
    os.system(COMMAND3)
    os.system(COMMAND4)
    print ("miRNAs removed")
```

How to improve the script: Allowing the user to choose specific functions.

Difficulties in the creation of the script

- To test the script
- To implement libraries
- To design the outputs in each step
- To have enough nemory to make all the test in the disk

How I solved it:

- Accessing to server
- Using faster functions for some tasks
- Combining python with Shell

• An example of use inputs / outputs

```python
def fastq_to_fasta(file):
    ''' This function converts the fastq files to fasta files
    fastq > fasta
    '''
    fileout=str(file)+".fasta"
    SeqIO.convert(file, "fastq", fileout, "fasta")
    print("Fastq to Fasta Done!")


def fasta_select_seqs(file2, min_length, max_length):
    ''' This function is used to select reads by their length, from a fasta file
    and returns other fasta file with the sequences
    fasta > fasta
    '''
    fileout=str(file2)+"26-31.fasta"
    selected_sequences=[]
    for record in SeqIO.parse(file2, "fasta"):
        if len(record.seq)>= min_length and len(record.seq)<= max_length:
            selected_sequences.append(record)
    SeqIO.write(selected_sequences, fileout, "fasta")
    print("Select reads by lenght Done!")

def plot_selected_seqs(file3):
    '''This function is used to show the sequence length distribution
    from a fasta file
    fasta > jpeg
    '''
    fileout=str(file3)+".jpeg"
    sizes = [len(rec) for rec in SeqIO.parse(file3, "fasta")]
    pylab.hist(sizes, bins=20)
    pylab.title(" \n %i sequences \nLengths %i to %i" \
        % (len(sizes), min(sizes), max(sizes)))
    pylab.xlabel("Sequence length (bp)")
    pylab.ylabel("Count")
    #pylab.show()
    pylab.savefig(fileout)

    print("Histograms done!")
```

```python
#1 The first step is obtain fasta files from fastq
print("Starting the process ... ")
#os.chdir("/Users/nataliallonga/Desktop/FinalProjectBioinf") #Establish the working directory

for file in glob.glob("*.fastq"):
    print (file)
    fastq_to_fasta(file)

#2 Plot some graphs in order to  check the information

for file2 in glob.glob("*fastq.fasta"):
    print(file2)
    print(plot_selected_seqs(file2))

#3 Select reads of interest. I use piRNA fraction that corresponds between 26 to 31 nucleotides of length

for file2 in glob.glob("*fastq.fasta"):
    fasta_select_seqs(file2, 26, 31)
```

For each library:

Fastq

↓

Fasta

↓

Fasta with reads between 26-31 nucleotides of lenght

Plot lenghts fasta reads