BIOINFORMATICS INSTITUTE

Development of a library of functions for express analysis of FASTA/FASTQ files

Supervisor: Evgeny Bakin

Students: Alena Kizenko
Alisa Morshneva
Polina Pavlova

General goals of the project

1. Improve our skills at programming in Python.
2. Try Biopython for performing routine bioinformatic tasks.
3. Create a tool for a ***rapid*** fasta/fastq data analysis.
4. **Compare different ways of biopython iteration over FASTA/FASTQ files for a speed and memory improvement*

Workflow

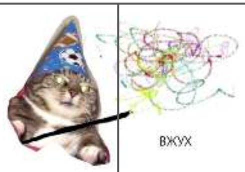
Master argparse, biopython, unittest, pandas, matplotlib, os etc.

↓

Write base of our tool and function modules.

↓

Function testing, time optimization, writing the tool documentation.



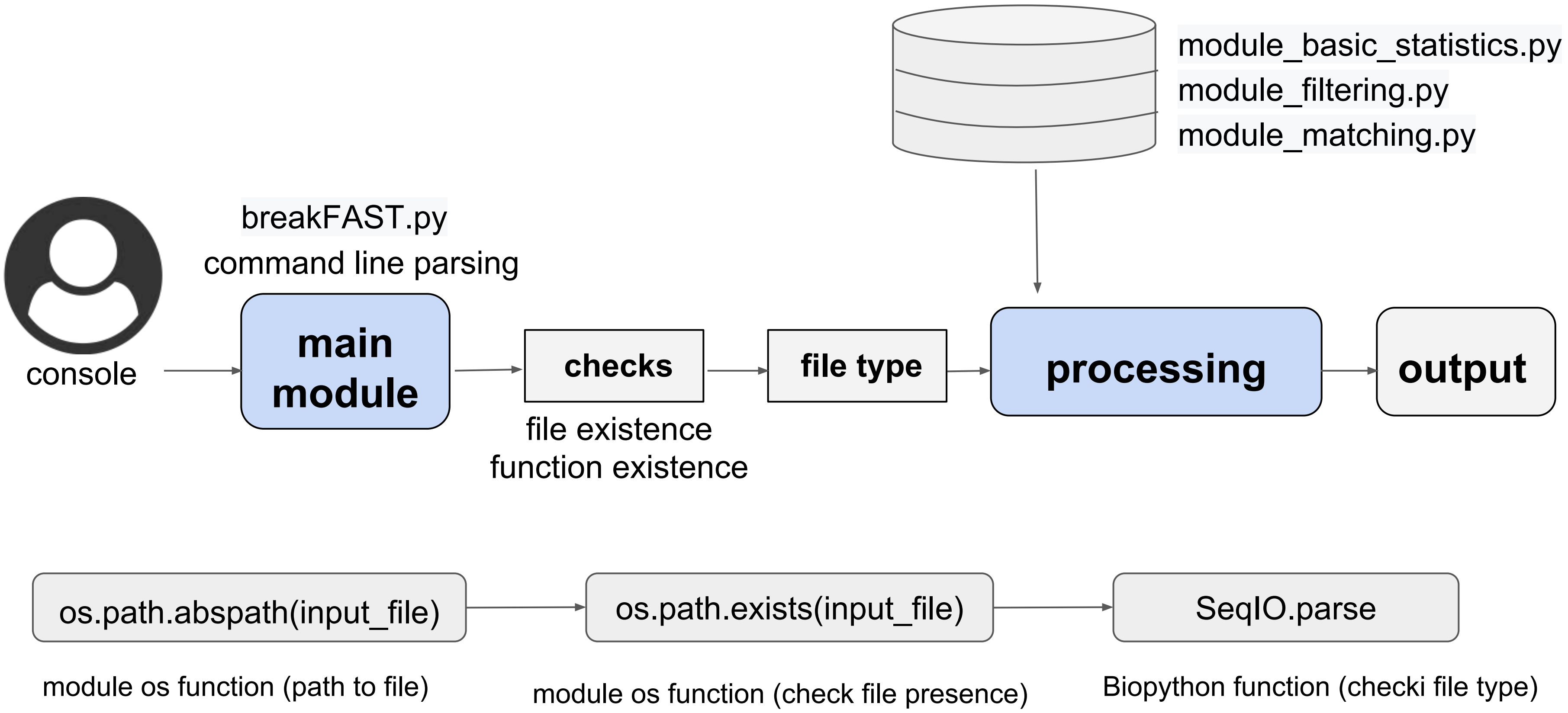
↓

The tool is ready!

What we have done

BreakFAST

The BreakFAST's structure


module_basic_statistics.py
module_filtering.py
module_matching.py
breakFAST.py
command line parsing
console
main module
checks
file existence
function existence
file type
processing
output
os.path.abspath(input_file)
os.path.exists(input_file)
SeqIO.parse
module os function (path to file)
module os function (check file presence)
Biopython function (checki file type)

We have learned such python libraries as biopython, argparse, pandas, numpy, matplotlib, memory profiler etc.

biopython

matplotlib

NumPy

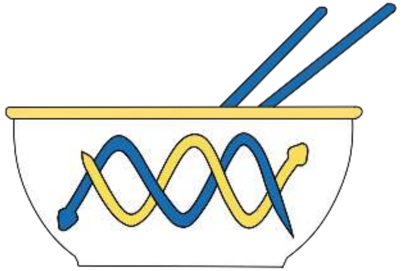
Pandas

What our tool does?

1. **Basic statistics**
   a) minimum, maximum, mean, total length
   b) GC content
   c) quality scores
   d) N base count

2. **Filtering**
   a) delete reads shorter than X
   b) delete reads containing Ns
   c) delete poor quality reads
   d) delete duplicates
   e) delete reads with a particular motif

3. **Matching files**
   a) join reads from several files
   b) find overlapping between several files
   c) subtract sets of reads from several files



Speed problem

Biopython is really cool, but a little bit slow…
That`s why we decided to investigate how to improve speed of fasta/fastq file analysis!

FastqGeneralIterator vs SeqIO.parser

FastqGeneralIterator

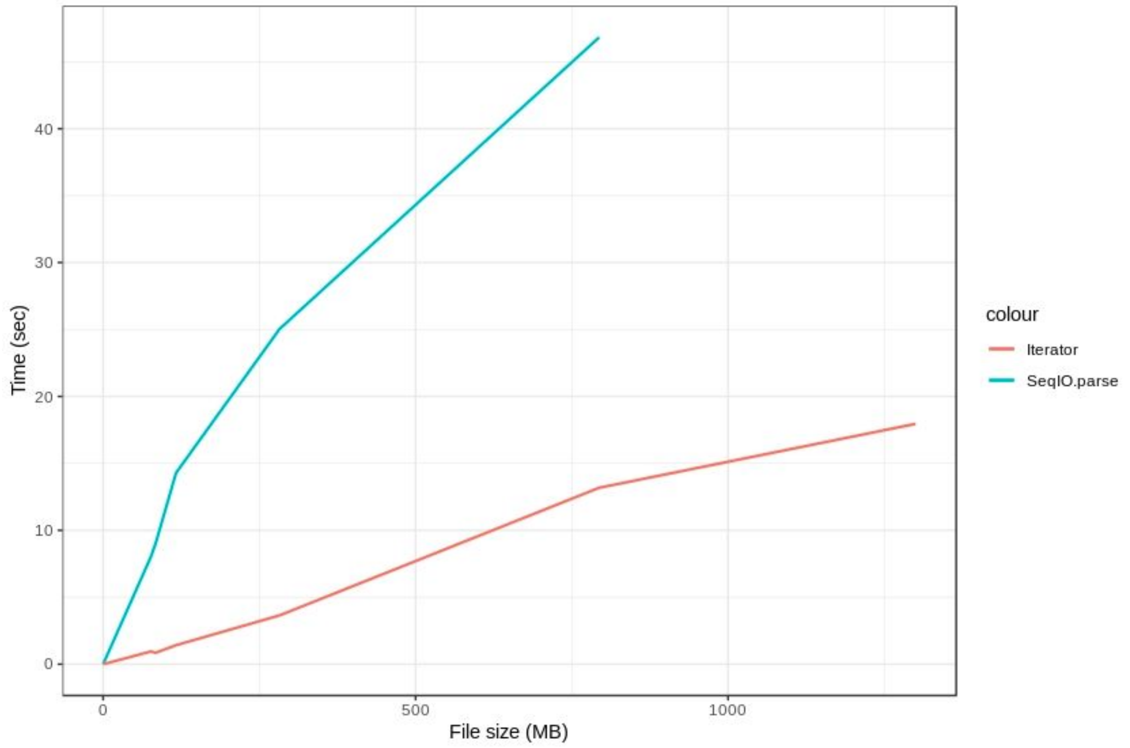
+ Works quickly
- Works only with a text part of fasta/fastq files, can define only title, sequence and quality lines in fastq file, other features must be written in code

SeqIO.parser

+ A lot of features for easy definition of file type, the read identificators, sequence, quality, phred etc.
- Works quite slowly

FastqGeneralIterator vs SeqIO.parser

def min_length



Time test

file size	Iterator	SeqIO.parser
76.5 MB	0.4989	8.43739
84.2MB	0.85635	9.5126
116,7 MB	1.42077	14.5419
282.3MB	3.6462	33.83207
793.7MB	13.17788	121.4919
1.3GB	17.9527	164.3023

Memory

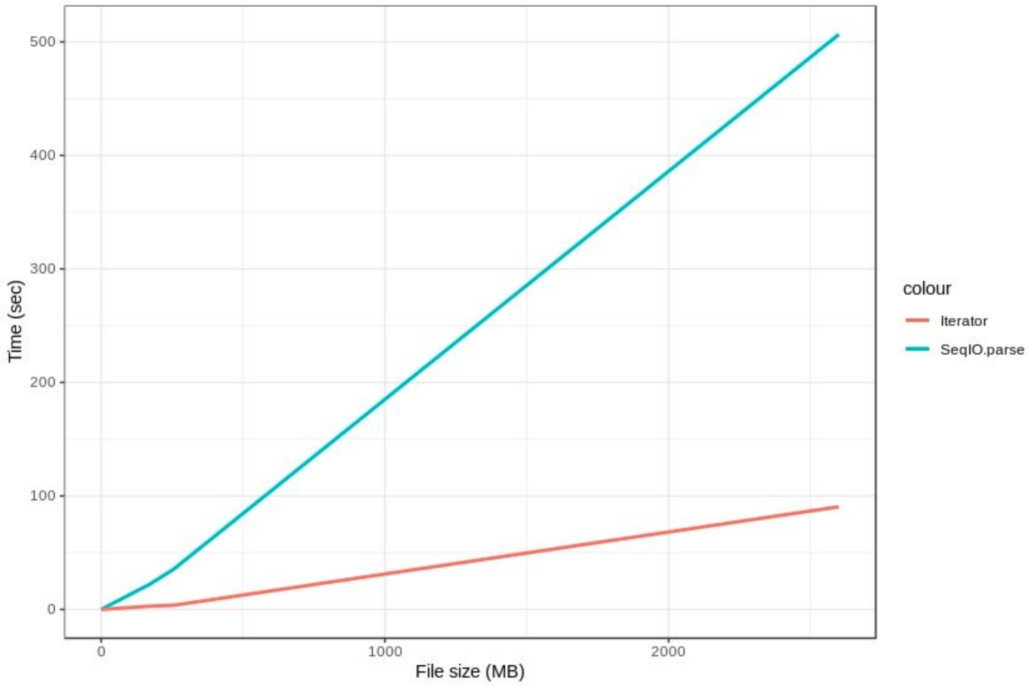
file size	Iterator	Trimmomatic
1.3Gb	~90Mb	~700Mb

Time test

file size	Iterator	Trimmomatic
1.3Gb	17.95	7

FastqGeneralIterator vs SeqIO.parser

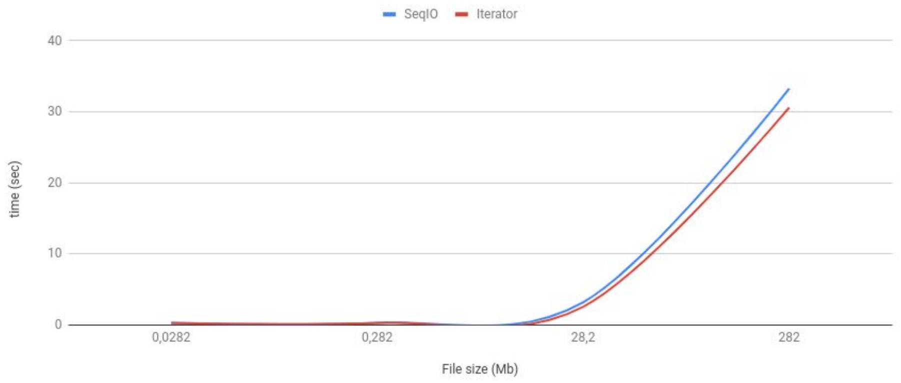
join_sequences()



File size	Time, seconds	
	Iterator	SeqIO.parser
0,024	0,00075	0,00083
170	2,84217	22,02943
256	3,67418	35.5297
2600	90,47383	506.6893

FastqGeneralIterator vs SeqIO.parser

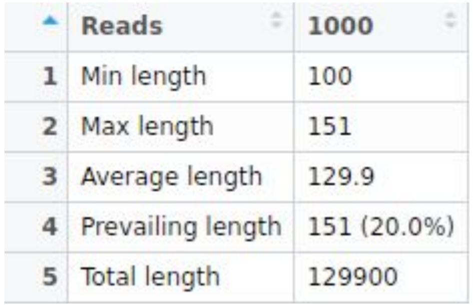
quality_score()



Number of sequences	Time, seconds	
	Iterator	SeqIO.parser
10	0,26	0,27
10000	0,29	0,29
100000	2,56	3,18
1000000	30,6	33,25

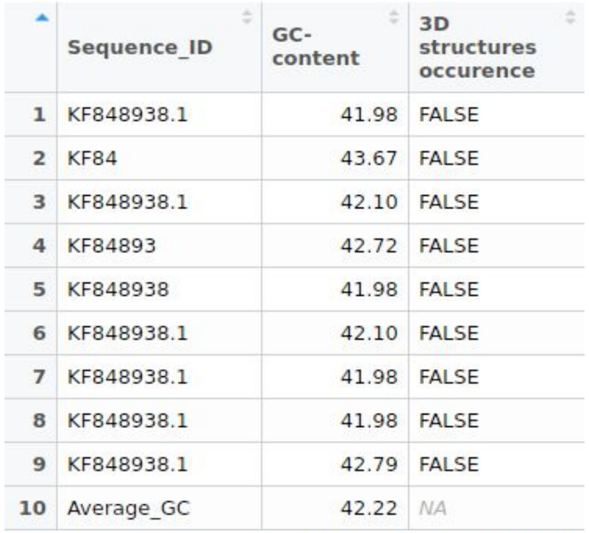
Basic statistics functions - examples

basic_statistics()



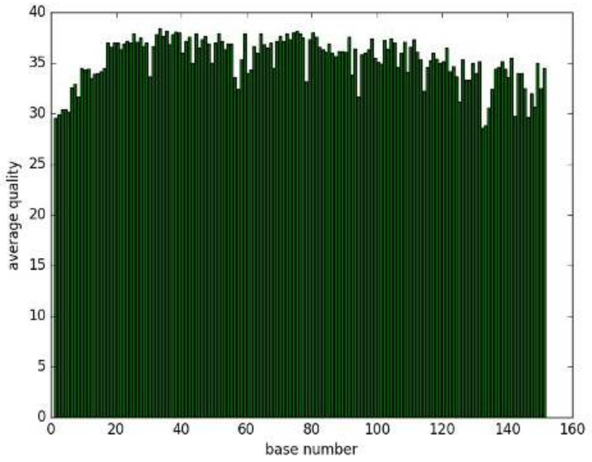
	Reads	1000
1	Min length	100
2	Max length	151
3	Average length	129.9
4	Prevailing length	151 (20.0%)
5	Total length	129900

gc_content_analysis()



	Sequence_ID	GC-content	3D structures occurence
1	KF848938.1	41.98	FALSE
2	KF84	43.67	FALSE
3	KF848938.1	42.10	FALSE
4	KF84893	42.72	FALSE
5	KF848938	41.98	FALSE
6	KF848938.1	42.10	FALSE
7	KF848938.1	41.98	FALSE
8	KF848938.1	41.98	FALSE
9	KF848938.1	42.79	FALSE
10	Average_GC	42.22	NA

quality_score()



```
     average quality
1         29.6
2         29.9
3         30.4
4         30.4
5         30.2
6         32.6
7         32.9
```

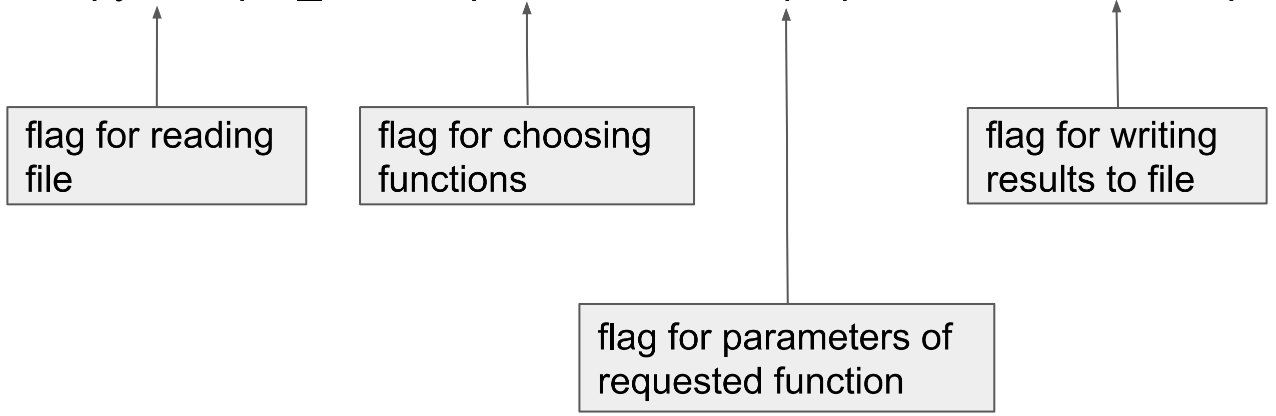
n50()

```
contigs.fasta
Only for contigs longer than 500
N50: 698474
```

Using argparse to read options from the command line

Command-line input structure:

python3 breakFAST.py -i <*input_file.fastq*> -f <*function*> -p <*parameters*> -o <*output_file.csv*>


flag for reading file
flag for choosing functions
flag for writing results to file
flag for parameters of requested function

Example of using

```
python3 breakFAST.py -i  fastq_test.fastq -f quality_score -p 33 -o result
```

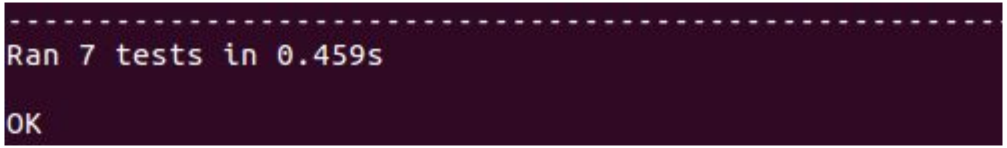
```
python3 breakFAST.py -i fasta_test1.fasta -f join_sequences -p fasta_test2.fasta -o result
```

```
python3 breakFAST.py -i fastq_test.fastq -f min_length -p 113 -o result
```

We’ve tested the tool’s functions with Unit-test

Unit-test

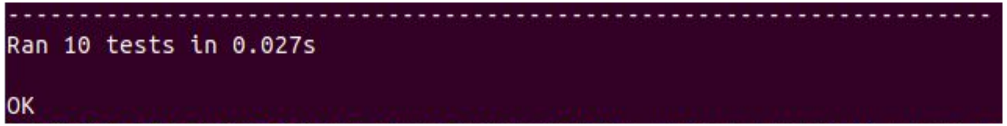
python3 -m unittest unittest_for_basic_statistics.py



```
----------------------------------------------------------------------
Ran 7 tests in 0.459s

OK
```

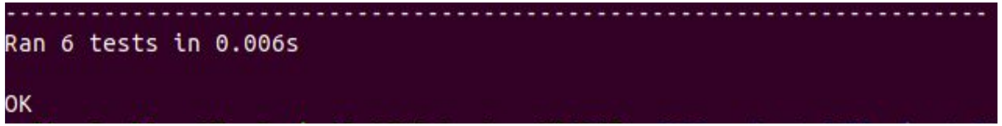
python3 -m unittest unittest_for_filtering.py



```
----------------------------------------------------------------------
Ran 10 tests in 0.027s

OK
```

python3 -m unittest unittest_for_matching.py



```
----------------------------------------------------------------------
Ran 6 tests in 0.006s

OK
```

Thanks for your attention!

We look forward to receiving your commits!

Use existing tool
Write your own tool using SeqIO.parse()
Write your own tool using Iterator
Write your own tool using dictionaries