cwiczenie1 - kontrola jakosci_odpowiedzi

April 25, 2023

1 Ćwicznia 1 - kontrola jakosci

Link przydatny w czasie wykonywania cwiczen: https://en.wikipedia.org/wiki/FASTQ_format

Środowisko pracy

Wszystkie programy niezbędne do wykonania ćwiczeń znajdują się na serwerze.

- Serwer: 150.254.120.213
- Użytkownik: linux
- Hasło: podane na zajeciach

Folder z danymi: `~/QC`

Znajdziesz tu katalog `data` oraz pliki z zadaniami

PAMIĘTAJ ABY WSZELKIE ZMIANY NA DANYCH WYKONYWAĆ W SWOIM PRYWATNYM FOLDERZE

Na serwerze zostało stworzone śrdowisko condy, z zainstalowanym programem FASTQC oraz . Aby je uruchomić wpisz: `conda activate qc`

Protokół

Na protokół składać się będzie plik tekstowy z odpowiedziami na pytania zamieszczone w rozpisce (wyłącznie te, które będą wskazane w rozpisce) oraz plik ze skryptem do zadania 7. Pliki należy na koniec ćwiczeń przesłać na adres marta.szustakowska@amu.edu.pl, w tytule wiadomości wpisując *Imię, nazwisko oraz datę ćwiczeń.*

Pliki

Pliki używane w dzisiejszym dniu zawierają:

- **single-end.fastq** oraz **single-end2.fastq** – Wyniki sekwencjonowania bibliotek cDNA opartej o małe RNA, długości 18-30 nt, sekwencjonowanie wykonywane było na 3 zmultipleksowanych próbkach
- **paired-end_1.fastq** oraz **paired-end_2.fastq** – Wyniki sekwencjonowania z dwóch końców biblioteki cDNA opracowanej według protokołu RNA-seq

1.1 Zadanie 1

Podana jest sekwencja w formacie FASTQ:

```
@HWI-EAS209_0006_FC706VJ:5:58:5894:21141#ATCACG/1
ACGTGCATAGCTAGCATTACGATACGAGCCGGCGGCAAATATAGCG
+
ACDFAAEHHHBBAA?;<<CDEAD98<;;<ADE??<874873012,/
```

Korzystając ze wzorów i informacji podanych na wykładzie oblicz i zapisz do protokołu:

1. W jakim systemie kodowania jest zapisana jakość tej sekwencji (phred +33 czy phred +64)?
2. Która pozycja ma najniższą jakość a która najwyższą?
3. Jaka jest średnia jakość dla tego odczytu?
4. Ile jest pozycji o jakości poniżej 20 w skali phred? Jaki procent odczytu one stanowią?
5. Czy obserwujesz spadek jakości na końcu odczytu?
6. Jakie jest prawdopodobieństwo, że nukleotyd został błędnie odczytany, jeśli jego jakość wynosi Q=28

1. W jakim systemie kodowania jest zapisana jakość tej sekwencji (phred +33 czy phred +64)?

[56]:
```
# %load solutions/solution1.py
characters = "ACDFAAEHHHBBAA?;<<CDEAD98<;;<ADE??<874873012,/."

'''for char in characters:
    out = ord ( str(char) ) - 33
    print(out, end = ' ')
'''

ph33 = [ord(char) - 33 for char in characters]
ph64 = [ord(char) - 64 for char in characters]
if any(i <0 for i in ph33):
    if all( i > 0 for i in ph64):
        print("Jakość tej sekwencji zapisana jest w systemie phred +64")
        print(ph64)
    else:
        print("Jakość tej sekwencji nie odpowiada ani skali phred +33 ani +64")
else:
    print("Jakość tej sekwencji zapisana jest w systemie phred +33")
    print(ph33)
```

```
Jakość tej sekwencji zapisana jest w systemie phred +33
[32, 34, 35, 37, 32, 32, 36, 39, 39, 39, 33, 33, 32, 32, 30, 26, 27, 27, 34, 35,
36, 32, 35, 24, 23, 27, 26, 26, 27, 32, 35, 36, 30, 30, 27, 23, 22, 19, 23, 22,
18, 15, 16, 17, 11, 14, 13]
```

2. Która pozycja ma najniższą jakość a która najwyższą?

[42]:
```
# %load solutions/solution2.py
import numpy as np

out_list = []
for char in characters:
    out = ord ( str(char) ) - 33
```

```
    out_list.append(out)

mini = min(out_list)
maxi = max(out_list)
'''
if sum([i == maxi for i in ph33]) > 1:
    print("Nukleotydy o najwyższej jakosci znajdują sie na pozycjach {}".
 ↪format([i + 1 for i in np.where(np.array(ph33) == maxi)[0]]))
else:
    print("Nukleotyd o najwyższej jakosci znajduje sie na pozycji {}".format(␣
 ↪out_list.index( maxi)+1 ))

if sum([i == mini for i in ph33]) > 1:
    print("Nukleotydy o najniższej jakosci znajdują sie na pozycjach {}".
 ↪format([i + 1 for i in np.where(np.array(ph33) == mini)[0]]))
else:
    print("Nukleotyd o najnizszej jakosci znajduje sie na pozycji {}".format(␣
 ↪out_list.index( mini)+1 ))
'''

print("Nukleotyd o najnizszej jakosci znajduje sie na pozycji {}".format(␣
 ↪out_list.index( mini)+1 ))
print("Nukleotyd o najwyższej jakosci znajduje sie na pozycji {}".format(␣
 ↪out_list.index( maxi)+1 ))
```

```
Nukleotyd o najnizszej jakosci znajduje sie na pozycji 45
Nukleotyd o najwyższej jakosci znajduje sie na pozycji 8
```

3. Jaka jest średnia jakość dla tego odczytu?

[43]:
```
# %load solutions/solution3.py
import numpy as np
round(np.mean(out_list),2)
```

[43]:
```
28.15
```

4. Ile jest pozycji o jakości poniżej 20 w skali phred? Jaki procent odczytu one stanowią?

[51]:
```
# %load solutions/solution4.py
'''out_array = np.asarray(out_list)
print(out_array[out_array<20].shape[0]) # liczba elementow o jakosci ponizej 20
print((out_array[out_array<20].shape[0]/out_array.shape[0])*100) # jaka czesc␣
 ↪wszystkich stanowia (w %)'''

print("Jest {} pozycji o jakości poniżej 20 w skali phred".format(len(np.
 ↪where(np.array(ph33) < 20)[0])))
print("stanowią one {}% wszystkich nukleotydów".format(round(len(np.where(np.
 ↪array(ph33) < 20)[0])/len(ph33) * 100,2)))
```

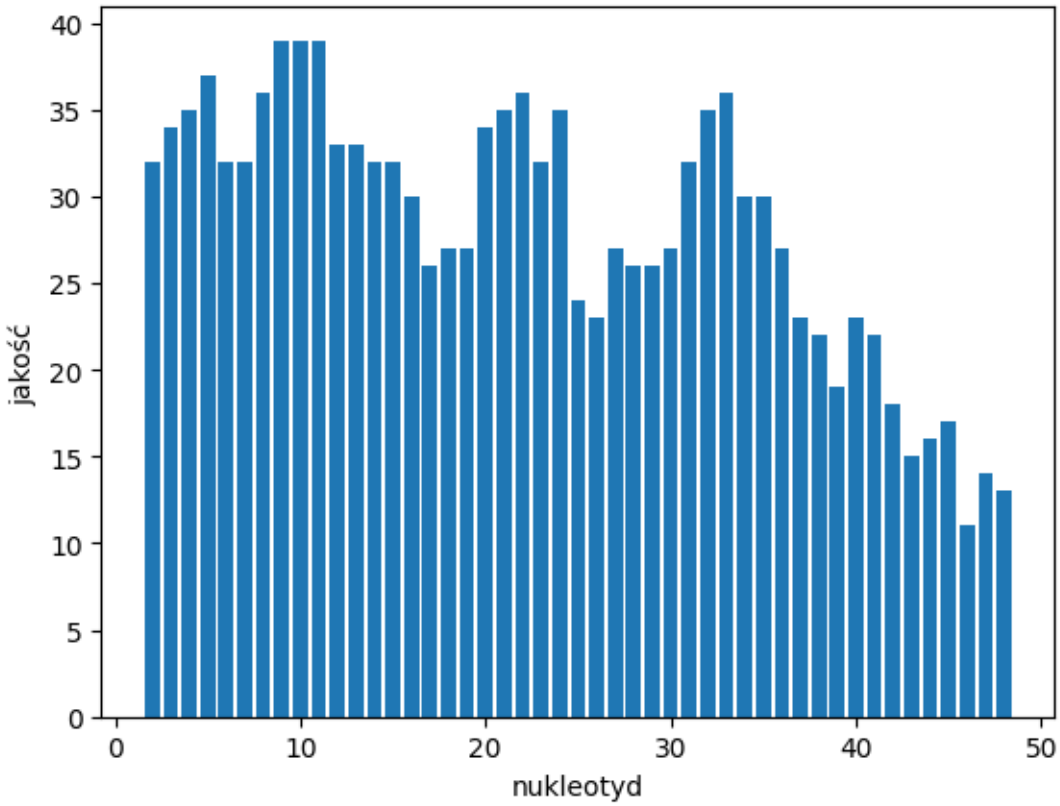
```
Jest 8 pozycji o jakości poniżej 20 w skali phred
stanowią one 17.02% wszystkich nukleotydów
```

5. Czy obserwujesz spadek jakości na końcu odczytu?

[52]:
```
# %load solutions/solution5.py
import matplotlib.pyplot as plt

x = [i+1 for i in range(1,len(ph33)+1)]

plt.bar(x,ph33)
plt.ylabel('jakość')
plt.xlabel('nukleotyd')
plt.show()
```



6. Jakie jest prawdopodobieństwo, że nukleotyd został błędnie odczytany, jeśli jego jakość wynosi Q=28

$$Q = -10 \log_{10} P$$

$$\frac{Q}{-10} = \log_{10} P$$

Uzywajac reguly, mowiacej ze

$$a = \log_b(b^a)$$

mozemy zapisac

$$\log_{10} 10^{\frac{Q}{-10}} = \log_{10} P$$

dalej, korzystajac z zaleznosci

$$\log_b f(x) = \log_b g(x) \Rightarrow f(x) = g(x)$$

otrzymujemy

$$10^{\frac{Q}{-10}} = P$$

Przyklad dla Q = 20:

$$10^{-2} = P$$

$$P = \frac{1}{100}$$

[55]:
```
# %load solutions/solution6.py
10**(28/-10)
```

[55]:
```
0.001584893192461114
```

1.2 cwiczenie 2

Wykonaj analizę jakości sekwencji znajdujących się w pliku data/single-end.fastq za pomocą programu fastqc. Program uruchamia się domyślnie w środowisku graficznym (skrypt fastqc), bądź z linii komend (lista opcji: fastqc –help). Przeanalizuj otrzymany raport oraz odpowiedz do protokołu na pytania:

Jaka jest zawartość duplikatów?

Czy jest znacząca liczba odczytów o średniej jakości poniżej 25?

Czy i jeśli tak, to w której pozycji 1 i 3 kwartyl dystrybucji jakości spada poniżej wartości 20?

Czy występują jakieś nadreprezentowane sekwencje? Jeśli tak, to jakie jest pochodzenie najbardziej nadreprezentowanej (użyj sekwencji, która nie została oznaczona jako RNA PCR Primer i wykorzystaj NCBI Blast, aby to zbadać)?

Niebieska linia - średnia Czerwone kreski - mediana drugi kwartyl (50%) Żółty kwadrat - lower i upper quartile (25% i 75% - pierwszy i trzeci kwartyl) “wąsy” - 10 (dziesiąty) i 90 percentile

For example, the 50th percentile (median) is the score below (or at or below, depending on the definition) which 50% of the scores in the distribution are found.

1.3 cwiczenie 3

Usuń adaptery z 3' końców odczytów (sekwencja: TGGAATTCTCGGGTGCCAAGG). W tym celu zastosuj narzędzie cutadapt. Lista opcji dostępna jest z przełącznikiem „–help". Pomocy możesz szukać także tutaj https://cutadapt.readthedocs.io/en/stable/guide.htmlNajpierw zastosuj domyślne (ale ustaw minimalną długość odczytu po usunięciu adaptara na 15) ustawienia i zwróć uwagę na statystyki dotyczące długości usuniętego adaptora. Następnie uruchom narzędzie po raz drugi, ustawiając wartość minimalnej długości adaptora tak, aby proces był specyficzny (liczba wykrytych adaptorów wyższa od oczekiwanej z losowego rozkładu). Ustaw również odpowiednie opcje tak, aby w pliku wynikowym znajdowały się tylko te sekwencje, z których został usunięty adaptor. Zapisz do protokołu liczbę sekwencji w plikach wynikowych.

[90]:
```
# %load solutions/solution7.py
import os
os.system("cutadapt -a TGGAATTCTCGGGTGCCAAGG -m 15 -o cutadapt_out1.fastq data/
 ↪single-end.fastq")
os.system("cutadapt -a TGGAATTCTCGGGTGCCAAGG -O 8 --trimmed-only -m 15 -o␣
 ↪cutadapt_out2.fastq data/single-end.fastq")
# zliczanie liczby rekordow wc -l i dzielenie przez 4
```

```
This is cutadapt 2.8 with Python 3.8.10
Command line parameters: -a TGGAATTCTCGGGTGCCAAGG -m 15 -o cutadapt_out1.fastq
data/single-end.fastq
Processing reads on 1 core in single-end mode …
Finished in 0.25 s (4 us/read; 15.15 M reads/minute).

=== Summary ===

Total reads processed:                  63,670
Reads with adapters:                    59,848 (94.0%)
Reads that were too short:               8,962 (14.1%)
Reads written (passing filters):        54,708 (85.9%)

Total basepairs processed:     3,247,170 bp
Total written (filtered):      1,257,017 bp (38.7%)

=== Adapter 1 ===

Sequence: TGGAATTCTCGGGTGCCAAGG; Type: regular 3'; Length: 21; Trimmed: 59848
times; Reverse-complemented: 0 times

No. of allowed errors:
0-9 bp: 0; 10-19 bp: 1; 20-21 bp: 2

Bases preceding removed adapters:
  A: 15.6%
  C: 32.0%
  G: 28.2%
```

```
  T: 24.2%
  none/other: 0.0%

Overview of removed sequences
length  count   expect  max.err error counts
3       98      994.8   0       98
4       36      248.7   0       36
5       26      62.2    0       26
6       28      15.5    0       28
7       25      3.9     0       25
8       30      1.0     0       30
9       52      0.2     0       52
10      33      0.1     1       32 1
11      38      0.0     1       34 4
12      41      0.0     1       39 2
13      36      0.0     1       36
14      26      0.0     1       26
15      42      0.0     1       35 7
16      50      0.0     1       44 6
17      64      0.0     1       53 11
18      49      0.0     1       46 3
19      66      0.0     1       59 7
20      193     0.0     2       186 7
21      105     0.0     2       100 5
22      292     0.0     2       283 9
23      389     0.0     2       375 13 1
24      614     0.0     2       560 49 5
25      936     0.0     2       884 48 4
26      1326    0.0     2       1266 57 3
27      11309   0.0     2       10910 374 25
28      2851    0.0     2       2735 109 7
29      4213    0.0     2       3916 280 17
30      7354    0.0     2       7030 307 17
31      1765    0.0     2       1661 99 5
32      2318    0.0     2       2199 115 4
33      1686    0.0     2       1581 103 2
34      2259    0.0     2       2124 125 10
35      9901    0.0     2       9132 733 36
36      2635    0.0     2       2429 203 3
37      2036    0.0     2       1901 127 8
38      867     0.0     2       824 42 1
39      404     0.0     2       388 15 1
40      498     0.0     2       454 42 2
41      421     0.0     2       402 18 1
42      598     0.0     2       569 28 1
43      957     0.0     2       860 90 7
44      697     0.0     2       642 51 4
45      789     0.0     2       740 43 6
```

```
46      524     0.0     2       458 61 5
47      178     0.0     2       154 24
48      62      0.0     2       59 3
49      45      0.0     2       43 2
50      112     0.0     2       108 4
51      774     0.0     2       710 63 1
This is cutadapt 2.8 with Python 3.8.10
Command line parameters: -a TGGAATTCTCGGGTGCCAAGG -O 8 --trimmed-only -m 15 -o
cutadapt_out2.fastq data/single-end.fastq
Processing reads on 1 core in single-end mode …
Finished in 0.26 s (4 us/read; 14.68 M reads/minute).

=== Summary ===

Total reads processed:                  63,670
Reads with adapters:                    59,635 (93.7%)
Reads that were too short:               8,962 (14.1%)
Reads written (passing filters):        50,673 (79.6%)

Total basepairs processed:     3,247,170 bp
Total written (filtered):      1,052,143 bp (32.4%)

=== Adapter 1 ===

Sequence: TGGAATTCTCGGGTGCCAAGG; Type: regular 3'; Length: 21; Trimmed: 59635
times; Reverse-complemented: 0 times

No. of allowed errors:
0-9 bp: 0; 10-19 bp: 1; 20-21 bp: 2

Bases preceding removed adapters:
  A: 15.6%
  C: 32.0%
  G: 28.2%
  T: 24.2%
  none/other: 0.0%

Overview of removed sequences
length  count   expect  max.err error counts
8       30      1.0     0       30
9       52      0.2     0       52
10      33      0.1     1       32 1
11      38      0.0     1       34 4
12      41      0.0     1       39 2
13      36      0.0     1       36
14      26      0.0     1       26
15      42      0.0     1       35 7
16      50      0.0     1       44 6
```

```
17      64      0.0     1       53 11
18      49      0.0     1       46 3
19      66      0.0     1       59 7
20      193     0.0     2       186 7
21      105     0.0     2       100 5
22      292     0.0     2       283 9
23      389     0.0     2       375 13 1
24      614     0.0     2       560 49 5
25      936     0.0     2       884 48 4
26      1326    0.0     2       1266 57 3
27      11309   0.0     2       10910 374 25
28      2851    0.0     2       2735 109 7
29      4213    0.0     2       3916 280 17
30      7354    0.0     2       7030 307 17
31      1765    0.0     2       1661 99 5
32      2318    0.0     2       2199 115 4
33      1686    0.0     2       1581 103 2
34      2259    0.0     2       2124 125 10
35      9901    0.0     2       9132 733 36
36      2635    0.0     2       2429 203 3
37      2036    0.0     2       1901 127 8
38      867     0.0     2       824 42 1
39      404     0.0     2       388 15 1
40      498     0.0     2       454 42 2
41      421     0.0     2       402 18 1
42      598     0.0     2       569 28 1
43      957     0.0     2       860 90 7
44      697     0.0     2       642 51 4
45      789     0.0     2       740 43 6
46      524     0.0     2       458 61 5
47      178     0.0     2       154 24
48      62      0.0     2       59 3
49      45      0.0     2       43 2
50      112     0.0     2       108 4
51      774     0.0     2       710 63 1
```

[90]: 0

kolumna expect mówi o tym, jaka szacowana liczba sekwencji może się dopadować do adaptera przypadkowo. Czyli im mniejsze expect tym bardziej prawdopodobne, że sekwnecja faktycznie jest adaptorem 1. 54 708 2. 50 673

1.4 cwiczenie 4

Wykonaj filtrowanie i skracanie odczytów pod względem jakości. W tym celu użyj programów z zestawu narzędzi fastx. Dostępne opcje można wyświetlać używając przełącznika „-h”. Wykonaj następujące kroki: 1. Użyj programu fastq_quality_trimmer w celu usunięcia nukleotydów o jakości poniżej 20 z 3’ końca odczytów. Pozostaw tylko sekwencje dłuższe niż 20 nt. 2. Używając

programu fastq_quality_filter pozostaw tylko odczyty które zawierają więcej niż 90% zasad o jakości powyżej 20. Zanotuj do protokołu liczbę sekwencji które pozostały po filtrowaniu.

[93]:
```
# %load solutions/solution8.py
os.system("fastq_quality_trimmer -t 20 -l 20 -i cutadapt_out2.fastq -o␣
 ↪fastx_out1.fastq")
os.system("fastq_quality_filter -p 90 -q 20 -i fastx_out1.fastq -o fastx_out2.
 ↪fastq")
```

```
CommandNotFoundError: Your shell has not been properly configured to use 'conda
activate'.
To initialize your shell, run

    $ conda init <SHELL_NAME>

Currently supported shells are:
  - bash
  - fish
  - tcsh
  - xonsh
  - zsh
  - powershell

See 'conda init --help' for more information and options.

IMPORTANT: You may need to close and restart your shell after running 'conda
init'.


sh: 1: fastq_quality_trimmer: not found
sh: 1: fastq_quality_filter: not found
```

[93]:
```
32512
```

1.5 cwiczenie 5

Wykonaj ponowną analizę jakości za pomocą programu fastqc. Porównaj wyniki z analizą przeprowadzoną przed filtrowaniem. Zanotuj do protokołu, które parametry uległy poprawie.

1.6 cwiczenie 6

Powtórz wszystkie etapy dla plików paired-end_1.fastq oraz paired-end_2.fastq. Ze względu na fakt, że są to odczyty typu paired-end: 1. Po wstępnej analizie jakości zdecyduj czy konieczne jest usuwanie adaptorów z 3' końca. 2. Na koniec napisz skrypt, który odfiltruje do osobnych plików odczyty, które nie mają swojej pary. Zanotuj do protokołu: 1. zaobserwowane różnice w wynikach analizy jakości w odniesieniu do pliku single-end.fastq 2. ilość odczytów będących parami pozostałych po całym procesie przygotowania , oraz ilości pojedynczych odczytów z odfiltrowanych

z plików _1 oraz _2

[10]:
```
# %load solutions/solution9.py
os.system("fastq_quality_trimmer -t 20 -i data/paired-end_1.fastq -o␣
 ↪fastx_out1pe1.fastq")
os.system("fastq_quality_trimmer -t 20 -i data/paired-end_2.fastq -o␣
 ↪fastx_out1pe2.fastq")
os.system("fastq_quality_filter -p 90 -q 20 -i fastx_out1pe1.fastq -o␣
 ↪fastx_out2pe1.fastq")
os.system("fastq_quality_filter -p 90 -q 20 -i fastx_out1pe2.fastq -o␣
 ↪fastx_out2pe2.fastq")
```

[10]: 0

[11]:
```
# %load solutions/solution10.py
from Bio import SeqIO

p1, p2 = {}, {}

fastq_parser1 = SeqIO.parse("fastx_out2pe1.fastq", "fastq")
for fastq_rec in fastq_parser1:
    p1[fastq_rec.name[:-2]] = (fastq_rec, fastq_rec.seq, fastq_rec.
 ↪letter_annotations)

fastq_parser2 = SeqIO.parse("fastx_out2pe2.fastq", "fastq")
for fastq_rec in fastq_parser2:
    p2[fastq_rec.name[:-2]] = (fastq_rec, fastq_rec.seq, fastq_rec.
 ↪letter_annotations)

handle_paired1 = open("paired_p1.fastq", "w")
handle_paired2 = open("paired_p2.fastq", "w")
handle_single1 = open("single_p1.fastq", "w")
handle_single2 = open("single_p2.fastq", "w")
for key, value in p1.items():
    if key in p2.keys():
        SeqIO.write( (p1[key])[0], handle_paired1, "fastq")
        SeqIO.write( (p2[key])[0], handle_paired2, "fastq")
        # Ma pare. Zapisz do pliku
    else:
        SeqIO.write( (p1[key])[0], handle_single1, "fastq")
        # Nie ma pary. Zapisz do pliku

for key, value in p2.items():
    if key not in p2.keys():
        SeqIO.write( (p2[key])[0], handle_single2, "fastq")
        # Nie ma pary. Zapisz do pliku
```

```
handle_paired1.close()
handle_paired2.close()
handle_single1.close()
handle_single2.close()
```

1.7 Zadanie 7

Poszukaj gdzie na serwerze znajdziesz folder programu fastqc:

1. Uruchom środowisko condy, w którym został zainstalowany program
2. Wpisz `whereis fastqc`
3. Wykonaj polecenie listujące wskazany folder `ls -l path/to/file`, pamiętaj, że prawdopodobnie ścieżka wskazuje na plik wykonywalny a nie na folder
4. Na liście wyników znajdź program fastqc (`grep` lub manualnie), obok jego nazwy powinna znajdować się odpowiednia ścieżka. Wyświetl zawartość katalogu Configuration.

Napisz skrypt, który: 1. wykona analizę jakości programem FASTQC pliku single-end2.fastq (folder *data*); 2. na podstawie informacji w pliku fastqc_data.txt z folderu wynikowego (single-end2_fastqc) ustali, jaki adapter należy usunąć z pliku single-end2.fastq; 3. pobierze odpowiednią sekwencję adaptera z pliku adapter_list.txt, który znajduje się w znalezionym przez Ciebie wcześniej katalogu Configuration 4. uruchomi program cutadapt i wytnie adapterz z PLIK_TODO 5. wykona ponowną analizę programem FASTQC

Sprawdź, czy w otrzymanych wynikach nie znajduje się już usunięty adapter.

[7]:
```
# %load solutions/solution12.py
#conda activate qc
#whereis fastqc
#ls -l /home/linux/anaconda3/envs/qc/bin/ | grep fastqc
#ls -l /home/linux/anaconda3/envs/qc/opt/fastqc-0.12.1/Configuration/


import os
import statistics as st

file = "data/single-end2.fastq"
adapter_list = "/etc/fastqc/Configuration/adapter_list.txt"
#adapter_list = "/home/linux/anaconda3/envs/qc/opt/fastqc-0.12.1/Configuration/
 ↪adapter_list.txt"

#os.system("fastqc --extract {}".format(file))
adapters = {}
with open ("data/single-end2_fastqc/fastqc_data.txt") as data:
    for line in data:
        if line.startswith(">>Adapter"):
            header = next(data).strip().split("\t")
            for h in header:
                adapters[h] = []
            line = next(data)
```

```
            while not line.startswith(">>"):
                values = line.strip().split("\t")
                for i in range(len(header)):
                    adapters[header[i]].append(float(values[i]))
                line = next(data)

adapter_delete = ""
for adapter in adapters:
    m = 0.0
    if adapter != '#Position':
        indexes = [ n for n,i in enumerate(adapters[adapter]) if i>0.0 ]
        if len(indexes) > 0:
            ix = indexes[0]
            tmp = st.mean(adapters[adapter][ix:])
            if tmp > m:
                m = tmp
                adapter_delete = adapter

print(adapter_delete)

with open (adapter_list) as l:
    for line in l:
        if adapter_delete in line:
            seq = line.strip().split()[-1]

print("Usuniety powinien zostać adapter {} o sekwencji {}\n".
 ↪format(adapter_delete, seq))
out_file = "cutadapt_out_7.fastq"
#os.system("cutadapt -a {} -m 15 -o {} data/single-end2.fastq".format(seq,␣
 ↪out_file))
#os.system("fastqc {}".format(out_file))
```

```
Illumina Small RNA 3' Adapter
Usuniety powinien zostać adapter Illumina Small RNA 3' Adapter o sekwencji
TGGAATTCTCGG
```