NGS Coursework - Part 2

Ioannis Valasakis[1]

[1]Birkbeck, University of London

April 4, 2019

Background

In 2011, a very infectious strain of *E. coli* spread through Germany. Nearly 4000 people were affected and over 50 people died as a result. In such cases, the focus is often on identifying potential drug targets among the genes of the new strain. Here is carried genome annotation of this strain and examine the conservation of virulent genes in this and other strains of *E. coli*.

Question 1

In order to further examine those genes the virulent data are obtained using the VFDB core data database, and downloading the fasta file using wget:

```
wget http://www.mgc.ac.cn/VFs/Down/VFDB_setA_nt.fas.gz
```

and extracting it, in order to use the *FASTA* file.

```
gunzip VFDB_setA_nt.fas.gz
```

This was in a very bad state though, so the newlines from each line of a sequence had to be removed, like that:

```
awk '!/^>/ { printf "%s", $0; n = "\n" }
/^>/ { print n $0; n = "" }
END { printf "%s", n }
' VFDB_setA_nt.fas > VFDB_no_line.fas
```

In order to filter the genome with our required E.Coli the following command is executed:

```
cat VFDB_no_line.fas | grep -A1 -E 'Escherichia|EHEC|EAEC|ETEC|UPEC' | sed '/^--$/d' > \
selected_virulent_ecoli.fas
```

which redirects the filtered`stdout`to another *FASTA* file. This would be further filtered in order to remove some fields as only the > *FASTA* header, the first gene acronym "Met" and the antibiotic family "NIMB" are needed.

In order to generate the correct headers more filtering is needed (gene name, family name). This has to read in the sequences file and re-create another *FASTA*!

Now that is an impressive regex!

```
#!/usr/bin/python

import re
import sys
import fileinput

with open('selected_virulent_ecoli.fas', 'r') as input_file, open('awesome.fas', 'w') as output_file:
    for line in input_file:
        if '>' in line:
            txt = line
            re1='.*?'   # Non-greedy match on filler
            re2='.' # Uninteresting: c
            re3='.*?'   # Non-greedy match on filler
            re4='.' # Uninteresting: c
            re5='.*?'   # Non-greedy match on filler
            re6='.' # Uninteresting: c
            re7='.*?'   # Non-greedy match on filler
            re8='.' # Uninteresting: c
            re9='.*?'   # Non-greedy match on filler
            re10='.'    # Uninteresting: c
            re11='.*?'  # Non-greedy match on filler
            re12='.'    # Uninteresting: c
            re13='.*?'  # Non-greedy match on filler
            re14='.'    # Uninteresting: c
            re15='.*?'  # Non-greedy match on filler
            re16='.'    # Uninteresting: c
            re17='.*?'  # Non-greedy match on filler
            re18='.'    # Uninteresting: c
            re19='.*?'  # Non-greedy match on filler
            re20='.'    # Uninteresting: c
            re21='.*?'  # Non-greedy match on filler
            re22='.'    # Uninteresting: c
            re23='.*?'  # Non-greedy match on filler
            re24='.'    # Uninteresting: c
            re25='.*?'  # Non-greedy match on filler
            re26='.'    # Uninteresting: c
            re27='.*?'  # Non-greedy match on filler
            re28='.'    # Uninteresting: c
            re29='.*?'  # Non-greedy match on filler
            re30='.'    # Uninteresting: c
            re31='.*?'  # Non-greedy match on filler
            re32='.'    # Uninteresting: c
            re33='.*?'  # Non-greedy match on filler
            re34='.'    # Uninteresting: c
            re35='.*?'  # Non-greedy match on filler
            re36='.'    # Uninteresting: c
            re37='.*?'  # Non-greedy match on filler
            re38='.'    # Uninteresting: c
            re39='.*?'  # Non-greedy match on filler
            re40='.'    # Uninteresting: c
```

```
    re41='.*?'  # Non-greedy match on filler
    re42='.'    # Uninteresting: c
    re43='.*?'  # Non-greedy match on filler
    re44='.'    # Uninteresting: c
    re45='.*?'  # Non-greedy match on filler
    re46='.'    # Uninteresting: c
    re47='.*?'  # Non-greedy match on filler
    re48='.'    # Uninteresting: c
    re49='.*?'  # Non-greedy match on filler
    re50='.'    # Uninteresting: c
    re51='.*?'  # Non-greedy match on filler
    re52='.'    # Uninteresting: c
    re53='.*?'  # Non-greedy match on filler
    re54='.'    # Uninteresting: c
    re55='.*?'  # Non-greedy match on filler
    re56='.'    # Uninteresting: c
    re57='(.)'  # Any Single Character 1
    re58='((?:[a-z][a-z]+))'    # Word 1
    re59='(.)'  # Any Single Character 2
    re60='.*?'  # Non-greedy match on filler
    re61='(\\[)'    # Any Single Character 3
    re62='((?:[a-z][a-z]+))'    # Word 2
    re63='(\\s+)'   # White Space 1
    re64='(\\()'    # Any Single Character 4
    re65='((?:[a-z][a-z]*[0-9]+[a-z0-9]*))' # Alphanum 1
    re66='(.)'  # Any Single Character 5

    rg = re.compile(re1+re2+re3+re4+re5+re6+re7+\
    re8+re9+re10+re11+re12+re13+re14+re15+re16+\
    re17+re18+re19+re20+re21+re22+re23+re24+re25\
    +re26+re27+re28+re29+re30+re31+re32+re33+re34\
    +re35+re36+re37+re38+re39+re40+re41+re42+re43\
    +re44+re45+re46+re47+re48+re49+re50+re51+re52\
    +re53+re54+re55+re56+re57+re58+re59+re60+re61\
    +re62+re63+re64+re65+re66,re.IGNORECASE|re.DOTALL)
    m = rg.search(txt)
    if m:
        c1=m.group(1)
        word1=m.group(2)
        c2=m.group(3)
        c3=m.group(4)
        word2=m.group(5)
        ws1=m.group(6)
        c4=m.group(7)
        alphanum1=m.group(8)
        c5=m.group(9)

        output = '>' + word1+ ' ' + word2 + ws1 + alphanum1 + '\n'

    output_file.write(output)
else:
```

```
            output_file.write(line)
```

In order to extract virulence genes from the annotated genome, gene family names were extracted from the above file.

```
cat awesome.fas | awk 'BEGIN {FS=" "}  {print $2}' > gene_family_names
```

In order to access the annotated genome file, that is in the directory of the departmental `hope`

server, in the location `cd ${st_path}/results_GC/annotation`

Having an idea of the gene family names of the virulent genes, these can now be searched and extracted from the annotated genome (`annotation.gff`).

```
cat annotation.gff | grep -E 'viru | toxin | esp | sepl | pet | aff | agg' \
| awk 'BEGIN {FS="\t"}  split($9, captured, /[(=);]/) >=10  \
{print "sequence1" "\t" $4 "\t" $5 "\t" captured[10] "\t" captured[4] "\t" $7}' > virulent.bed
```

This is the format required to convert to *BED* file.

```
sequence1   49143   49520   cbtA_1  toxin of toxin-antitoxin system -
sequence1   246137  249610  bcsC    cellulose synthase subunit  +
sequence1   430042  430359  BAG79191.1 toxin RelE   toxin RelE  -
sequence1   662864  663328  LOCUS_06440 toxin of the SohB   -
sequence1   850343  850720  cbtA_2  toxin of toxin-antitoxin system -
sequence1   1007024 1007431 cptA    toxin of CptAB toxin-antitoxin pair +
sequence1   1143399 1143734 mazF    mRNA interferase toxin  +
sequence1   1216928 1217347 ygjM    transcriptional regulator   +
sequence1   1293916 1294392 ratA    ubiquinone-binding protein  +
sequence1   1448310 1448576 mvpT    post-segregational killing toxin    +
sequence1   1951524 1951778 yoeB    toxin of the YoeB-YefM toxin-antitoxin system   +
sequence1   1964435 1964698 yeeV    toxin of the YeeV-YeeU toxin-antitoxin system   -
sequence1   2625880 2626146 hipB    antitoxin of HipAB toxin-antitoxin system   +
sequence1   2626146 2627468 hipA    regulator with hipB +
sequence1   3177621 3178073 mchD    hypothetical protein    -
sequence1   3271578 3271847 stx2B   Shiga toxin 2 subunit B -
sequence1   3271859 3272818 stx2A   Shiga toxin 2 subunit A -
sequence1   4152682 4152960 yafQ    toxin-antitoxin pair YafQ/DinJ  +
sequence1   4625905 4626255 chpB    toxin of the ChpB-ChpS toxin-antitoxin system   -
sequence1   4907377 4907754 cbtA_3  toxin-antitoxin system  +
sequence1   4918986 4919159 ghoT    toxin of GhoTS toxin-antitoxin pair -
```

Once we have the annotation and genomic position a fasta is generated using `bedtools`:

```
 /s/software/bedtools/v2.27.1/bin/bedtools getfasta -name -s -fi\
 ${st_path}/results_GC/annotation/genome.fna -bed \
 virulent.bed -fo present_virulent.fas
```

Now the two *FASTA* files are merged:

```
cat selected_virulent_ecoli.fasta present_virulent.fas > easy_peasy.fas
```

Now using *BRIG* to annotate these genes on the genome of different strains of E. Coli!

```
cp ${st_path}/results_GC/annotation/genome.fna \
${st_path}/results_GC/wholeGenomeExamples/AFPN02.1.genome.fasta
```

Then the software is run with the preset settings from the Coursework 2. Unfortunately this couldn't be completed and generate a ring (as can be seen in Fig 1).

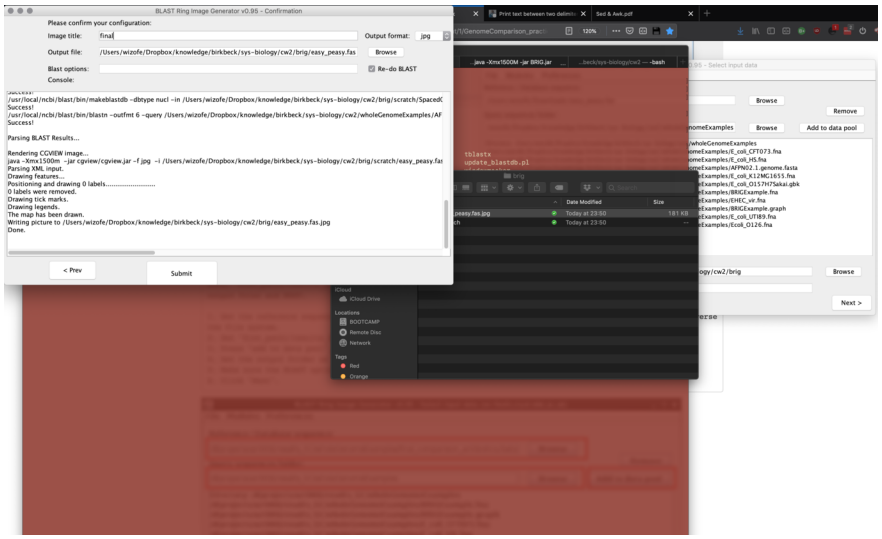

Figure 1: BRIG, Step 1

As the ....network is down!

```
iris:~ wizofe$ ssh vi001@hope-ext.cryst.bbk.ac.uk
ssh: connect to host hope-ext.cryst.bbk.ac.uk port 22: Network is down
```