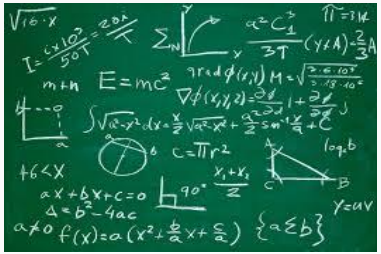
Little bioinformatician’s pragmatic guide to internships in Debian

By Tatiana Malygina & Nadiya Sitdykova

What is bioinformatics?

Biology



Mathematics

Computer Science

- Genomics
- Transcriptomics
- Proteomics
- Structural Biology
- Genome assembly
- Molecular sequence analysis
- etc...

Bioinformatics in Debian

The Debian Med Pure Blend contains 880 packages which are grouped by metapackages. Here are some metapackages related to bioinformatics:

- Biology
- Next Generation Sequencing
- Phylogeny
- Laboratory
- Research
- Statistics
- etc...

Internships in Debian

Internship	 Google Summer of Code	OUTREACHY
When?	Summer	Summer & Winter
How many projects in Debian to choose from?	20 (summer 2016)	4 (summer 2016)
Who is eligible?	Students	People from underrepresented groups

How did we learn about it and our motivation to join

1. Outreachy is not so commonly known as GSoC. We learnt about it through friend who knew someone who knew someone … who participated ("sarafan radio").

2. Here is some reasons that made an Outreachy a perfect choice:

    - Welcomes such newbies as we are
    - Doesn’t require to be a student as other internships
    - Helps to overcome “first contribution anxiety”
    - Real opportunity to resolve known problems of bioinformatics software

What did we learn: Don’t hesitate to ask the community

“The story of incognito option”

It happened during testing fastx-toolkit package...

```
fastq_quality_filter -i fastq_qual_filter1.fastq -q 33 -p 100
```

FAIL

```
fastq_quality_filter: bug: got empty array at fastq_quality_filter.c:97
```

...some investigation ...confusion ...hesitation to ask for help ...finally writing to Debian Med List … got an answer:

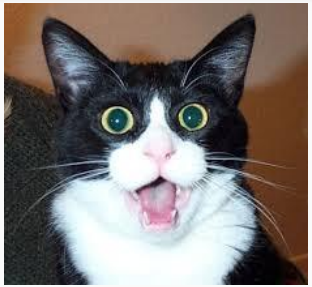
>Hi Nadiya,

>FASTX-toolkit has an undocumented option `-Q` to set the quality offset.

```
fastq_quality_filter -Q64 -i fastq_qual_filter1.fastq -q 33 -p 100
```

PASS

What did we learn: Don’t skip system requirements



FAIL

“The story of enormous allocation”

It happened during testing kraken package...

```
kraken-build --build --db test_db
```

```
terminate called after throwing an instance of 'std::bad_alloc'
  what():  std::bad_alloc
/usr/lib/kraken/build_kraken_db.sh: line 155: 12644 Aborted
```

...some investigation ...locating the place of bad allocation:

```
uint64_t entries = 1ull << (nt * 2);
```

where nt = 15 by default, so it’s trying to allocate 2147483648 bytes! No surprise it’s bad alloc...but why so much?

System Requirements:
Memory: *The default database size is 75 GB (as of Feb. 2015), and so you will need at least that much RAM*

Luckily after creating my own mini-database I was able to run kraken-build with custom nt parameter

```
kraken-build --build --minimizer-len 5 --db test_db
```

PASS

What did we learn: elementary, my dear Watson!

“The story of mysterious metastudent”

It happened when metastudent package tests at CI servers began to fail out of the random:

FAIL

```
installation-test    FAIL non-zero exit status 1
```

...some investigation ...dependency had changed recently from legacy-blast to blast+... run metastudent like this:

```
metastudent -i test.fasta -o test.result --temp-dir=. --keep-temp
```

...million times, located all metastudent code fragements, which depend on blast output format,

Fixed regular expressions in all code fragments to match blast+ output to parse it correctly.

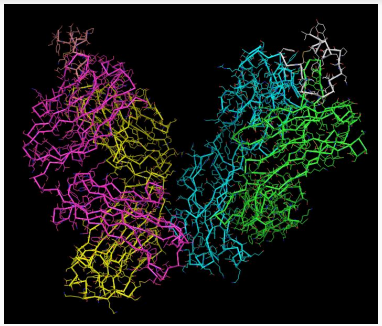
Run this again:

```
metastudent -i test.fasta -o test.result --temp-dir=. --keep-temp
```

...until final metastudent output began to look correct...

PASS

What did we learn: legacy issues



“The stories of code reuse and the cruel scientist, Time”

It's hard to write good tests for something you like. But PyMOL has examples/ folder with different scripts in it! So what if ... reuse?

Write a small script in bash and run all these scripts one by one, fix if there are any errors.

...as a bonus, you can provide a good set of updated (and perfectly working!) examples for package users.

There is also a great package autopkgtest-perl made by Perl Group, which finds tests in perl packages and runs it. Saved us some time with perl packages.

Sometimes authors of old packages don't answer letters or communicate when you need it.

Sometimes you find forgotten pieces of code with comments made by people who definitely was struggling while working on it. Feels like time travel!

Sometimes when you fix errors you learn how to write better code.

What did we learn:

A good mentor is half the battle!

- Answers your questions in time
- Helps to deal with self-inconfidence
- Supports
- His name is Andreas Tille

Thanks!