**HIV drug resistance**

Natural selection is the process by which some form of selective pressure favours a **phenotypic** trait or change. These phenotypic traits can be the blood group of a person, whether a pea is wrinkly or not, or whether an infectious organism is susceptible or resistant to a drug. Many times these phenotypic traits, or physical attributes, are caused by genetics.

**Genotyping** is the process by which one can infer this phenotypic trait from a genotype, and this is used more and more frequently in medicine. For exampe, in breast cancer treatment, the BRCA (BReast CAncer) genes are genotyped to determine whether these cancer suppressing genes are intact. If there is a deleterious or damaging mutation in one of these genes, it can increase the risk of developing breast cancer, thus a phenotype of increased risk of breast cancer.

For most organisms, the copying of genetic material happens by very precise enzymes or pathways, but occasionally mutations do occur. If a mutation occurs and is sufficiently damaging, it gets removed from the gene pool. However, if the mutation is sufficiently beneficial, it increases the survival of this genetic variation and might biasly select for it.

We already discussed **ARVs** (antiretrovirals) and how these drugs changed the landscape of HIV infection by preventing the development of AIDS. We mentioned that ARVs suppress viral replication. One of the steps in HIV replication is the conversion of its single-stranded RNA to DNA, which can then be incorporated into the DNA of infected cells. The enzyme responsible for this conversion is reverse transcriptase, and it has a high error rate when doing this conversion. One can thus say that HIV has a high evolutionary rate, or mutation rate. These genes are translated into viral proteins, which are required to make more virions (viral particles). Proteins are strings or polymers of amino acid residues with an alphabet of 20 choices of amino acids or letters. The sequence of the DNA or RNA influences the sequence of the protein; thus, mutations in the DNA or RNA can result in changes in the protein, and our targets for stopping HIV replication are proteins/enzymes.

There are various classes of ARVs which interfere with viral replication by inhibition of viral enzymes. If the DNA or RNA sequence encoding this enzyme is changed, the result might be an unfit virus not capable of further infection or replication. On the other hand, if this mutation results in an ARV-resistant virus, replication and infection can still continue in the presence of the ARV in question, possibly causing the ARV to become ineffective in stopping replication.

The question remains, why do people develop resistance? The short answer: it’s a numbers game.

If the patient received the correct regimen of ARVs (known as **HAART**, or highly active antiretroviral treatment) and is taking the doses correctly, the viral load will suppress. Suppression is caused by stopping viral replication, and if the virus is not replicating, the error-prone reverse transcriptase can’t cause mutations, which in turn cannot be favoured by selective pressure. If the patient is not taking any treatment, the virus is replicating and thus inevitably mutating, but there is no selective pressure to select for these variants. Lastly, if the patient is adhering poorly to the treatment, there are times where the levels of the treatment are too low to effectively suppress viral replication completely. In this scenario, mutants with a mutation which makes them less susceptible to the treatment will replicate more than the wild type counterparts – these are called escape mutants.

The reason why this is a numbers game is that the virus is mutating randomly and one resulting amino acid residue could be replaced by any of 19 other amino acid residues. It is only when this change causes an increase in replicative fitness while there is some form of selective pressure that this mutant can become a dominant quasi-species and the patient develops resistance.

Mutations are expressed using the notation [WT AA][POS][Mutant AA], where:

* WT denotes wild type (the typical genotype)
* AA denotes amino acid residue
* POS denotes the position on the protein
* Mutant means the changed genotype

We mentioned some classes of ARVs in part 1. To the viral reverse transcriptase, **NRTIs** (Nucleoside/Nucleotide Reverse Transcriptase Inhibitors) look like the building blocks of DNA called nucleotides. If the reverse transcriptase incorporates one of these ‘fake’ nucleotides, it is not able to further extend the DNA strand, leaving it incomplete, thus interfering with replication. Not all mutations cause the same level of resistance. These levels are:

| **Level** | **Total score** |
| --- | --- |
| Susceptible | 0 to 9 |
| Potential low-level resistance | 10 to 14 |
| Low-level resistance | 15 to 29 |
| Intermediate resistance | 30 to 59 |
| High-level resistance | >= 60 |

[Source](https://hivdb.stanford.edu/page/release-notes/)

We can plot resistance scores for five commonly used NRTIs.

suppressPackageStartupMessages({

library(dplyr)

library(readr)

library(stringr)

library(tidyr)

library(ggplot2)

library(knitr)

library(broom)

})

nrti\_dr\_scores <- read\_tsv("ScoresNRTI\_1555579653110.tsv", col\_types = "cdcddddddd")

nrti\_dr\_scores %>%

select(Rule, ABC:AZT, FTC:TDF) %>%

gather(arv, score, 2:6) %>%

filter(!grepl(" ", Rule)) %>%

mutate(effect = ifelse(score > 0, "resistance", "hyper-susceptible")) %>%

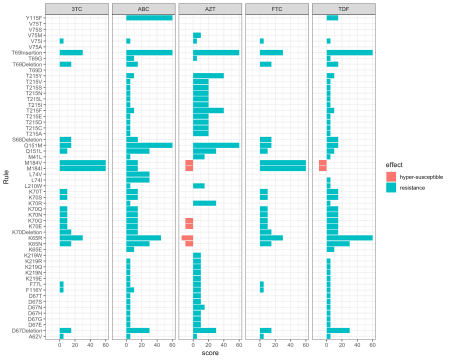
ggplot(aes(x = Rule, y = score, fill = effect)) +

geom\_col() +

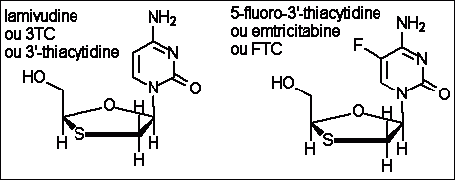
coord\_flip() +

theme\_bw() +

facet\_grid(. ~ arv)



We can see that 3TC and FTC have the exact same profiles, and they are chemically also very similar, as shown in the figure below.



(#fig:3TC and FTC)The chemical structures of 3TC (left) and FTC (right). Available at <http://aras.ab.ca/articles/HAART-Nukes-AIDS-Umber>

Also, note that some of the mutations increase susceptibility for AZT and TDF, indicated by a negative value for resistance. This is called **hyper-susceptibility**, and is used by clinicians treating patients.

For example, the mutation **M184V** means that the wild type AA at position 184 is a methionine (M) and it has been mutated to valine (V). Although this mutation makes the virus highly resistant to 3TC, it has a crippling effect on viral replication, i.e., the virus can still replicate in the presence of 3TC, but slower. This mutation also makes the virus hypersusceptible to AZT and TDF. The way clinicians use this knowledge is to keep patients on 3TC in order to keep the selective pressure for M184V, and use AZT or TDF as the other NRTI. It is typical to have a patient on two NRTIs, which is sometimes referred to as the “back bone”, and then one drug from another drug class to which the patient is fully susceptible. Knowing the genotype of the virus allows us to infer the phenotype, which in this case is the drug-resistance profile.

**PhyloPi: An affordable, purpose built phylogenetic pipeline for the HIV drug resistance testing facility**

The goal of HIV drug resistance genotyping is to determine which drugs will produce the best response in the patient, and, as mentioned earlier, we use the viral sequence information for this. Due to the rapid evolution of HIV, we can use this attribute in quality assurance. **PCR** (polymerase chain reaction) is very sensitive to contamination, and if gross cross-contamination occurred during this process, the sequences of, say, two unrelated individuals might be very similar. Also, the viral sequences of a patient over time will be more similar than the sequences between different people.

Let’s say we genotyped a patient five years ago and we have a current genotype sequence. It should be possible to retrieve the previous sequence from a database of sequences without relying on identifiers only, or at all. Sometimes when someone remarries they may change their surname or transcription errors can be made, which makes finding previous samples tedious and error-prone. So instead of using patient information to look for previous samples to include, we can instead use the sequence data itself, and then confirm the sequences belong to the same patient, or investigate any irregularities. If we suspect mother-to-child transmission from our analysis, we confirm this with the health care worker who sent the sample.

We recently published an automated pipeline for maintaining a sequence database, automatically retrieving the most similar sequences from previous genotyped viral isolates, calculating genetic distances and phylogenetic inference. Let’s look at each of these steps.

Firstly, we cannot conduct phylogenetic analysis on all past and present sequences; this would be very computationally expensive and time-consuming, and the result will be very difficult to interpret. Rather, we want to focus on the current batch of sequences the laboratory generated, but also the most similar sequences from previous batches stored in our rolling database:

* We used a tool called BLAST (Basic Local Alignment Search Tool) for this. This tool is used to add our new submissions to the current rolling database and then also retrieve the most similar previous sequences.
* These sequences are aligned using MAFFT.
* The resulting multiple sequence alignment is automatically curated with trimAl.

Finally, the sequences are ready for phylogenetic inference.

* For this, we used FastTree. As its name implies, it is fast and capable of handling large datasets requiring minimal resources.
* The resulting tree is rendered using the ETE3 python API.
* R is used to calculate a distance matrix from the multiple sequence alignment using the ape library and plotly for visualization.

In part 3 of this series, we will talk more about the distance matrix calculation and how logistic regression was used to look at inter- and intra-patient genetic distances of HIV sequences by mining a large public database at the [Los Alamos HIV sequence database](https://www.hiv.lanl.gov/content/sequence/HIV/mainpage.html). This was important, as the insights gained here were used to colour the distance matrix so that the user’s attention is drawn to relevant samples.

This is an R for medicine blog post, but there is a lot of jargon in the paragraph above. We can clear things up a bit

**How does it work?**

Firstly, our DNA sequences are strings consisting of an alphabet: A, C, G, and T. Also, genetic distances are much like [Levenshtein](https://en.wikipedia.org/wiki/Levenshtein_distance) or [Hamming](https://en.wikipedia.org/wiki/Hamming_distance) distances, or other [edit distance](https://en.wikipedia.org/wiki/Edit_distance) algorithms.

**Raw strings**

Consider the following strings, A, B and C:

A: peter kicked the ball really far

B: i think it was yesterday when peter kicked the ball really far

C: pieter kicked the round ball really hard

We can see that there are obvious similarities between these three sentences, but it would be much easier if they where aligned.

**Aligned strings**

A: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_p eter kicked the \_\_\_\_\_ ball really far

B: i think it was yesterday when p eter kicked the \_\_\_\_\_ ball really far

C: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_pieter kicked the round ball really hard

By aligning the string it is much easier to calculate the similarities or differences.

**Curated strings**

Next, we remove the overhangs since it is possible that in reality strings A and C also had more text on the left-hand side, but it was not sampled. Depending on your situation, we could also remove the internal ‘gaps’ like the word ‘round’. For our pipeline, insertions and deletions, like the letter ‘i’ in our example and the word ‘round’ are real features we would like to include. We also have a substitution in C, where the ‘f’ in A and B was changed to an ‘h’.

A: p eter kicked the \_\_\_\_\_ ball really far

B: p eter kicked the \_\_\_\_\_ ball really far

C: pieter kicked the round ball really har

**Calculation**

A: p eter kicked the \_\_\_\_\_ ball really far

B: p eter kicked the \_\_\_\_\_ ball really far

M: 111111 111111 111 11111 1111 111111 111

We can see for A and B we have matches for all of the features. If we sum up all the ones, we get 33, so the distance between them:

\[ d = \frac{33 – 33}{33} = 0\]

B: p eter kicked the \_\_\_\_\_ ball really far

C: pieter kicked the round ball really har

M: 101111 111111 111 00000 1111 111111 011

\[ d = \frac{33 – 26}{33} = 0.212\]

After the multiple sequence alignment and curation, each sequence is compared to each in order to calculate a distance matrix. This can then be used to create a phylogenetic tree, like a kind of dendrogram that can be calculated using hierarchical clustering. The above is very simplified, but should give enough background to understand the rest of the post. The resource at [EMBL-EBI Train Online](https://www.ebi.ac.uk/training/online/course/introduction-phylogenetics/what-phylogenetics) is a good place to get started if you want to know more

**The pipeline on a Raspberry Pi**

The [Raspberry Pi](https://www.raspberrypi.org/) is a small and cheap single-board computer. It is used amongst many hobbyists for all kinds of projects, for example:

* [Militarizing Your Backyard with Python: Computer Vision and the Squirrel Hordes](https://pyvideo.org/pycon-us-2012/militarizing-your-backyard-with-python-computer.html)
* [Brewing beer with the help of R](https://hackaday.com/2013/01/20/raspberry-pi-and-r/)
* [Retro gaming machines](https://retropie.org.uk/)

One of the motivations behind developing this computer was to teach kids to [code or engage in electronics](http://blog.sparkfuneducation.com/teaching-coding-to-kids-using-raspberry-pi-3-and-scratch)

All of the above are very important, but the Raspberry Pi has made its way into **science and medicine** as well. For example, a group developed a cheap [instrument](https://pubs.rsc.org/en/content/articlehtml/2017/sc/c7sc03281a) to diagnose Ebola virus infection in the field. Researchers can attach various sensors to the Raspberry Pi and use it for data collection.

**Benchmarking**

For our application, we needed to show that the Pi can handle the problem we wanted it to solve, so we did some benchmarking.

We used [Selenium WebDriver](https://www.seleniumhq.org/) to operate the pipeline as a human would, by actually browsing for an input file and submitting it through the button. Time stamps were taken for each step, and the number of blast hits that were included in the phylogenetic inference was also recorded. For this exercise, we set the number of closest sequences to retrieve for each sample to 5, which means the submitted sample and 4 of the genetically closest samples. However, it is possible that different submitted sequences have retrieved a sequence in common; these will be included in the analysis only once. When we start analyzing this data, we will see this.

# Read csv with time data

time\_dat <- read\_csv(

"timeFile.csv",

col\_types = "ccd",

col\_names = c("Run", "Description", "Measure")

)

head(time\_dat) %>%

kable(caption = "First few lines of the benchmarking data.")

| Table 1: First few lines of the benchmarking data. | | |
| --- | --- | --- |
| **Run** | **Description** | **Measure** |
| final5best\_random\_1 | blastHits | 5.000000 |
| final5best\_random\_1 | blast | 11.219230 |
| final5best\_random\_1 | mafftTime | 13.404623 |
| final5best\_random\_1 | trimalTime | 0.111737 |
| final5best\_random\_1 | fasttreeTime | 0.986582 |
| final5best\_random\_1 | heatmapTime | 2.354820 |

The Run column shows some info regarding the benchmarking experiment. We know we asked for the five best hits to be included; the sequences were pseudo-randomly selected. We started with one sequence for submission and then incremented this by one up to 50. The above again shows how data is not always in the best format for working with. We need to extract the digits at the end of the Run variable. Previously we used the tidyr::gather() function to pivot data from wide to long. This time we will use the spread() function to make long data wide.

time\_dat <- time\_dat %>%

mutate(nSubmitted = str\_extract(Run, "\\d+$") %>% as.numeric) %>%

select(-Run ) %>%

spread(Description, Measure)

head(time\_dat) %>%

kable(caption = "First few lines of the benchmarking data after some cleaning.")

| Table 2: First few lines of the benchmarking data after some cleaning. | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **nSubmitted** | **blast** | **blastHits** | **fasttreeTime** | **heatmapTime** | **mafftTime** | **renderTime** | **trimalTime** |
| 1 | 11.21923 | 5 | 0.986582 | 2.354820 | 13.40462 | 1.686239 | 0.1117370 |
| 2 | 22.08694 | 10 | 3.129514 | 2.369152 | 30.26920 | 1.890183 | 0.2699649 |
| 3 | 33.67705 | 15 | 5.480334 | 2.400223 | 47.42213 | 2.107776 | 0.4849610 |
| 4 | 43.58782 | 21 | 4.627502 | 2.437273 | 76.47209 | 2.243336 | 0.7980120 |
| 5 | 55.43246 | 25 | 10.753521 | 2.476636 | 105.21836 | 2.494058 | 1.0820050 |
| 6 | 65.18629 | 30 | 9.688977 | 2.516058 | 128.93219 | 2.653201 | 1.4656579 |

We got rid of the useless data in the Run variable and extracted the useful information into the nSubmitted variable.

Below are the explanations for the variables.

* nSubmitted: Number of sequences submitted or uploaded to the pipeline
* blast: time in seconds for blast to find most similar previously sequenced samples
* blastHits: the number of sequences retrieved
* mafftTime: the time it took to create a multiple-sequence alignment
* trimalTime: the time it took to clean the multiple-sequence alignment
* fasttreeTime: the time it took for phylogenetic inference
* heatmapTime: the time it took to produce the heatmap
* renderTime: the time it took to render the tree

**Number of sequences submitted *vs.* most similar sequences retrieved**

time\_dat %>%

ggplot(aes(x = nSubmitted, y = blastHits)) +

geom\_smooth(method = lm, se = FALSE, colour = "black", formula = y ~ x - 1, size = 0.25) +

geom\_point() +

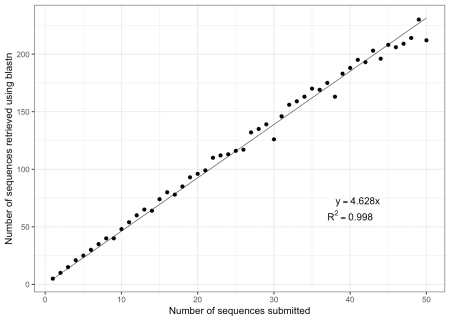
theme\_bw() +

xlab("Number of sequences submitted") +

ylab("Number of sequences retrieved using blastn") +

annotate("text", x = 41, y = 72, label = "y == 4.628 \* x", parse = TRUE) +

annotate("text", x = 40, y = 60, label = "R^2 == 0.998", parse = TRUE)



fit <- lm(blastHits ~ nSubmitted - 1, data = time\_dat)

tidy(fit) %>%

kable(caption = "Regression analysis of the number of blast hits retrieved.")

| Table 3: Regression analysis of the number of blast hits retrieved. | | | | |
| --- | --- | --- | --- | --- |
| **term** | **estimate** | **std.error** | **statistic** | **p.value** |
| nSubmitted | 4.628026 | 0.0280312 | 165.1026 | 0 |

A linear line fits the data really well. We mentioned that if different sequences retrieve the same sequence from the database, it is used only once. The slope of this line will depend on the genetic diversity of the database. A more diverse database will have a steeper slope, whereas a less diverse database will have a shallower slope. Also, theoretically, at some point, the line will reach an asymptote as the number of requested sequences start to saturate the number of available sequences. Practically, one would not have to submit more than 16 – 24 samples at a time; thus, we are in the linear part of the rarefaction curve. We can thus see from this that for the Los Alamos data used in the analysis, about 4.5 sequences get retrieved for every sequence submitted.

**BLAST time *vs.* number of sequences submitted**

time\_dat %>%

ggplot(aes(x = nSubmitted, y = blast)) +

geom\_smooth(method = lm, se = FALSE, colour = "black", formula = y ~ x, size = 0.25) +

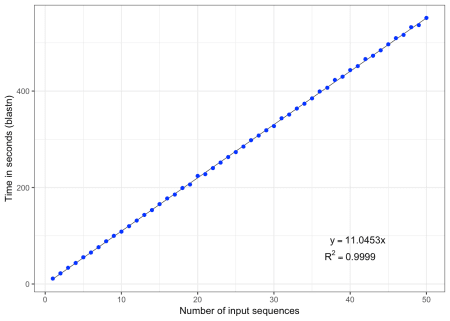
geom\_point(colour = "blue") +

theme\_bw() +

xlab("Number of input sequences") + ylab("Time in seconds (blastn)") +

annotate("text", x = 41, y = 90, label = "y == 11.0453 \* x", parse = TRUE) +

annotate("text", x = 40, y = 60, label = "R^2 == 0.9999", parse = TRUE)



fit <- lm(time\_dat$blast ~ time\_dat$nSubmitted)

tidy(fit) %>%

kable(caption = "Regression analysis of blastn time vs. number of sequences.")

| Table 4: Regression analysis of blastn time vs. number of sequences. | | | | |
| --- | --- | --- | --- | --- |
| **term** | **estimate** | **std.error** | **statistic** | **p.value** |
| (Intercept) | -0.8176139 | 0.5185500 | -1.576731 | 0.121426 |
| time\_dat$nSubmitted | 11.0453236 | 0.0176978 | 624.105409 | 0.000000 |

Again, we see a linear relationship for blastn and the time it takes to complete. For every sequence submitted, it takes about 11 seconds to search a database of about 11,000 sequence entries. We can say the blastn displays linear time complexity or \(O(n)\) time. We did not discover anything new here. Remember, the purpose of this is to show off the Pi flexing its muscles.

**Multiple sequence alignment time *vs.* number of total sequences, submitted and retrieved**

fit <- lm(mafftTime ~ I(blastHits^2) - 1, data = time\_dat)

time\_dat %>%

ggplot(aes(x = blastHits, y = mafftTime)) +

geom\_point(colour = "blue") +

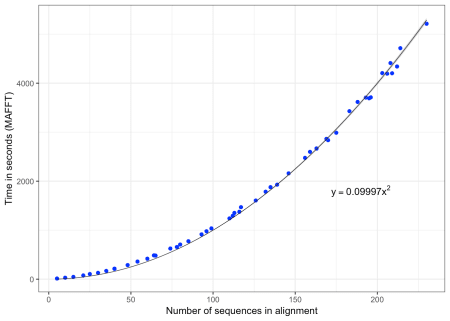
geom\_smooth(method = "lm",formula = y ~ I(x^2) - 1, colour = "black", size = 0.25) +

annotate("text", x = 190, y = 1800, label = "y == 0.09997 \* x^2", parse = TRUE) +

theme\_bw() +

xlab("Number of sequences in alignment") +

ylab("Time in seconds (MAFFT)")



tidy(fit) %>%

kable(caption = "Regression analysis of multiple sequence alignment.")

| Table 5: Regression analysis of multiple sequence alignment. | | | | |
| --- | --- | --- | --- | --- |
| **term** | **estimate** | **std.error** | **statistic** | **p.value** |
| I(blastHits^2) | 0.099974 | 0.0004048 | 246.9813 | 0 |

Since in multiple sequence alignment, each sequence is aligned with each other sequence, we would expect \(O(N^2)\) time complexity. We can see in our regression result that we are very close to what we expect. And \(O\) is a bit less than a sixth of a second. Thus, if we would analyse 16 sequences, we would retrieve \(16 \* 4.5 = 72\), and the multiple-sequence alignment would take \(0.09997 \* 72^2 = 518\) seconds or ~8.6 minutes, which is not bad. Also consider that you can submit your samples and walk away.

**Impact**

It is important to mention that PhyloPi is not used for tracking or detecting transmission clusters, but rather offers a way of automating phylogenetic analysis. Some patients will be genotyped more than once, and these sequences will cluster very closely on a phylogenetic tree. This offers a spot check into the quality of the results. Sometimes we find that the patient has two different first names, which they interchangeably use depending on the health care worker and patient language preference. We have also detected sample swaps which otherwise would have gone unnoticed.

**What next?**

In part 3, we will discuss how the inter- and intrapatient HIV genetic distances were analyzed using logistic regression to gain insights into the probability distribution of these two classes. This is also where we asked Andrie from RStudio for help. It was useful for us biologists and virologists to have someone not just to oversee the analysis we did, but also to implement the correct analysis to get the job done. Hope to see you in the next section!