Bringing bioinformatics into the classroom

# Bioinformatics

## The Power of Computers in Biology

A PRACTICAL GUIDE

Version: 6 June 2019

GOBLET

# The Power of Computers in Biology

## Overview

This Practical Guide introduces simple bioinformatics approaches for database searching and sequence analysis. A 'mystery' gene is used as an exemplar: we first characterise the gene, then use it to explore the impact of gene loss in humans. Analyses are run both online and at the command line, the latter specifically using Raspberry Pi computers running the 4273$\pi$ variant of Linux (4273pi.org).

## Teaching Goals & Learning Outcomes

This Guide introduces a popular Web-based tool for searching biological sequence databases, and shows how similar functionality can be achieved using the Linux command line. On reading the Guide and completing the exercises, you will be able to:

- search biological sequence databases using the online program BLAST, and navigate GenPept sequence records;
- execute some basic Linux commands to perform a set of simple file-manipulation tasks;
- perform BLAST searches via the Linux command line; and
- evaluate the biological implications of search results, with reference to mutations and function.

## 1 Introduction

The advent of computers and computer-driven technologies has opened new opportunities for, and has given unprecedented power to, biological investigations. Studies that once took months or years to complete may now be performed in hours or days. In the 1940s and '50s, for example, it took 10 years to determine the sequence of the first protein[1-3] – **insulin** – before automatic methods in the late 1970s made it possible to sequence proteins in more practical timescales. Technologies for rapid, high-throughput **DNA** sequencing and for *storing* DNA sequences didn't become available until the 1980s. The first DNA sequence database – the EMBL data library – was publicly released in June 1982: it held 568 sequences[4]. Today, we take it for granted that we have instant access not just to hundreds but to *millions* of biological sequences, and to software tools and **algorithms** for analysing them, allowing us to perform comprehensive searches in minutes and detailed analyses within hours.

The interdisciplinary discipline that encompasses these developments – fusing aspects of molecular biology, statistical analysis and computer science to gather, store, analyse and visualise biological data on a vast scale – is *bioinformatics*[5]. Continued advances in computer technology mean that bioinformatics approaches are now used routinely to underpin evolutionary studies, genetic and genomic research, personalised medicine, environmental and forensic analyses, and much more, in ways that were never possible before.

The ability to sequence DNA and to store nucleotide sequences means that we can rapidly compare human gene sequences with each other and with other animals, and identify differences – **mutations** – between them: ~1 nucleotide in 1,000 differs from one person to another, and from one genome to another; between any two individuals, that amounts to something like 3.3 million mutations! Most mutations have no effect (are neutral) but some may be associated with a particular trait, pre-disposition or genetic disease.

Common types of mutation include i) substitution, where the nucleotide base is different between two compared sequences – in **coding regions**, such differences may or may not lead to changes at the protein level; and ii) insertions or deletions, where a nucleotide base, or bases, are absent in one sequence relative to the other – changes like this will have significant impacts if they cause **frameshifts** that disrupt the **reading frame** of a protein-coding sequence.

This Guide explores how a popular bioinformatics tool can be used to search sequence databases and analyse particular genes. Our starting point is a 'mystery' DNA sequence. We first characterise the gene it encodes, examine how the gene has changed in different species, and use this approach to investigate gene loss in humans.

## 2 About this Guide

This Guide outlines bioinformatics approaches for searching public genetic sequence databases, for exploring the occurrence of genes in a variety of species, and for analysing genes in the human genome. Exercises are provided to help use a well-known search tool both via its Web interface and via the **command line**. Throughout the text, key terms – rendered in **bold** type – are defined in boxes. Additional information is provided in supplementary boxes.

### KEY TERMS

**Algorithm:** a set of steps or 'recipe' for solving a particular problem
**Coding region**: the portion of a DNA sequence that is transcribed & ultimately translated into a protein product
**Command line**: the means of interacting with a computer system via text commands typed directly at a keyboard
**DNA**: deoxyribonucleic acid, a molecule comprising two chains that coil together, forming a double-helix that carries genetic information
**Frameshift**: in a coding region, a shift in the reading frame caused by insertion or deletion of one or two nucleotides
**Insulin:** a hormone that regulates carbohydrate & fat metabolism
**Mutation**: a heritable change of genetic material (*e.g.*, a base change in a gene, or addition/loss of a chromosome or part of a chromosome)
**Reading frame**: a way of dividing a nucleotide sequence into consecutive, non-overlapping nucleotide triplets; triplets in coding regions are termed codons

# 3 Identifying a 'mystery' nucleotide sequence

## 3.1 The database search tool, BLAST

Our investigation begins by trying to identify a DNA sequence of unknown function. Our starting point is a 'raw' sequence that bears no **annotation**, referred to simply by the label *Sequence R*. We will use this sequence to search public DNA databases. The format of Sequence R is called FastA format (see **Figure 1**): this is a concise way of storing biological sequences for efficient database searching.

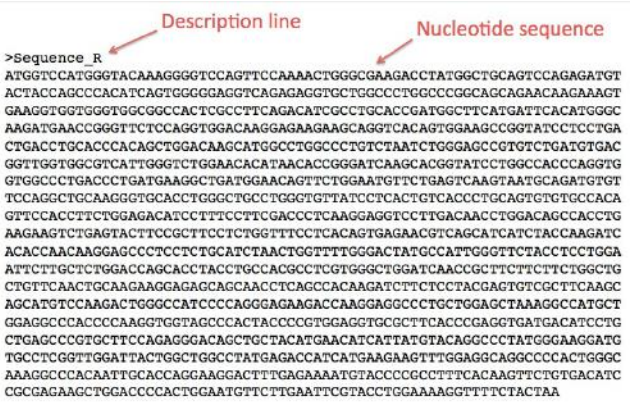



**Figure 1** ***Sequence R in FastA format***. *The file begins with a short descriptor or label that identifies the sequence, preceded by the '>' symbol, & is followed by the sequence itself in single-letter notation.*

### Nucleotide base & amino acid residue notation

**Figure 1** shows the nucleotide bases of DNA Sequence R depicted as single letters: specifically, A, C, T & G. Protein sequences can also be depicted in single-letter notation: in this case, the letters represent each of the 20 amino acids. For reference, the single-letter codes for DNA & proteins, respectively, are shown in the tables below.

| Code | Nucleotide base | Code | Nucleotide base |
|---|---|---|---|
| A | Adenine | G | Guanine |
| C | Cytosine | T | Thymine |

| Code | Amino acid | Code | Amino acid |
|---|---|---|---|
| G | Glycine | P | Proline |
| A | Alanine | V | Valine |
| L | Leucine | I | Isoleucine |
| M | Methionine | C | Cysteine |
| F | Phenylalanine | Y | Tyrosine |
| W | Tryptophan | H | Histidine |
| K | Lysine | R | Arginine |
| Q | Glutamine | N | Asparagine |
| E | Glutamic acid | D | Aspartic Acid |
| S | Serine | T | Threonine |

### EXERCISES

1 Open a Web browser.
2 Go to the 4273π website: 4273pi.org. Click on the '*NEWS*' tab (4273pi.org/schools), where you will find a link to *Sequence R*. Copy the full sequence, including the descriptor line ('*>Sequence_R*').
3 Keep the browser window open during the next tasks, so that you can copy *Sequence R* whenever you need to.

To try to identify the biological role of this 'mystery' sequence, we will search a database for sequences that are highly similar to a translation of Sequence R – *i.e.,* to its protein product. To do this, we'll use a popular online search tool – called BLAST[6] – at the **NCBI**. The NCBI maintains vast databases that contain most known DNA and protein sequences, and makes them and their search tools (some of which are shown in **Figure 2**) freely available for public use.

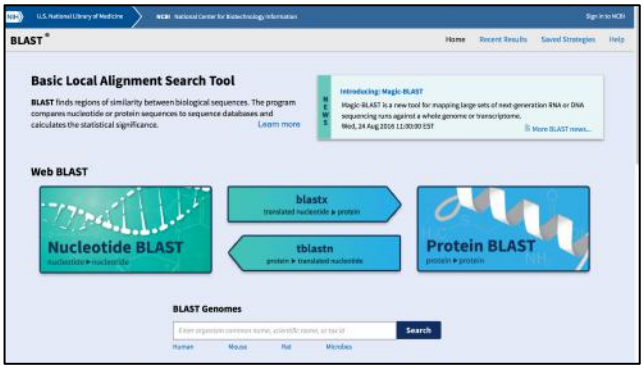
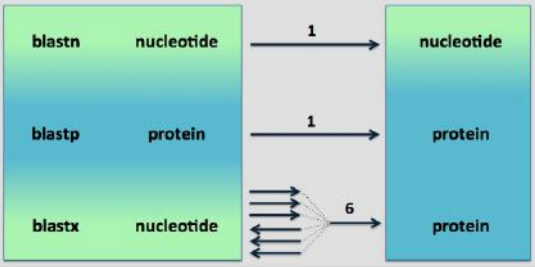



**Figure 2** ***The NCBI BLAST homepage***. *There are several BLAST programs, each adapted for a different type of search, depending on whether the starting point is a nucleotide or a protein sequence.*

### BLAST programs

Below are some popular BLAST programs & their corresponding query sequence type (see left-hand panel); the target database type is shown on the right; the centre shows the number of searches made by each program: *e.g.*, blastn searches nucleotide sequence databases with nucleotide sequence queries; blastx searches protein sequence databases with nucleotide sequence queries – these must first be translated into all 6 reading frames, so blastx performs 6 searches.




There are many other BLAST[7] tools; some have been customised for Web-based use, others are suitable for running at the command line.

### EXERCISES

1 Open a new tab in your Web browser.
2 Go to the NCBI homepage: www.ncbi.nlm.nih.gov.
3 From '*Popular Resources*' (right-hand menu), click '*BLAST*' to access the BLAST homepage (Figure 2). Click on '*blastx*'.
4 In the Web browser, in the box labelled '*Enter Query Sequence*', paste in *Sequence R*.
5 Click on the '*BLAST*' button to submit the search.

### KEY TERMS

**Annotation**: notes included within database entries to make them both informative & re-usable
**NCBI**: National Center for Biotechnology Information, part of the National Library of Medicine of the National Institutes of Health, USA

The blastx program, running on NCBI's servers, uses the **genetic code** to translate Sequence R in each possible reading frame. An illustration of the code is shown in Figure 3.

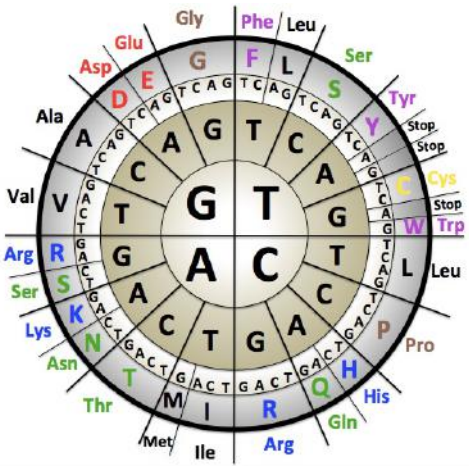
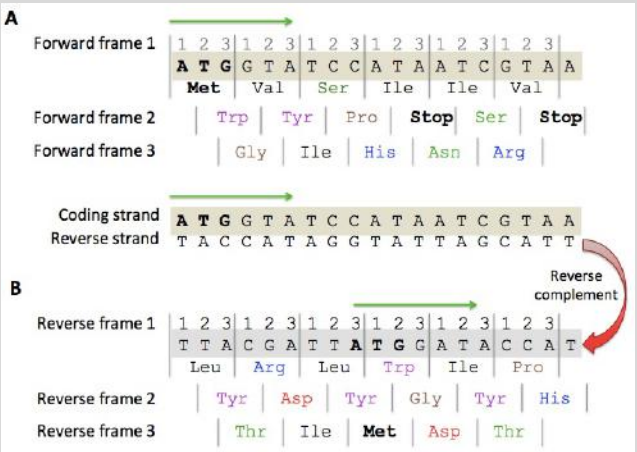



**Figure 3** ***The genetic code.*** *The wheel is divided into quadrants, one for each nucleotide base, which are shown at the centre – this represents the first base position of a codon; the second & third concentric circles denote the second & third base positions; the respective translations sit on the circumference (e.g., reading from inside-out, codon CAT points to the amino acid histidine in the bottom-right quadrant).*

## Translating DNA sequences

Translation of DNA proceeds in three forward frames (shown in **A** below) & three reverse frames (shown in **B** below). To generate the reverse frames, we effectively reverse the reverse strand & translate it in a forward direction (this new strand is termed the 'reverse complement'). The red arrow denotes formation of the reverse complement; the green arrows denote the direction of translation from the start codon in the first forward frame & from the start codon in the third reverse frame.




Note how different the six amino acid translations are from one another; & remember, start codons can occur in both forward and reverse frames, as shown above.

Once blastx has generated the six possible translations of Sequence R, the program compares them with every protein in the target sequence database. The search may take a few minutes to run. During this time, the *Status* of the job will be indicated on the Web page as *Searching*. When the search has completed, the results are shown in a single long page, headed *BLAST Results*. Some typical features of a BLAST results page are illustrated in Figure 4.

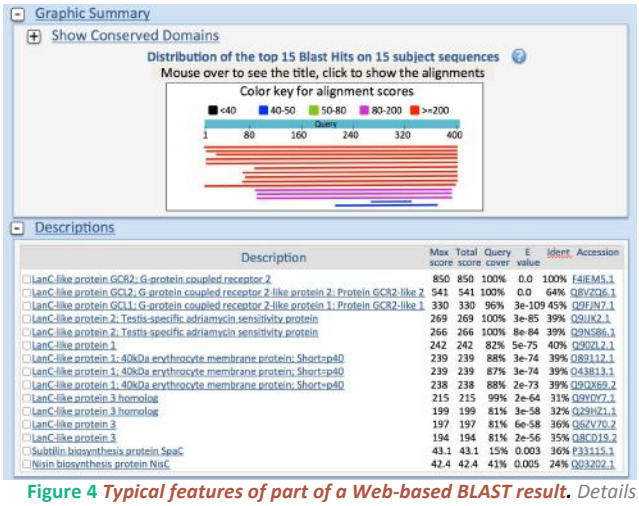



**Figure 4** ***Typical features of part of a Web-based BLAST result.*** *Details of each match (e.g., the name of the sequence, query coverage, E-value, % identity) follow a graphical overview of the matches.*

Initially, the output of a BLAST search may appear daunting. However, the result is essentially broken down into three parts: Figure 4 shows the first two. At the top, a colour-coded graphic summarises the overall quality of each matched sequence: the bars show the region of the database sequence matched, and the colour indicates the match 'strength' – red bars show strong matches, so usually appear at the top of the output.

The second section lists the top matches, giving more specific details: *e.g.*, the name or *Description* of each sequence; the extent of overlap between the query and the matched sequence (the greater the *Query cover* the better the match); the ***E value***, which indicates the reliability or significance of the match; the % identity (*Ident*); and the ***Accession number*** of the matched sequence – clicking on this number directs you to its full entry in the GenPept database (note: to obtain further, more detailed information, there are many comprehensive BLAST tutorials & guides available online[10-14]).

## GenPept

GenPept is a protein sequence database derived from translations of coding sequences in GenBank, the NCBI's principal nucleotide sequence data archive[8] (GenBank was first released in December 1982, with 602 sequences – today, it stores more than 200 million nucleotide sequences from 420,000 species[9]). GenPept is synchronised with each release of GenBank.

Entries in GenPept adhere to a strict format, with specific fields holding particular types of data: *e.g.*, the *DEFINITION* field contains the description or name of the protein; *ACCESSION* holds the entry's accession number; *SOURCE* identifies the source organism; *REFERENCE* cites relevant literature, including article *AUTHORS*, *TITLE* & *JOURNAL* of publication; *ORIGIN* holds the amino acid sequence itself.

## KEY TERMS

**Accession number**: a unique (generally invariant) computer-readable number or code that identifies a particular entry in a database

**E value**: or Expect value, the number of matches of this quality (or better) that are expected to occur simply by chance (the closer the value to 0, the better the match, & the stronger likelihood that the retrieved match is a **homologue**)

**Genetic code**: the set of rules cells use to translate information sequestered in nucleotide sequences (*i.e.*, within codons) into proteins

**Homologue**: a similar sequence that has shared evolutionary ancestry

Clicking on any of the listed names or description lines takes you to the final part of the results – the alignments between your *Query* sequence and each of the matched database sequences (denoted *Sbct*, short for Subject), as illustrated in **Figure 5**. The quality of each alignment is summarised in terms of i) the number of residue *Identities* and similarities (note that similar residues are denoted by the + symbol and are hence referred to as *Positives*), ii) the number of *Gaps* (which pinpoint sites of insertion or deletion – so-called **indels**), and iii) the *Expect* or E value. The smaller the number of insertions or deletions, the better the alignment, and hence the better the match. Long, high-quality matches have E-values closer to 0.

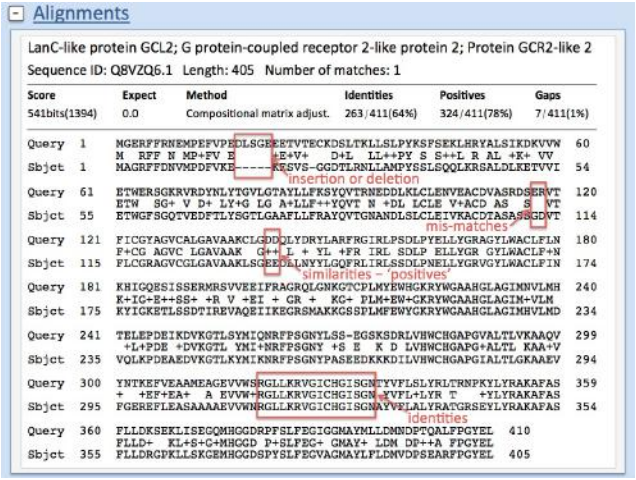



**Figure 5** ***Typical features of part of a Web-based BLAST result.*** *For each matched database sequence, an alignment with the query sequence is shown, along with details of the alignment quality.*

## EXERCISES

1 What is the E value of the best *blastx* match to *Sequence R*?

2 Scroll down to see the actual alignment of the translation of *Sequence R* (*'Query'*) against sequences from the protein database (*'Sbjct'*). For help interpreting the output, refer to Figures 3 & 4.

3 Click the link indicated by *'Sequence ID'* (beneath the protein name).

4 The translation of *Sequence R* has an extremely good match to a protein – so good, in fact, that we can assume that *Sequence R* encodes this protein. What is the protein?

5 In which organism is the protein found (look at the *'SOURCE'* line in the protein's *GenPept* record)?

6 What is the biological role of the protein? To answer this, you could begin with a general Web search for the name of the protein (which you just noted). This should give an idea of its biological role.

7 Verify the biological role of the protein by looking at the *'TITLE'* lines in the *GenPept* entry. These show the titles of various scientific studies performed on the protein. Many of these should make sense in light of your more general Web search.

### 3.2 Searching the human genome

Using the database search tool, blastx, we were able to discover the name of the protein encoded by Sequence R. Furthermore, by delving into the sequence's GenPept record, we were able to pinpoint the source organism and details of the biological role of the protein in that species. In the next part of this practical exercise, we will search the human genome to try to discover whether it harbours a functional version of the same gene.

## EXERCISES

1 Copy *Sequence R* again from 4273pi.org/schools.

2 Return to the NCBI BLAST homepage, blast.ncbi.nlm.nih.gov.

3 Beneath the *'BLAST Genomes' 'Search'* box, click on *'Human'*.

4 Paste *Sequence R* into the input box.

5 Ensure *'Database'* is set to *'Genome (GRCh38.p12 reference)'*.

6 Under *'Optimize for'*, select *'Somewhat similar sequences (blastn)'*.

7 Click on *'BLAST'* to run a *blastn* search with *Sequence R* as query.

This blastn search seeks similarity to Sequence R anywhere in the human genomic DNA. The results reveal many places where there are good matches; but Sequence R is from a non-human species, so we don't expect a perfect match. **Figure 6** shows the kinds of difference that may occur between the query and database sequences. Specifically, there may be evidence of substitutions, or there may be evidence of insertions or deletions (as denoted by the – symbol).

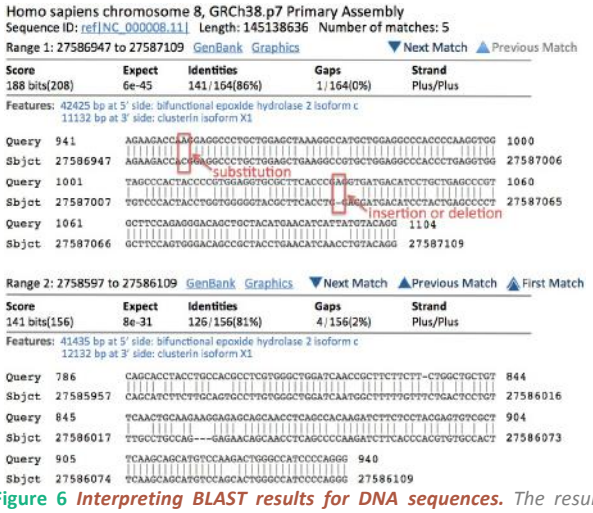



**Figure 6** ***Interpreting BLAST results for DNA sequences.*** *The result shows evidence of typical substitution & insertion or deletion mutations.*

## EXERCISES

1 From the *blastn* search of the human genome, on which chromosome have the best matches been found?

2 Examine the top results of the *BLAST* output very carefully. Scroll down, below the *Graphic summary* & *Descriptions*, to see fragments of *Sequence R* (*'Query'*) aligned with fragments of human genomic DNA found by the search (*'Sbjct'*).

3 Within these alignments, can you see evidence of a substitution mutation? If so, sketch the region of the alignment that includes it.

4 Is there evidence of an insertion or deletion mutation? If so, sketch the region of the alignment that includes the insertion or deletion.

Insertions or deletions of one or two bases in protein-coding DNA indicate frameshift mutations – see **Figure 7**. If a match between Sequence R and the human genomic sequence includes such a mutation, this offers strong evidence that this part of the genome no longer encodes a functional product. Such former, no-longer-functional genes are referred to *pseudogenes.*

## KEY TERMS

**Indel:** an insertion or deletion ambiguity where, given two sequences of different length (say, from different species), we can't infer whether one species lost part of a sequence or the other gained a part

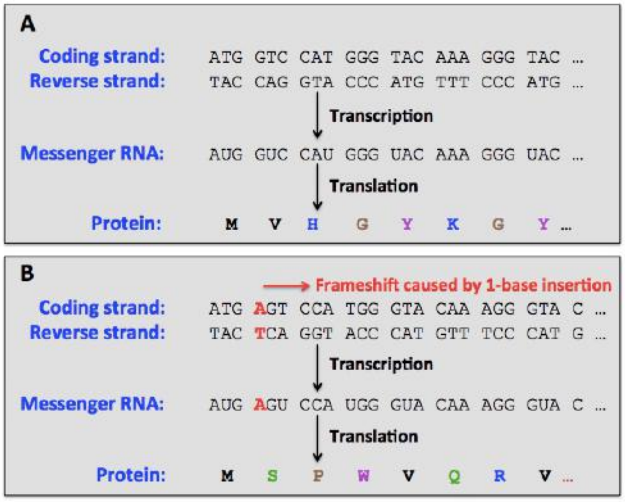



**Figure 7** ***A frameshift mutation arising from one single-base insertion.***
*A) Coding DNA & its encoded protein sequence; B) Single-base insertion leading to a completely different protein sequence.*

## EXERCISES

1 Do you think the human genome includes a functional version of *Sequence R*, or is there a pseudogene instead? Why?

2 *Sequence R* is from the mouse. It's the coding sequence of *Gulo*, which encodes L-gulonolactone oxidase, an enzyme that synthesises vitamin C. What does the answer above tell us about the diet of humans & how this differs from the diet of mice?

### 3.4 Using BLAST at the command line

We typically interact with computers via Graphical User Interfaces (GUIs), using a mouse or touch-screen, with perhaps a little typing. So far, we've been using a GUI to run BLAST on the NCBI's Web servers. GUIs are excellent for allowing exploratory analyses and for data visualisation, and often provide access to the most up-to-date data. Above all, they're generally designed to be easy to use; however their simplicity can also be their downfall, often making them rather inflexible and quite limiting.

An alternative is to use the command line. At first, the command line may seem daunting. However, it's a more flexible way of interacting with a computer, and provides much greater control. For example, the command line allows us to specify precise data and software versions (which will remain the same until *we* change them); better still, it lets us view the commands we just ran, making it easy to record all the steps we took – this is useful for subsequently being able to repeat the analyses we've performed. By contrast, it can be difficult to recall exactly what we clicked on in a GUI; worse, GUIs let us run analyses as simple **black boxes**, without us knowing, for example, which settings or **parameters,** which version of a database or what version of a program we actually used.

Many bioinformatics tasks can be performed via GUIs or via the command line. Windows and macOS do have command lines, but we find the Linux **Operating System (OS)** command line to be the most convenient. Linux is very widely used in scientific computing, and is available free for most computers, including the **Raspberry Pi**. The Raspberry Pi computers used in this task run the 4273$\pi$ variant of Linux: **4273pi.org**.

#### *Introduction to the command line*

The next exercises will be performed at the Raspberry Pi command line. We will use a text editor to create and save a file, then use some basic commands to locate, read and remove files. You may be familiar with files and folders; in the Linux OS, folders are usually referred to as *directories* (often shortened to *dir*).

## EXERCISES

1 Start a terminal window by clicking the black rectangular icon (in the menu bar at the top of the computer screen).

2 At the prompt, type

```
nano test.txt
```

3 Press '*ENTER*' to open a new file, `test.txt`, in the `nano` editor. You can now add text, navigate with the arrow keys, delete text, *etc.*

4 Type any text (this can be your name, your pet's name, *etc.*): *e.g.,*

```
hello world
```

5 To save your file & exit `nano`, press '*ctrl-x*'.

6 `nano` will then ask

```
Save modified buffer (ANSWERING "NO" WILL DESTROY CHANGES)?
```

7 Press '`y`' to save your file.

8 `nano` will ask for a file name, suggesting the name you gave earlier (`test.txt`).

9 Press '*ENTER*' to close `nano` & return to the terminal window.

Having created your first file, it's helpful to check whether everything worked correctly. To do this, we must find the file and display its contents. If satisfied that the task worked, we can delete the file.

## EXERCISES

1 To locate the file you just created, we need to list all files in the current directory. At the prompt, type

```
ls
```

2 Press '*ENTER*' (note, `ls` is short for *list* – it therefore starts with a lower-case *L*, not the numeral *one*).

3 This shows all files in the directory, including the one you just created, `test.txt`.

4 To display the contents of the file, type

```
cat test.txt
```

5 Press '*ENTER*'. Examine the displayed file contents. If this is what you expected to see, you can now delete the file.

6 To permanently delete the file, type

```
rm test.txt
```

7 Press '*ENTER*'. To check that the file has been removed, type

```
ls
```

8 Press '*ENTER*'.

9 If it was removed successfully, the file name `test.txt` should no longer be displayed in the directory contents.

## KEY TERMS

**Black box:** a system, device or object that can be understood & utilised in terms of its inputs & outputs, but without any knowledge of its internal workings – algorithms are often implemented as black boxes

**Operating system:** the software that controls a computer's hardware & software resources, & provides its program processes

**Parameter:** a value that's passed to a command or a program that informs the command or program how to behave: *e.g.*, BLAST might require the name of the database to be searched, or the name of a file in which to write its results

**Raspberry Pi**: a small single-board computer designed to help teach basic computer science in schools

**Some useful Linux commands**

| | |
|---|---|
| *ctrl-c* | Usually aborts a program running at the command line (useful for terminating a program launched by accident). |
| *ctrl-d* | Indicates end-of-file; if typed at the command prompt, it closes the terminal window, but doesn't shut down the computer. |
| `cat` *filename* | Displays contents of the file named *filename*. |
| `cd` *dirname* | Moves into directory named *dirname* (commands then view this as the default location for files). |
| `cd ~` | Moves to your home directory, the starting directory when you launch the terminal. |
| `cp` *file1 file2* | Copies file *file1* to *file2*. If *file2* already exists, its original contents will be over-written. If *file2* is a directory name, a copy of *file1* is placed in that directory. |
| `head -20` *filename* | Displays the first 20 lines (or other desired number) of the file named *filename*. |
| `ls` | Lists contents of the current directory. |
| `ls -Fl` | Lists contents of current directory, showing which are directories (name followed by a /), which are executable (name followed by a *), & giving information on files, including file size. |
| `ls -Fl` *dirname* | As above, but lists only contents of the directory named *dirname*. |
| `man` *command* | Displays the Linux manual for the named *command*, if there is one. |
| `mkdir` *dirname* | Makes a directory named *dirname*. |
| `mv` *file1 file2* | Moves file *file1* to *file2*. `mv` can move or rename files. If *file2* is an existing file name, the original file contents will be destroyed. If *file2* is the name of an existing directory, *file1* will be moved into that directory. |
| `nano` *filename* | Opens a file called *filename* in the **`nano`** editor. *ctrl-x* exits the editor, & will prompt you to save any changes you've made before exiting. |
| `nano -v` *filename* | Opens file named *filename* in the `nano` editor in **read-only** mode. Use this to look at important files you wish to preserve unchanged. |
| `rm` *filename* | Deletes the file named *filename*. Warning: there is no 'trash can' – this command is irreversible without special software. |
| `sudo` *command* | Runs the named *command* as an administrator ('super-user'). |
| `sudo shutdown -h now` | Shuts down the computer so you can safely switch it off. |

A broader guide to Linux commands is available online[15].

### *Sequence data in text files*

Having created a file and used various basic file-manipulation commands, we are now ready to start working with some real data.

**EXERCISES**

1 Move to the directory containing the sequence data for the next practical tasks by typing

```
cd 4273pi/GULO
```

2 Press '*ENTER*'. All files will now read from (& write to) this directory.

3 List the contents of the directory by typing

```
ls
```

4 Press '*ENTER*'. The file `sequence_r.fa` contains the same sequence you used previously. To display the file contents, type

```
cat sequence_r.fa
```

5 Press '*ENTER*'. Note: it's time-consuming & error-prone to type file names. If you start typing & then press the '*TAB*' key, the command line will 'auto-complete' the filename, where possible.

6 To save time, type

```
cat s
```

7 Press the '*TAB*' key. This should reveal the full file name, `sequence_r.fa`

8 Press '*ENTER*' to run the command.

The file `Homo_sapiens.GRCh38.dna.chromosome.8.fa` contains the DNA sequence of human **chromosome** 8 (downloaded in Fasta format from **Ensembl**, **www.ensembl.org**). This will be the target sequence for our next set of BLAST searches.

**EXERCISES**

1 To view the contents of the file, we can use the `cat` command. Note: the command line is case-sensitive, so *H* & *h* aren't the same. Also, `Homo_sapiens.GRCh38.dna.chromosome.8.fa` is a long name, so typing it out by hand is tedious & error-prone.

2 To save time & avoid typing errors, type

```
cat H
```

3 Press the '*TAB*' key. This should reveal the full file name.

4 Press '*ENTER*' to run the command. The sequence is very long: as you can see, the computer is taking a long time to display the file.

5 To stop the process, press '*ctrl-c*'.

### *Running BLAST at the command line*

Having located our data files, we can now use Sequence R as a query in a BLAST search, with chromosome 8 as the database.

**EXERCISES**

1 To launch the *BLAST* search, type (on one line):

```
blastn -task blastn -query sequence_r.fa -subject
Homo_sapiens.GRCh38.dna.chromosome.8.fa > results.txt
```

`blastn` tells the computer to run a search with *blastn*, which searches a DNA sequence database with a DNA sequence query

`-task blastn` is equivalent to the '*Optimize for somewhat similar sequences (blastn)*' option we used earlier with BLAST online

`-query sequence_r.fa` tells the computer to use the sequence in `sequence_r.fa` as the query for the search

`-subject Homo_sapiens.GRCh38.dna.chromosome.8.fa` directs the computer specifically to search the `Homo_sapiens.GRCh38.dna.chromosome.8.fa` file

`> results.txt` instructs the computer to send the search results to a text file

2 Press '*ENTER*' to run the command. This may take a few moments.

**KEY TERMS**

**Chromosome**: the organisational unit of a cell's hereditary material, generally comprising tight DNA-protein complexes

**Ensembl**: a central resource that stores the genomic data of humans & other species, & provides visualisation tools for online data browsing

**Nano:** a **non-GUI** text editor that runs within a computer terminal

**Non-GUI:** non-Graphical User Interface, an interface that allows interaction with a computer via text commands, rather than via icons, *etc.*

**Read-only**: a mode of accessing a file that allows it to be viewed but does not allow changes to be made to it

## EXERCISES

1 To examine the results in *nano*, type

```
nano results.txt
```

2 Press '*ENTER*'. Increase the size of the terminal window, by clicking & dragging an edge of the window with the mouse (see **Figure 8**).

3 To explore the results file in *nano*, scroll up & down one line at a time with the arrow keys, or a page at a time with '*ctrl-y*' & '*ctrl-v*' .

4 From these results, is there evidence of a functional gene in the human genome, or is there a pseudogene instead? Is this the same conclusion you drew from the search you ran via NCBI's website?

5 Exit *nano* by pressing '*ctrl-x*'

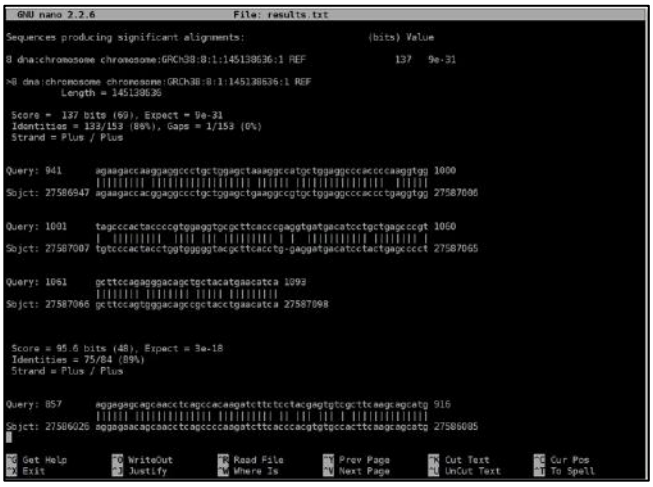



**Figure 8** ***Screen-shot from the nano editor.*** *A menu at the bottom of the screen provides a range of navigation & file-manipulation options: these include commands for getting help (ctrl-g), for moving through the file page-by-page (ctrl-y, ctrl-v), finding a given piece of text within the file (ctrl-w), exiting the file (ctrl-x), & so on.*

To document your work, you can display a log of recent commands using the `history` command. The results can be pasted into a word processor, printed out, copied into a lab book, *etc*. This can help you track down mistakes, or share your work with others.

## EXERCISES

1 In the terminal window, type

```
history
```

2 Press '*ENTER*'.

3 Use the '*up arrow*' key to scroll up through your command history; use the '*down arrow*' key to scroll back down the list.

4 To recall & execute the previous command, type

```
!!
```

(note: any previous command may be recalled & executed – *e.g.*, !5 recalls & executes command 5 from the history)

5 To save your command history to a file, type

```
history > mycommands.txt
```

6 To view the contents of the file, type

```
cat mycommands.txt
```

### *Investigating functional genes in other species*

In our study, we've discovered that mice have a functional *Gulo* gene, and can therefore synthesise their own vitamin C. In humans, the *Gulo* gene isn't functional; consequently, human cells can't synthesise the vitamin, so must obtain it from dietary sources. But why might human ancestors have lost the ability to synthesise vitamin C?

We can shed light on this question by considering which animals can make vitamin C, and which have a pseudogene and hence need to obtain it from their diets. Let's use the skills you've learnt during these practical tasks to investigate which animals have a functional *Gulo* gene: consider a range of animals (such as illustrated in **Figure 9**), and think about their diets.

**Figure 9** ***Animals in their habitats.*** *Which can synthesise vitamin C?*

## EXERCISES

1 Return to the NCBI BLAST homepage, blast.ncbi.nlm.nih.gov.

2 In the '*BLAST Genomes*' '*Search* box', type the name of your chosen animal. A drop-down list should appear – this may be slow to load, so be patient (typing a space after the name sometimes helps).

3 Select the name of your animal of interest (*e.g.*, '*Norway rat*') from the drop-down list, then click the '*Search*' button. This pre-selects the genome of your animal as the target database for *BLAST*.

4 Paste *Sequence R* into the input box. Ensure that the search tab is set to '*blastn*'.

5 Under '*Optimize for*', select '*Somewhat similar sequences (blastn)*'.

6 Click on '*BLAST*' to search the genome with *Sequence R*.

7 Examine the results carefully to determine whether or not the genome harbours a functional *Gulo* gene. Make a note of your answer.

8 Try the search again with a range of animals from different habitats.

9 Why might we (humans) or our ancestors have lost the ability to synthesise vitamin C?

## TAKE HOMES

Having completed these exercises, you can now:

1 Search biological sequence databases using the default options of some of the NCBI's *BLAST* programs;

2 Determine the significance of matches retrieved by *BLAST* searches;

3 Verify biological details of retrieved *BLAST* matches with reference to their records in the *GenPept* database;

4 Customise NCBI *BLAST*'s input parameters to search the latest available version of the human genome;

5 Identify mutations in *BLAST* results;

6 Use some basic Linux commands to perform a variety of simple file-manipulation tasks;

7 Perform *BLAST* searches via the Linux command line;

8 Customise NCBI *BLAST*'s input parameters to search the genomes of animals of your choice;

9 Evaluate the biological implications of search results, with reference to specific mutations & their likely functional impacts.

## 4 References & further reading


1 Sanger F. (1945) **The free amino groups of insulin.** *Biochem. J.*, **39**, 507-515.
2 Sanger F *et al.* (1955) **The amide groups of insulin.** *Biochem. J.*, **59**(3), 509–518.
3 Ryle AP *et al.* (1955) **The disulphide bonds of insulin.** *Biochem. J.*, **60**(4), 541–556.
4 Hamm GH & Cameron GN. (1986) **The EMBL data library.** *Nucleic Acids Res.*, **14**(1), 5-9.
5 Torrance J *et al.* (2012) **Higher Biology.** Hodder Gibson.
6 Altschul SF *et al.* (1990) **Basic local alignment search tool.** *J. Mol. Biol.*, **215**(3), 403-410.
7 Altschul SF *et al.* (1997) **Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.** *Nucleic Acids Res.*, **25**(17), 3389-3402.
8 Burks C *et al.* (1985) **The GenBank nucleic acid sequence database.** *Comput. Appl. Biosci.*, **1**(4), 225-233.
9 Sayers EW *et al.* (2018) **GenBank.** *Nucleic Acids Res.*, 47(D1), D94-D99.
10 **BLAST Guide: Overview of the various NCBI BLAST services & reports: ftp.ncbi.nih.gov/pub/factsheets/HowTo_BLASTGuide.pdf**
11 **NCBI Tutorials: www.ncbi.nlm.nih.gov/home/tutorials**
12 **The New BLAST Results Page: Enhanced graphical presentation and added functionality: ftp.ncbi.nih.gov/pub/factsheets/HowTo_NewBLAST.pdf**
13 Pertsemlidis A & Fondon JW 3rd. (2002) **Having a BLAST with bioinformatics (and avoiding BLASTphemy).** *Genome Biology*, **2**, reviews 2002.1–2002.10.
14 Attwood TK & the GOBLET Foundation. (2018) **A Critical Guide to BLAST.** [version 1; not peer reviewed]. *F1000Research*, **7**, 1435 (document) (**doi.org/10.7490/f1000research.1116052.1**).
15 Attwood TK & the GOBLET Foundation. (2018) **A Critical Guide to Unix.** [version 1; not peer reviewed]. *F1000Research*, **7**, 1436 (document) (**doi.org/10.7490/f1000research.1116050.1**).


## 5 Acknowledgements & funding


GOBLET Practical Guides build on GOBLET's Critical Guide concept, using layout ideas from the Higher Apprenticeship specification for college-level students in England (**www.contentextra.com/lifesciences/unit12/unit12home.aspx**). The contents expand on materials made freely available by the 4273π project (**4273pi.org**).

This Guide was developed with the support of a donation from EMBnet to the GOBLET Foundation.

Design concepts and the Guide's front-cover image were contributed by CREACTIVE.


## 6 Licensing & availability



The Guide is freely available for download via the GOBLET portal (**www.mygoblet.org**), EMBnet website (**www.embnet.org**) and the F1000Research Bioinformatics Education and Training Collection (**f1000research.com/collections/bioinformaticsedu?selectedDomain=documents**).

Lesson plans and handouts are freely available from **4273pi.org/teacher-resources**.

## 7 Disclaimer

## About the organisations

### GOBLET

GOBLET (Global Organisation for Bioinformatics Learning, Education & Training; **www.mygoblet.org**) was established in 2012 to unite, inspire and equip bioinformatics trainers worldwide; its mission, to cultivate the global bioinformatics trainer community, set standards and provide high-quality resources to support learning, education and training.

GOBLET's ethos embraces:

- ***inclusivity**: welcoming all relevant organisations & people*
- ***sharing**: expertise, best practices, materials, resources*
- ***openness**: using Creative Commons Licences*
- ***innovation**: welcoming imaginative ideas & approaches*
- ***tolerance**: transcending national, political, cultural, social & disciplinary boundaries*

Further information can be found in the following references:


- Attwood *et al.* (2015) **GOBLET: the Global Organisation for Bioinformatics Learning, Education & Training.** *PLoS Comput. Biol.*, **11**(5), e1004281.
- Corpas *et al.* (2014) **The GOBLET training portal: a global repository of bioinformatics training materials, courses & trainers.** *Bioinformatics*, **31**(1), 140-142.


GOBLET is a not-for-profit foundation, legally registered in the Netherlands: CMBI Radboud University, Nijmegen Medical Centre, Geert Grooteplein 26-28, 6581 GB Nijmegen. For general enquiries, contact **info@mygoblet.org**.

### EMBnet

EMBnet, the Global Bioinformatics Network, is a non-profit organisation, founded in 1988 to establish and maintain bioinformatics services in Europe. Eventually expanding beyond European borders, EMBnet created an international network to support and deliver bioinformatics services across the life sciences: **www.embnet.org**.

Since its foundation, EMBnet has had a keen interest in Education and Training (E&T), and has delivered tutorials and courses worldwide. Perceiving a need to unite and galvanise international E&T activities, EMBnet was a principal founder of GOBLET. For more information and general enquiries, contact **info@embnet.org**.

### The 4273π Project

4273π provides a freely available, customised distribution of Raspbian GNU/Linux for the Raspberry Pi computer. 4273π is for those wishing to teach, learn or use bioinformatics on the Raspberry Pi. The project focuses on bioinformatics education, computational science, open educational resources, public engagement and democratisation of science: **4273pi.org**.

### CREACTIVE

CREACTIVE, by Antonio Santovito, specialises in communication and Web marketing, helping its customers to create and manage their online presence: **www.gocreactive.com**.

## About the authors

### Daniel Barker[§]

Daniel Barker is a Reader in Bioinformatics and, from September 2019, Director of the MSc Bioinformatics programme at the University of Edinburgh. He is particularly interested in bioinformatics education, computational phylogeny and philosophy of science. With colleagues (including Heleen Plaisier), he released the 4273π SD card for Raspberry Pi as an Open Educational Resource in 2013. As part of the 4273π project, he has been helping bring a version of the workshop presented in this GOBLET resource to schools and teachers around Scotland since 2016.

### Heleen Plaisier[‡]

Heleen Plaisier is Assistant Herbarium Curator (Cryptogams) at the Royal Botanic Garden Edinburgh. Prior to this, from 2016-2018, she was a Postdoctoral Research Associate in Bioinformatics Education at the University of Edinburgh, responsible for the running, delivery and development of the 4273π bioinformatics education project. During this time, the project became one of the largest bioinformatics-at-school projects in the world.

### Stevie Anne Bain[§]

Stevie A Bain is a Postdoctoral Research Associate in Bioinformatics Education at the University of Edinburgh. She is currently responsible for the running, delivery and development of the 4273π bioinformatics education project. Stevie is an evolutionary biologist with a research background in genomics and epigenetics.

### Affiliation


[§]University of Edinburgh, [‡]Royal Botanic Garden, Edinburgh (UK).


### Teresa K Attwood (orcid.org/0000-0003-2409-4235)

Teresa (Terri) Attwood is a Professor of Bioinformatics with more than 25 years' experience teaching introductory bioinformatics in undergraduate and postgraduate degree programmes, and in *ad hoc* courses, workshops and summer schools, in the UK and abroad.

Focusing on protein sequence analysis (particularly G protein-coupled receptors), she created the PRINTS database, co-founded InterPro, and co-developed tools for sequence analysis, and for linking research data and scientific articles (Utopia Documents).

She wrote the first introductory bioinformatics text-book; her third book was published in 2016:


- Attwood TK & Parry-Smith DJ. (1999) ***Introduction to Bioinformatics.*** Prentice Hall.
- Higgs P & Attwood TK. (2005) ***Bioinformatics & Molecular Evolution.*** Wiley-Blackwell.
- Attwood TK, Pettifer SR & Thorne D. (2016) ***Bioinformatics challenges at the interface of biology and computer science: Mind the Gap.*** Wiley-Blackwell.


### Affiliation


School of Computer Science, The University of Manchester, Oxford Road, Manchester M13 9PL (UK).