Babraham Bioinformatics

Learning to Program With Perl

Course Exercises – v1.1

Licence

Section 1: Getting Started with Perl

Exercise 1: Scalars and Scalar Variables

1a
Write a script which prints out Hello World to the console, ending with a newline.

1b
Write a script which stores your name in a variable. Have it print out your name as part of a hello statement sent to the screen. Try breaking your name into separate first name and last name variables.

1c
Write a script which prints out the text string on the following line – try using both single and double quotes and note the differences in escaping needed:

Mike@example.com says “$500 for a car – that’d be a good deal!”

1d
Use a here document to quote and print a multi-line formatted piece of text. Make at least one variable substitution within the body of the text.

1e
Write a script to find the answer to the following equation:

$$x = \left(\sqrt{(a+b)^2}\right) / c$$

For the following values:
a=2 b=3 c=4
a=-20 b=5 c=3

Exercise 2: Scalar Functions

2a
Write a script which will print out the length of any variable as part of a suitably formed sentence (eg: The string ‘dog’ is 3 letters long). Use this to determine the length of the following words (remember there is an electronic version of this manual on your CD so you don't have to type these out!):

Perl
Sisyphean
Antidisestablishmentarianism
Pneumonoultramicroscopicsilicovolcanoconiosis

Exercise 3: Conditions

3a
Write a script which takes a string and will test to see if it is palindromic (reads the same when written backwards). The test should be case insensitive. It doesn’t have to cope with spaces

being in different places (although you can add this functionality after you've reached Section 4).

3b

Write a script which works out how old a child should be to know a certain word. The classification should be based on the length of the word, as follows:

5 years <= 3 letters
6 years <= 4 letters
8 years <= 6 letters
10 years <= 10 letters
12 years = any length

3c

A cheery one this. Write a script to calculate when you're going to die. Assume that the average life expectancy is 70 and then adjust this according to the following recorded variables:

- Are you male or female? Females get an extra 4 years.
- Are you a smoker? Add 5 years if not, subtract 5 years if you are.
- How often (per week) do you exercise? Subtract 3 years if never, add one year for each exercise session.
- How many units of alcohol do you drink per week. Remove 0.5 year for every unit over 7. Gain 2 years if teetotal.
- Do you eat fatty food? Add 3 years if not.

Calculate the life expectancy of a male non-smoker who exercises twice a week, drinks 10 units of alcohol a week and eats fatty food.

NB This is NOT a scientific calculation. I cannot be held responsible for mis-predicting your death! I did look at a proper scientific questionnaire but when it started asking about bowel movements I decided to make one up.

3d

Write a small password checking script. This will record the username, old password and new password. The rules are that a password is OK if it is >7 characters long, contains some uppercase characters and is different to the old password. The admin user (username 'admin') can do whatever they like. Print out whether the new password is OK.

Try doing this as one compound if statement.

Section 2: Arrays, Hashes and Loops

Exercise 4: Arrays (and loops)

4a

Create an array of the names of Snow White's dwarves. Check that there are the right number there. Use a loop to say "Hi ho!" to each dwarf in turn.

4b

Create an array and populate it with 100 random integers between 1 and 100 (perldoc –f rand). Sort the array numerically and write out the lowest 10 numbers on one line separated by tabs.

4c

Write a program to calculate the melt temperature of a DNA primer. Take a primer sequence and split it into its component characters (see perldoc –f split), then work out the temperature by adding 4 degrees for every G or C and 2 degrees for every A,T or U.

Exercise 5: Hashes (and loops)

5a

Make a hash of 3 letter amino acids codes (keys) to their molecular weight (values). Check that you have included tryptophan, and print out its weight. Print out a list of all of the amino acids sorted by their molecular weights (heaviest to lightest). Find the sequence for mouse lysozyme protein and work out its molecular weight.

5b

Create a hash which translates codons into 3 letter amino acid codes. Write a script which will extract, translate and print out the CDS from refseq entry NM_020661. You can copy over the raw sequence and the CDS start and end into your script.

For bonus points print out the original codons and translation aligned with each other. For more bonus points use the hash from exercise 5a to calculate the molecular weight as you go. Leave these additions until you've done the other exercises though.

Exercise 6: Loops

6a

Write a program to calculate prime numbers. This should run until it has found 10,000 prime numbers. For a number to be prime it only has to not divide exactly by all the previous prime numbers, you should therefore store and reuse all previous primes in a suitable data structure. Print out the 10,000th prime when you find it. There are lots of nice opportunities for optimisations in this exercise – let's see who can find the 10,000th prime the quickest (my example script finds it in 16 seconds on my 2GHz desktop PC, but if you do it right you should be able to find the answer in a fraction of a second).

[NB: 1 is not a prime number, but 2 is – as a guide to tell you whether your program is getting the right answer the 100th prime is 541]

Section 3: File Handling

Exercise 7: Reading and Writing Files

7a

You will be provided with a raw microarray data file called 'raw_data.txt' This file contains 6 columns of information:

- Probe Name
- Chromosome
- Position
- Feature
- Sample A data
- Sample B data

You should write a program to filter this data. The first line is a header and should be kept. For each other line calculate the $\log_2$ of Sample A data and Sample B* data and keep the line only if:

- Log_2 of either Sample A or B is greater than 2
- The $\log_2$ difference (either positive or negative) between Sample A and B is greater than 3 (ie an 8 fold change in raw value)

Print the filtered results in a file called filtered_data.txt

*[Perl does not have a function to directly calculate $\log_2$. However you can use any base log to do this calculation since $\log_2(x) = \log(x)/\log(2)$. You can therefore use the perl log() function – which calculates the natural log, to work out $\log_2$.]

7b

Write a script which will generate simulated FastQ sequence files with defined sequence composition. You should generate separate files which average GC content of 10,20,30..90% over a length of 50bp.

Initially you could generate the files with a fixed quality value for each base, but you could also try assigning a random quality in the range 0-30 to each base.

You can check the results of your script by loading the generated files into FastQC and looking at the profile which is generated.

[There is a very good wikipedia article describing the FastQ format. You'll need to use rand to generate the random bases and scores. To encode a quality score you use the perl 'chr' function to turn a numeric score into an ascii text letter]

7c

Write a script which shows a user a secret message only when they enter a valid password. The user should be prompted for the password. If the provided password is incorrect they should be informed and prompted again. The script should quit after the user enters the right password, or after they have provided 5 incorrect guesses.

[NB For this program the password will be visible when the user types it. It is possible to hide the text the user types, but this requires the use of a module (which will be covered in Section7)]

7d

You will be provided with two files. Annotation.txt contains a list of sequence accession codes and their associated descriptions, separated by tabs. Data.txt has the same list of accessions (though not in the same order) alongside some tab separated data values. You should combine these files to produce a single file containing the accession, data and description for each gene. Your script should perform basic sanity checks on the data it reads (eg checking that you have both an accession and description for each gene, and checking that each accession in the data file really does have annotation associated with it before printing it out).

Exercise 8: Filesystem operations

8a

Write a script which takes a directory name and provides a series of statistics about that directory. It should say:

- How many files it contains
- How many of the files are readable and writable
- How many subdirectories it contains
- It should also break down all the files into file types based on the last 3 letters in their name. These should be listed with the most common extensions first.

If the name provided either doesn't exist, or isn't a directory the script should throw a helpful error message.

Section 4: Regular Expressions

Exercise 9: Regular Expressions

9a

You will be provided with a file of words. Construct regular expressions which can pick out words from this file with the following characteristics.

- Words containing an exclamation mark
- Words containing 3 or more consecutive numbers
- Words containing 5 or more consecutive vowels
- Words of at least 13 characters with z as the last letter
- Words of at least 8 characters containing no vowels

9b

You will be provided with a list of gene descriptions (desciptions.txt). Some of these descriptions contain enzyme classification codes (which look like EC 1.2.3.4 where each number can be of any length and there may or may not be a space between EC and the first number). Write a script which will parse through the file pulling out all the EC numbers it can find and print out this list.

9c

You will be provided with a file called angstrom.txt. Write a script which will read this file containing measurements in Angstroms and will convert them into nm. To save you looking it up 1 Angstrom is 0.1nm. Your replacement should be able to cope with both whole and fractional Angstrom measures, and for them to be suffixed with A or Angstrom.

It should find and replace:

3A
12 A
2.75 angstroms
0.123Angstroms

It should not alter the following examples:

I like the number 3. A very nice number.
There are 27 Aardvarks in London Zoo.

[NB A standard replacement will treat the right hand part of the replacement as a double quoted string. If you want to put code in this part (as you will need to for this), you need to tell perl to evaluate the replacement by using the 'e' modifier after the replacement. You can use the . operator to concatenate strings and calculations in the replacement,

eg `s/(\d+)/$1 . "squared is" . $1**2/e`

9d

Write a program to produce a restriction map of a sequence. It should read in a sequence in fasta format and produce a list of restriction site positions in it.

Fasta format looks like this:

```
>sequence_name
GATAGTCGTAGTGCTAGTGCTAGTGCTAGTGC
GATAGTCGTAGTGCTAGTGCTAGTGCTAGTGC etc…
```

The restriction sites to search for are shown below along with their recognition sites. The gap in the site represents the cut site, and the base after this gap should be reported as the position of the site.

For bonus points have the restriction map written out to a file named after the sequence name (eg sequence_name_map.txt).

You will be provided with a fasta file to test.

Restriction sites to use:

- EcoRI `g aattc`
- BtgI `c crygg`
- MslI `caynn nnrtg`

`[r = a or g; y = c or t; n = a or g or t or c]`

[NB For this exercise all matches must be looped to make sure you don't just find the first cut site, and must be case insensitive as fasta sequences can be either upper or lower case].

Section 5: Subroutines, References and Complex Data Structures

Exercise 10: Subroutines

10a

Write a program containing a subroutine which will take in a string of DNA sequence and will return the reverse complement. The subroutine should check that the string passed to it contains only valid letters (GATC), with no spaces or line breaks. The returned string should keep the same capitalisation as the submitted string.

[NB for converting the string you will need to use the tr function – see the end of Section4]

10b

Write a script containing a subroutine to calculate the mean, variance and standard deviation of an array of numbers. Generate a random data set of between 10 and 20 measures with values between 0 and 100 and pass this to the subroutine.

The formula for calculating the variance of a sample is:

$$v = (\sum (x - m)^2) / (n - 1)$$

v = variance
x = each individual measure
m = mean of all measures
n = number of measures

The Standard Deviation is the square root of the variance.

Exercise 11: References and Complex Data Structures

11a

Write a script containing a subroutine which takes in two lists of values and returns the set of values present in both lists. You will be provided with two sets of gene names (Unigene IDs) representing significantly expressed genes in two expression studies (result1.txt and result2.txt). Find the genes which were expressed in both data sets.

For bonus points annotate the list of common ids with the frequency from each set and the gene’s description (where one is present).

11b

You will be provided with a file (tissues.txt) containing the official list of all tissues. Each entry consists of an official name (term), an identifier, a definition and a pubmed reference to the definition. Write a script which parses this file and puts all of the information in it into a suitable data structure. All entries in this file are on a single line. Lines starting with ! are comments.

Use this datastructure as part of a script which prompts the user for a tissue name. If they enter a valid name return all the details for that tissue. If they enter a partial name (eg ‘ov’) then return them the list of possible completions (ovary, ovary cancer cell, OVCAR-3 cell, OVCAR-5 cel, oviduct, ovotestis). All matches should be case insensitive.

Section 6: Perl Modules

Exercise 12: Perl Modules

12a

Use the `Math::Trig` module to provide a value of PI you can use to calculate the area of a circle of radius 5cm.

12b

"Acniocdrg to rceresah at an Eslnigh Uisvrnitey, it dseon't mteatr in waht oerdr the ltrtees in a wrod are, the olny ipoatnrmt tnhig is taht the fsirt and and lsat ltteer is at the rhgit plcae. The rset can be a toatl mses and you can stlil raed it wtiuhot pbrelom. Tihs is bsaceue we do not raed erevy lteer by iseltf but the wrod as a wlohe."

Write a program which will take a section of text and will mix it up to read like the above example. To mix each word you should split it into an array of letters and use the `List::Util` module to shuffle the contents (excluding the first and last letters). Use the `Text::Wrap` module to write out the reformatted string to fill 80 char columns.

12c

The `LWP::UserAgent` module allows you to retrieve web pages into your Perl programs.

Write a script which will retrieve the code behind the NCBI database statistics page at http://www.ncbi.nlm.nih.gov/genbank/statistics and can extract automatically from the table at the bottom the number of bases and sequences which were in GenBank in Feb 2014. What you'll get back from the server is the raw HTML behind the page so look through that for Feb 2014 and then work out how to get the information next to that programmatically.

12d

Find out how good the perl `rand()` function is. Find and install a module from CPAN which will perform a Chi-Square test. Write a script which simulates rolling a dice 10,100,1000 and 10000 times and see how random the chi-square test says the data is.

For bonus points use the `Net::Random` module to get some truly random data from random.org for the same test. See how the results for the real random numbers stack up against `rand()`. Because `Net::Random` fetches information from an external web site please only do this part of the exercise once you're sure that you've got the `rand()` version working properly. We don't want to go upsetting the people at random.org.

12e

Write a module called `DNA::Manipulate`. This should provide two subroutines:

1. `revcomp` should take a DNA sequence and return its reverse complement (you wrote the code for this in exercise 10a)

2. `translate` should take a DNA sequence and a frame (1,2,3) and return a protein translation of that frame.

Install the module and check you can call it from another program.

If you're feeling bold then try rewriting the module as an Object-Oriented module where the DNA sequence is passed to the `new()` method and is then stored for use in the other methods.

Section 7: Interacting with external programs

Exercise 13: External Programs

13a
Write a program which will take a list of text documents (use some of the ones from previous exercises and will open them consecutively in notepad. The notepad program can be found at C:/Windows/notepad.exe and it will open any file given as the first argument to it when it is launched. Check that notepad exited cleanly.

13b
Use the ‘date’ and ‘time’ programs to collect the time and date and print them out together. Both programs should be run with the /T switch so that they don’t prompt you to change the date/time after they have been displayed.

13c
The ‘help’ program displays a help page about any available Windows command. There can be a lot of information presented, but the first line returned is always a quick summary of what the command does. Write a script which uses the help program to find the function of the following commands:

- help
- attrib
- dir
- assoc

13d
The nslookup.exe program will find the IP address of any computer name given to it. It’s output looks like this (the IP address in red is the one you want):

```
M:\>nslookup bilinws3.babraham.bbsrc.ac.uk
Server:  biifs.babraham.bbsrc.ac.uk
Address:  149.155.144.29

Name:    bilinws3.babraham.bbsrc.ac.uk
Address:  149.155.147.34
```

Write a script which will take in a list of computer names and, using nslookup, will identify their IP addresses. Use it to find the addresses of the following machines:

- www.ebi.ac.uk
- rscb.org
- seqanswers.com

13e
The ‘driverquery’ command returns a list of all the currently loaded drivers on your system. Write a script which will pull out a list of all the driver modules for the file system (those listed as “file system” in the “Driver Type” column. The data from this command is fixed width, not delimited, so you will need to use substr rather than split to parse it.