

Bio 595

Computers in Biomedical Research

Fall 2003

Slides for Monday, November 3

Introduction to PERL

Description of Perl (Perl 1, 1987, by author Larry Wall)

“Perl is a interpreted language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information. The language is intended to be practical (easy to use, efficient, complete) rather than beautiful (tiny, elegant, minimal). It combines (in the author's opinion, anyway) some of the best features of C, sed, awk, and sh, so people familiar with any of those languages should have little difficulty with it. Language historians will also note some vestiges of Pascal, and BASIC-PLUS. Expression syntax corresponds quite closely to C expression syntax.”

Installing Perl

See if perl is already installed on your computer. If so, is it version 5?

Go to perl home page, download, install binary at www.perl.com

Install binary distribution for Win32; use link to ActivePerl page of ActiveState.com (download Win32 binary)

Needs Windows Installer 9x/Me

Current version on website, ActivePerl 5.6.1 build 635 (8.6 MB)

Also download unstuffit if you don't have it already:

<http://www.alladinsys.com>

Installation of perl requires Windows Installer (also available at ActivePerl)

How to run a Perl program

Programs use a filename extension .pl

Launch my_program.pl by typing my_program in the MS-DOS command window, or typing perl this_program.pl

To get MS-DOS command window: under Programs, click on MS-DOS prompt

To quit MS-DOS, type “exit”

Use your choice of text editor: if you use Word, save program as text only (ASCII). Can also use NotePad or other text editors

Programming process in Perl

Identify required inputs (data files, user info)

Write overall design plan: choose the algorithm

Decide what output should look like: a text file? A graphics display?

Refine the design, adding details.

Finally, write the perl code.

Some perl samples

We'll start by examining a few simple programs in perl:

Transcribe DNA to RNA

Concatenate sequences

Make reverse complement of sequences

Read sequence data from files

Ask how GC-rich your DNA is

Ask how hydrophobic your protein is

DNA sequence data representation

IUB/IUPAC nucleic acid codes

A adenine adenosine

C cytosine cytidine

G guanine guanosine

T thymine thymidine

U uracil uridine

Peptide sequence data representation

IUB/IUPAC amino acid codes

A	ala	alanine
B	asx	aspartic acid or asparagine
C	cys	cysteine
D	asp	aspartic acid
E	glu	glutamic acid
F	phe	phenylalanine
G	gly	glycine
H	his	histidine
I	ile	isoleucine
K	lys	lysine
L	leu	leucine
M	met	methionine

Peptide data representation continued

N	asn	asparagine
P	pro	proline
Q	glu	glutamine
R	arg	arginine
S	ser	serine
T	thr	threonine
V	val	valine
W	trp	tryptophan
X	xxx	unknown
Y	tyr	tyrosine
Z	glx	glutamic acid or glutamine
-	---	gap of indeterminate length

Store a DNA sequence

```
# !/usr/bin/perl -w  
# Storing DNA in a variable, and printing it out  
# First we store the DNA in a variable called $DNA  
$DNA =  
    'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';  
# Next we print the DNA sequence on the screen  
print $DNA;  
# Finally, we quit the program  
    exit;
```

Run program in Windows

Type in an MS-DOS command window, perl example1

First line with `#!` is for Unix or Linux-based computers; in Windows command window, `perl my_program` works; this line makes programs platform-compatible

Flag `-w` stands for “warnings” – show error messages (or, “use warnings”)

Observe:

Perl statements end in semicolons.

Variable names use letters, numbers, underline; first character must be a letter.

Scalar variables prefixed by `$`

Strings in single or double quotes.

Concatenating DNA strings

```
# !/usr/bin/perl -w  
# Concatenating DNA  
# Store DNA fragments into the two variables $DNA1 & $DNA2  
$DNA1 = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';  
$DNA2 = 'ATAGTGCCGTGAGAGTGATGTACTA';  
print $DNA1, "\n";  
print $DNA2, "\n\n";  
# Concatenate DNA fragments into a third variable and print.  
$DNA3="$DNA1$DNA2";  
# An alternative way using the "dot operator":  
$DNA3 = $DNA1 . $DNA2;  
exit;
```

Regarding the last example...

\n signifies a “newline”; must use double quotes.

Commas print all items in a list

String interpolation – variable followed by variable

Transcribing DNA to RNA

**Here reverse complement strand codes mRNA –
equivalent to complement of a complement, so Ts
could be replaced with Us.**

Transcribing DNA into RNA

```
# !/usr/bin/perl -w  
# Transcribing DNA into RNA  
# here's the DNA  
$DNA = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';  
print "Here is the starting DNA:\n\n";  
print "$DNA\n\n";#transcribe the DNA by substituting all Ts with Us  
$RNA = $DNA;  
$RNA = ~s/T/U/g;  
#print the RNA on screen  
print "Here is the result of transcribing the DNA to RNA:\n\n";  
print "$RNA\n";  
exit;
```

Regarding the last example...

Meaning of line `$RNA = ~s/T/U/g;`

Binding operator `=~`: apply operation on right to string in variable on the left.

Substitution operator: `s` indicates this is a substitution.

`T` indicates element to be substituted.

`U` is element that will replace the `T`

The `g` stands for “global”, a modifier specifying the substitution is to be made throughout the entire string.

The “ / ” separates arguments within the command.

Note that it's common to check a DNA database with reverse complement of query – you may have in hand the opposite strand of some known gene.

Does this algorithm work?

```
# !/usr/bin/perl -w  
# calculating reverse complement of DNA strand  
$DNA = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';  
print "Here is the starting DNA:\n\n";  
print "$DNA\n\n";  
# Calculate reverse complement  
$revcom = reverse $DNA;  
#now substitute bases by their complement:  
# A → T, T → A, C → G, G → C  
#revcom =~s/A/T/g;  
$revcom =~s/T/A/g;  
$revcom =~s/G/C/g;  
$revcom =~s/C/G/g;  
print "Here is the reverse complement: \n\n";  
$revcom = reverse $DNA;  
print $revcom\n";  
exit;
```

No, but this one does

```
# !/usr/bin/perl -w  
# calculating reverse complement of DNA strand  
# the DNA:  
$DNA = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';  
# print the DNA  
print "Here is the starting DNA:\n\n";  
print "$DNA\n\n";  
# Calculate reverse complement  
# this first method is a bad algorithm that doesn't work  
$revcom = reverse $DNA;  
#now substitute bases by their complement:  
# A → T, T → A, C → G, G → C  
$revcom =~tr/ACGTacgt/TGCAtgca/;  
print "Here is the reverse complement: \n\n";  
print $revcom\n";  
exit;
```

Regarding the last example...

In this example check result by reading second strand backwards- it's the complement of the first DNA strand.

“reverse” creates a reverse order sequence from a string
“tr” translates one set of characters into a second set.

Next time

Proteins, files and arrays: reading protein files

Files can be on Zip, CD, floppy, etc.

Save a file; call it NM_021964/fragment.pep

This is part of the human zinc finger protein

File is from GenBank ID