

LAB 3 – Perl III

To get the lab approved, send your code to: david.sundell@plantphys.umu.se

Installation

OBS: On the course computers these programs are already installed!

ActivePerl: <http://www.activestate.com/activeperl>

Open Perl IDE: <http://open-perl-ide.sourceforge.net/>

BioPerl using the Perl Package Manager: http://www.bioperl.org/wiki/Installing_Bioperl_on_Windows

BLAST: <ftp://ftp.ncbi.nlm.nih.gov/blast/executables/release/LATEST/> (OBS: This lab does not work with the new Blast+). Install BLAST to e.g. c:/BLAST by creating the folder "BLAST" and executing the installation file in that folder (installing under e.g. "Program Files" might cause problems due to the space character). To get BLAST to work you need to set the following system paths: BLASTDB, DATA and PATH: see http://www.ncbi.nlm.nih.gov/staff/tao/URLAPI/pc_setup.html

Download the Yeast proteome (yeast.aa.gz), unzip and place the file in the folder pointed to by BLASTDB: <ftp://ftp.ncbi.nlm.nih.gov/blast/db/FASTA/>. Format the Yeast database file so that BLAST can use it. Run "formatdb -i yeast.aa -p T" in the database folder (the folder pointed to by BLASTDB) using the Windows command line.

Task 1 – Regular expressions

a) Regular expressions are typically used in if-sentences to test for matches in a string:

```
my $text = "InternationaleBayernMunchen";
if ($text =~ /Inter/) {
    print "Inter plays the final\n";
} else {
    print "Mourinho is sad\n";
}
```

Anything that matches a regular expression inside parentheses is stored in the scalars \$1, \$2, ...

```
if ($text =~ /Internationale(.*)$/) {
    print "Inter plays $1 in the final\n";
}
```

```
} else {  
    print "Mourinho is sad\n";  
}
```

Regular expressions are also commonly used to substitute parts of a string with another string:

```
$text =~ s/Internationale/ACMilan/;  
print "Wishful thinking: $text\n";
```

Note that the `=~` operator is not an assignment. It returns the matches/substitutions:

```
my @matches = ($text =~ m/a/g);  
print "No. as in the text: @matches\n";
```

The `g` option ensures that all occurrences are found.

Now you try it. Construct regular expressions (match operators) for the following and test them on examples:

b) any string that contains an "a" or "b" followed by any 2 characters followed by an "a" or a "b". The strings "axxb", "alfa" and "blka" match, and "ab" does not.

c) upper case "A" followed by anything except "x", "y" or "z".

d) string that contains nothing but white space.

e) an HTML anchor tag (e.g. ``) and prints the address (e.g. `www.umu.se`).

f) Write a perl program that reads in the HTML file `hvidsten.com.html` and replaces all `<H5>`,`</H5>` tag pairs with `<H1>`,`</H1>` tags.

Task 2 – BioPerl beginners version

a) Use the non-object oriented beginners version of BioPerl (i.e. `Bio::Perl`) to read in the GenPept accession numbers in the file `genpept.txt`, retrieve their corresponding sequences from GenPept and write them to a file. You can find help here: http://classes.soe.ucsc.edu/bme060/Winter07/bptutorial.html#i.2_quick_getting_started_scripts

b) Repeat Task 2a using the object oriented, full version of BioPerl. You can find help here: <http://www.bioperl.org/wiki/HOWTO:Beginners>

Task 3 – Searching for paralogs in Yeast.

Use BioPerl (the object-oriented version) to find paralogs in Yeast. That is, run BLAST on pairs of Yeast sequences and retrieve (and write to file) the ones that have a significant E-score.

You can find help here: <http://www.bioperl.org/wiki/HOWTO:Beginners>

and here: <http://www.bioperl.org/wiki/HOWTO:SearchIO>