

Bio 595

Computers in Biomedical Research Fall 2003 Slides for Friday, December 5

FASTA format

Lines of sequence data contain newlines so the data can be printed or displayed on a screen; plus,
Header information: starts with '>' character; usually contains name of the DNA or gene it comes from; often uses vertical separator bar (|) to provide other information like the experiment that produced the data. See 'sample.dna' file provided.

Enter data into an array, or process line by line (best for really big files)

What does the program do? Open and read file into an array (1st subroutine), and then extract DNA data (2nd subroutine)

Notes about subroutines in ex26

Declare these 'my variables' in \$line:

```
Foreach my $line (@fasta_file_data) {
```

Here's a regular expression that eliminates non-sequence information: the \s matches whitespace – space, tab, formfeed, carriage return, newline, the '^' matches the beginning of the line, the '\$' matches the end of the line:

```
if($line =~ /^!\s*S/) {
```

This expression has nothing or just whitespace at beginning of a line, up to pound sign:

```
} elsif($line =~ /^!\s*#/) {
```

This expression matches the > sign at the beginning:

```
} elsif($line =~ /^!>/) {
```

And this removes whitespace, including newlines:

```
$sequence =~ s/\s//g;
```

Print format subroutine: see slide 9 for ex27

sample.dna file

```
> sample dna (This is a typical fasta header.)
> Fasta-style headers can go on for multiple lines.
agatggcggcgctgaggggtcttgggggctctaggccggccactactgg
tttgacgggagcagcgaatggggcctgcgcaataggagtacgctgctt
gggaggcgtgactagaagcggaaagttagttgtggcgcccttgcaaccg
tgggacgcccggagtggtctgtgcaagttcgcggtcgcgtgcccgggt
cgtgagggagtgccgggagggagata tgaaggagatggttcaagcc
cagagcctccagatgcccgggaggacagcaagtccgaaatggggagaat
gocccactctactgcatctgcccgaacccggaactcaactgctctatgat
cgggtgtgacaactgcaatgagttgttccatgggactgcaaccggaatca
ctgagaagatggccaaggccatccgggagtggtactgctgggagtgca
gagaagaccaccaagctagagattcgcatacggcacaagaagtcaaggga
gocggatggcgaatgagcgggacagcagtgagccccgggatgagggtggag
ggcgcaagaggcctgtccctgataccagacctgcaagcggcggcagggtca
gggacaggggtggggccatgctgctcggggctctgcttgcgcccaaa
atcctctccggcggccttggggcacaaccagcaagcactcaacagcagc
agcagcagcagatcaaacggtaagcccgcaatggtgtgtagtgtaggca
tgtggcgcactgagagctgtgttcaactgtgattctgtgggaactgaa
gaagttcggggggcccaacaagaatccgggaagaagtcggcgtgcccagt
gocagctgcccgggggaaatgatacaagactcctctctcgtctca
ccagtgagccctcagagttccctgccaaggccccggcggcactgcccac
ccaacagcagccacagccatcaacagaagttagggcgcatccgtgaagatg
agggggcagttggcgtcatcaacagtcaggagcctcctgaggctacagcc
accctgagccactctagatgaggaacta
```

ex26 main program

```
#!/usr/bin/perl -w
# Example ex26: read a fasta file and extract the
sequence data
use strict;
use warnings;
# use BeginPerlBioinfo;
# Declare and initialize variables
my @file_data = ( );
my $dna = '';
# Read in the contents of the file "sample.dna"
@file_data = get_file_data("sample.dna");
# Extract the sequence data from the contents of the
file "sample.dna"
$dna = extract_sequence_from_fasta_data(@file_data);
# Print the sequence in lines 25 characters long
print_sequence($dna, 25);
exit;
```

get FASTA data subroutine

```
# get_file_data: a subroutine to get FASTA data from a
file given its filename
sub get_file_data {
    my ($filename) = @_;
    use strict;
    use warnings;
    # Initialize variables
    my @filedata = ( );
    unless( open(GET_FILE_DATA, $filename) ) {
        print STDERR "Cannot open file
\"$filename\"\n\n";
        exit;
    }
    @filedata = <GET_FILE_DATA>;
    close GET_FILE_DATA;
    return @filedata;
}
```

extract FASTA data from array subroutine

```
# a subroutine to extract FASTA sequence data from an array
sub extract_sequence_from_fasta_data {
    my($fasta_file_data) = @_;
    use strict;
    use warnings;
    # Declare and initialize variables
    my $sequence = '';
    foreach my $line ($fasta_file_data) {
        # discard blank line
        if ($line =~ /\s*$/) {
            next;
        }
        # discard comment line
        elsif($line =~ /^#/) {
            next;
        }
        # discard fasta header line
        elsif($line =~ />/) {
            next;
        }
        # keep line, add to sequence string
        else {
            $sequence .= $line;
        }
    }
    # remove non-sequence data (in this case, whitespace) from $sequence string
    $sequence =~ s/\s//g;
    return $sequence;
}
```

print sequence data subroutine

```
# print_sequence: a subroutine to format and print
sequence data
sub print_sequence {
    my($sequence, $length) = @_;
    use strict;
    use warnings;
    # Print sequence in lines of $length
    for ( my $pos = 0 ; $pos < length($sequence) ;
        $pos += $length ) {
        print substr($sequence, $pos, $length), "\n";
    }
}
```

Ex27 read a FASTA file

```
#!/usr/bin/perl -w
# ex 27: read a fasta file and extract the DNA sequence data
# translate it to protein, print it out in 25-character-long lines
use strict;
use warnings;
# Initialize variables
my @file_data = ( );
my $dna = '';
my $protein = '';
# Read in the contents of the file "sample.dna"
@file_data = get_file_data("sample.dna");
# Extract sequence data from contents of the file "sample.dna"
$dna = extract_sequence_from_fasta_data(@file_data);
# Translate the DNA to protein
$protein = dna2peptide($dna);
# Print the sequence in lines 25 characters long
print_sequence($protein, 25);
exit;
```

Examine all reading frames of DNA

Very little of DNA in the human genome make up genes. Also, genes are often fragmented, and peptides are spliced together during transcription and translation. Usually we don't know where translation is to start: at the first base? Second? Third? (Fourth is same as first.) Each starting point gives different triplet sequence, thus a different amino acid sequence. And transcription, translation can occur from either strand. So, in all, there are six possible sequences to examine when looking for coding regions. So, strategy is to look at all six possibilities, check for long amino acid sequences that lack 'stop' codons: these stretches are ORFs (open reading frames)

translate frame subroutine

```
# translate_frame: a subroutine to translate a frame
of DNA
sub translate_frame {
    my($seq, $start, $end) = @_;
    my $protein;
    # To make the subroutine easier to use, you won't
    need to specify
    # the end point-it will just go to the end of the
    sequence
    # by default.
    unless($end) {
        $end = length($seq);
    }
    # Finally, calculate and return the translation
    return dna2peptide
        (substr($seq,$start-1,$end-$start +1));
}
```

ex28: translate DNA into all six reading frames

```
#!/usr/bin/perl -w
# ex28: translate a DNA sequence in all six reading frames
# Initialize variables
my @file_data = ( );
my $dna = '';
my $revcomp = '';
my $protein = '';
# Read in the contents of the file "sample.dna"
@file_data = get_file_data("sample.dna");
# Extract the sequence data from the contents of the file "sample.dna"
$dna = extract_sequence_from_fasta_data(@file_data);
# Translate the DNA to protein in six reading frames
# and print the protein in lines 70 characters long
print "\n -----Reading Frame 1-----\n\n";
$protein = translate_frame($dna, 1);
print_sequence($protein, 70);
print "\n -----Reading Frame 2-----\n\n";
$protein = translate_frame($dna, 2);
print_sequence($protein, 70);
print "\n -----Reading Frame 3-----\n\n";
$protein = translate_frame($dna, 3);
print_sequence($protein, 70);
# Calculate reverse complement
$revcomp = revcomp($dna);
print "\n -----Reading Frame 4-----\n\n";
$protein = translate_frame($revcomp, 1);
print_sequence($protein, 70);
print "\n -----Reading Frame 5-----\n\n";
$protein = translate_frame($revcomp, 2);
print_sequence($protein, 70);
print "\n -----Reading Frame 6-----\n\n";
$protein = translate_frame($revcomp, 3);
print_sequence($protein, 70);
exit;
```