Perl for Biologists

Session 12

May 20, 2015

Interacting with websites and ftp sites

Jaroslaw Pillardy

Session 11 Exercise 1. Retrieve an E. coli genome from NCBI (Genbank accession NC_000913). Make a fasta file with 500bp upstream regions of all transcripts. Hint: You can do this by modifying script1.pl of this lecture.

```
#!/usr/local/bin/perl
use strict;
use warnings;
use Bio::Perl;

my $db = Bio::DB::GenBank->new();
my $seqobj = $db->get_Seq_by_acc('NC_000913');
my $seqlen = $seqobj->length();
my $out = Bio::SeqIO->new(-file => ">upstream500.fasta" ,
                                  -format => 'fasta');
LOOP:for my $feat_object ($seqobj->get_SeqFeatures) {
   if ($feat_object->primary_tag eq "gene") {
                my ($gene_name) = $feat_object->get_tag_values("gene");
                my $start = $feat_object->start();
                my $end = $feat_object->end();
                my $strand =$feat_object->strand();
                my ($fragstart, $fragend);
                my  $new_seqobj;
```

```
            if ($strand==1) {
                    $fragstart = $start-500;
                    $fragend = $start-1;
                    if ($fragstart<1) {next LOOP;}
                    $new_seqobj = $seqobj->trunc($fragstart, $fragend);
            }
            elsif ($strand==-1) {
                    $fragstart = $end+1;
                    $fragend = $end+501;
                    if ($fragend>$seqlen) {next LOOP;}
                    $new_seqobj = $seqobj->trunc($fragstart, $fragend)->revcom();
        }
            else {
                    print "warning:wrong strand at $gene_name\n";
                    next LOOP;
        }
        $new_seqobj->display_id("$gene_name.500bp");
        $new_seqobj->desc("");
            $out->write_seq($new_seqobj);
  }
}
```

Session 11 Exercise 2. Modify script4.pl, so that this script can take in a third parameter maximum evalue, and only output HSP with evalue below the cutoff.

```
#!/usr/local/bin/perl
use Bio::SearchIO;
use strict;
use warnings;
my ($infile, $outfile, $evalue_cutoff) = @ARGV;

open OUT, ">$outfile";
$,="\t";

my $searchio = Bio::SearchIO->new(-format => 'blast',
                                  -file   => $infile);
while (my $result = $searchio->next_result)
{

        # Get info about the entire report
        my $query_name = $result->query_name;
        my $query_length = $result->query_length;


        # get info about the first hit
        while (my $hit = $result->next_hit)
        {
                my $hit_name = $hit->name;
                my $hit_length = $hit->length;
```

```
# get info about the first hsp of the first hit
LOOP2:while (my $hsp = $hit->next_hsp)
{
        my $rank = $hsp->rank;
        my $num_conserved = $hsp->num_conserved ;
        my $num_identical= $hsp->num_identical ;
        my $hsp_length= $hsp->hsp_length ;
        my $bits= $hsp->bits ;
        my $evalue = $hsp->evalue ;
        if ($evalue>$evalue_cutoff) {
                next LOOP2;
        }
        my $hsp_qstart = $hsp->query->start;
        my $hsp_qend = $hsp->query->end;
        my $query_strand = $hsp->query->strand;

        my $hsp_hstart = $hsp->hit->start;
        my $hsp_hend = $hsp->hit->end;
        my $hit_strand = $hsp->hit->strand;

        my $query_string = $hsp->query_string ;
        my $hit_string = $hsp->hit_string ;
        my $homology_string = $hsp->homology_string ;
        print OUT 1, $query_name, $hit_name, $query_length,
$hit_length, $rank, $num_identical, $num_conserved, $hsp_length, $bits, $evalue,
$hsp_qstart, $hsp_qend, $query_strand, $hsp_hstart, $hsp_hend, $hit_strand,
$query_string, $hit_string, $homology_string, "", "";
        print  OUT "\n";
}
}
}
```

Network Services

Network service is a way of providing (“exposing”) software functionality on a network.

The software can serve data, accept commands etc.

All the things we do on the Internet are related to network services.

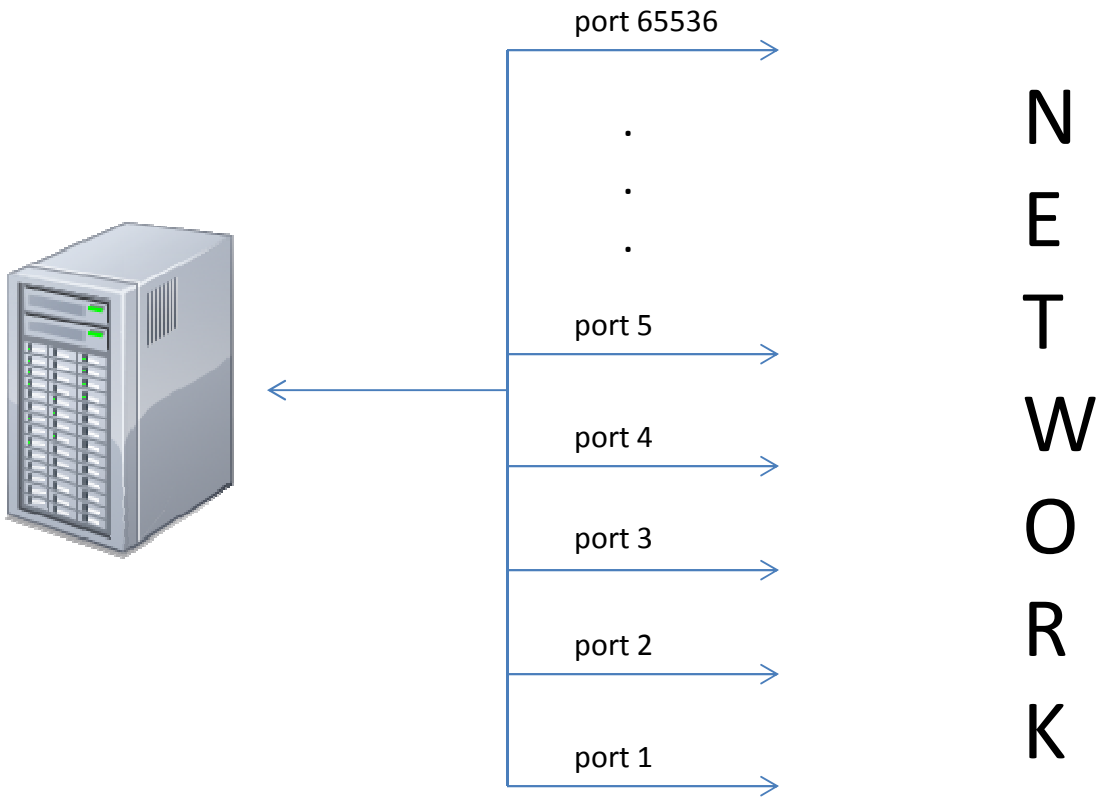
Network Services

Each service is defined by:

Service Protocol: a set of commands and data structures used in communicating with this particular service ("language")

Communication port address: a 16 bit number identifying port address on a computer ("address")

Network Services



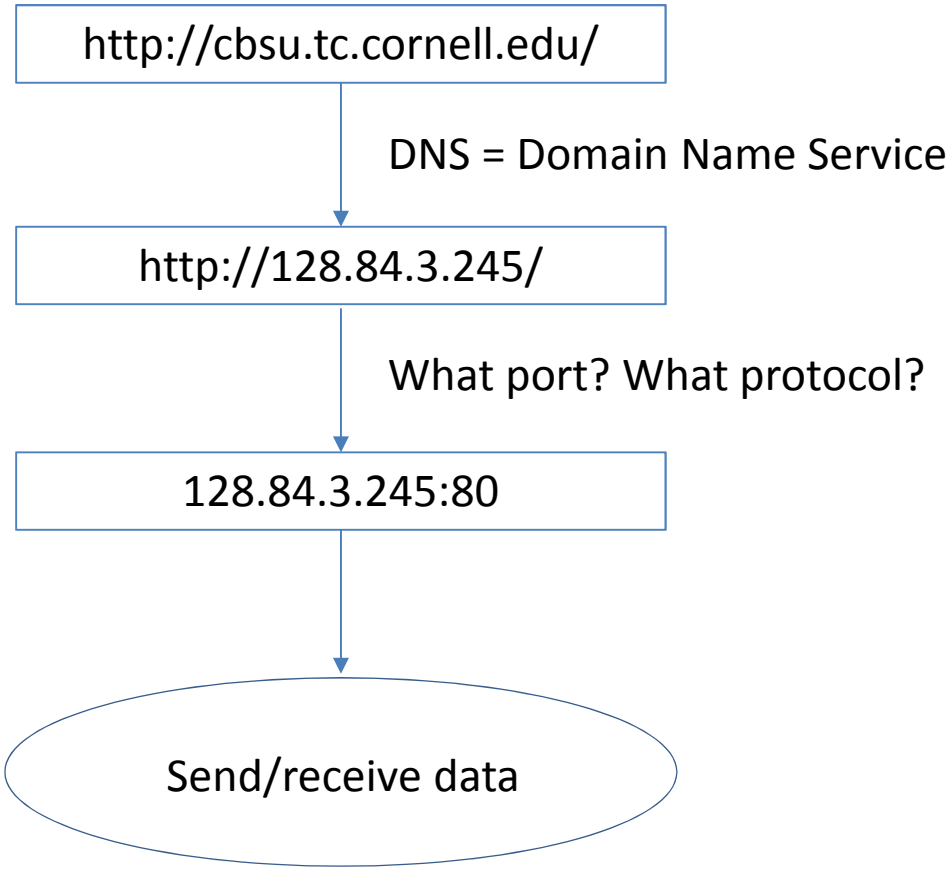
Computer full address: ip_number:port i.e. 128.8.3.22:22

Network Services

Service (protocol)	Port
FTP	20 and 21
TELNET	23
SSH	22
SMTP (mail service)	25
DNS (domain name system)	53
HTTP (www)	80
HTTPS (www secure)	443

Services can be accessed directly using telnet client or generic Perl module – it is just a bi-directional data flow - but then you need to know how to talk to it (follow protocol)

Network Services Example



Example 1: HTTP

server name

service port

```
>telnet   cbsu.tc.cornell.edu   80
Trying 128.84.70.251...
Connected to cbsu.tc.cornell.edu.
Escape character is '^]'.
GET /test1.htm HTTP/1.1
host: cbsu.tc.cornell.edu

HTTP/1.1 200 OK
Content-Type: text/html
Last-Modified: Mon, 18 May 2015 17:29:15 GMT
Accept-Ranges: bytes
ETag: "4a2df9299091d01:0"
Server: Microsoft-IIS/8.0
X-Powered-By: ASP.NET
Date: Mon, 18 May 2015 17:30:43 GMT
Content-Length: 175

<html>
<head>
</head>
<body>
   <h2>Test page</h2>

      <p>This is a <em>test page</em> for testing <strong>plain html</strong></p>
      <p>Enjoy!</p>

</body>
</html>
```

our command in HTTP protocol

Example 2: SMTP

```
>telnet appsmtp.mail.cornell.edu 25
Trying 128.84.106.29...
Connected to appsmtp.mail.cornell.edu.
Escape character is '^]'.
220 alva02.serverfarm.cornell.edu ESMTP Sendmail 8.14.4/8.14.4; Tue, 21 May 2013 14:41:48 -0400
HELO cbsum1c2b014.tc.cornell.edu
250 alva02.serverfarm.cornell.edu Hello cbsum1c2b014.tc.cornell.edu [128.84.43.165], pleased to meet you
MAIL FROM:<jp86@cornell.edu>
250 2.1.0 <jp86@cornell.edu>... Sender ok
RCPT TO:<jarekpp@yahoo.com>
250 2.1.5 <jarekpp@yahoo.com>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Subject: Test

This is a test
.
250 2.0.0 r4LIfmiV016599 Message accepted for delivery
QUIT
221 2.0.0 alva02.serverfarm.cornell.edu closing connection
Connection closed by foreign host.
```

Generic Perl network service module

```
use IO::Socket::INET;


my $socket = new IO::Socket::INET (
        PeerHost => $host,
        PeerPort => $port,
        Proto => 'tcp');
```

this is “transport protocol”
NOT “service protocol”

```
print $socket "Data\n"; #send data to socket

my $txt = <$socket>;     #read data from socket
```

script1.pl (1)

```
#!/usr/local/bin/perl

use IO::Socket::INET;

my $host = "cbsuss05.tc.cornell.edu";
my $port = 80;
my $document = "index.html";

my $socket = new IO::Socket::INET (
        PeerHost => $host,
        PeerPort => $port,
        Proto => 'tcp');

if(! defined $socket)
{
        print "ERROR: Cannot open connection to $host:$port\n";
        exit;
}
print "Connection established\n";
```

```
print $socket "GET /$document HTTP/1.1\n";
print $socket "host: $host\n\n";

my $file = "$host:$port/$document";
$file =~ s/\//_/g;
open out, ">$file";

my $n=0; $m=0;
while(my $txt = <$socket>)
{
        $n++;
        $m += length($txt) - 1;
        print out $txt;
}
print "$n lines with $m characters retrieved\n";
close(out);
$socket->close();
```

Using FTP in Perl

```
#module declaration
use Net::FTP;

#start ftp session
$ftp = Net::FTP->new("server", Passive => 1);

#login with userid and password
#anonymous and e-mail may be used
$ftp->login("userid",'password');

#change current remote directory
$ftp->cwd("directory");

#change transfer mode to binary
#(the other is ascii)
$ftp->binary;
```

Using FTP in Perl

```
#download file
$ftp->get($filename);

#upload file
$ftp->put($filename);

#get information about a file or list directory
#standard wildcards are accepted (*, ?)
$ftp->dir($filename);
$ftp->dir($dir);

#end ftp session
$ftp->quit;
```

Example: Local BLAST databases

- Create a local BLAST database depository
- download databases as fasta files
- databases listed in a file (first line base dir, then databases)
- format databases locally with formatdb
- store fasta file and formatted db in different directories

script2.pl (1)

```
#!/usr/local/bin/perl
use Net::FTP;

my $basedir;
my %dbfiles;

open in, "script2.config" or die "Cannot open config file\n";

my $line = 0;
while(my $txt=<in>)
{
        if($txt =~ /^#/){next;}
        chomp $txt;
        $line++;
        if($line == 1)
        {
                $basedir = $txt;
        }
        else
        {
                my ($file, $type) = split /\t/, $txt;
                $dbfiles{$file} = $type;
        }
}
```

```
#make the new directory
my @date=localtime (time);
my ($year, $month, $day)= ($date[5]+1900,$date[4]+1,$date[3]);
my $date = join '', ($year, sprintf("%02d", $month), sprintf("%02d", $day));
my $newdir="$basedir/$date";
system ("mkdir $newdir");
system ("mkdir $newdir.fasta");

foreach my $db (keys %dbfiles)
{
        get_db($db, $dbfiles{$db}, $newdir);
}
```

time and localtime

- time() function returns number of seconds since 00:00 UTC January 1st 1970.

- localtime($time) converts this integer into more suitable formats:

List context:
($sec, $min, $hour, $mday, $mon, $year, $wday, $yday, $isdst) = localtime(time);

12 1 11 22 4 113 3 141 1

Scalar context:
$date = localtime(time);

see time.pl

Wed May 22 11:01:12 2013;

```
sub get_db
{
        my ($filename, $format_type, $newdir)=@_;
        print "starting on $filename\n";
        my $ftp = Net::FTP->new("ftp.ncbi.nih.gov", Debug => 0, Passive => 1);
        $ftp->login("anonymous",'jp86@cornell.edu');
        $ftp->cwd("/blast/db/FASTA");

        #generate the update file
        my $temp=${filename};
        $temp=~s/\.gz$//i;
        open (OUT, ">$newdir/${temp}.update");

        my ($pub_date) = $ftp->dir($filename);
        chomp $pub_date;
        my @pub_date = $pub_date=~/(\S+)/g;
        $pub_date=join ' ', @pub_date[5, 6, 7];
        print "pub_date=$pub_date\n";
        print OUT "Database was last updated locally on ${date}.\n";
        print OUT "NCBI update date is ${pub_date}.\n";
        close OUT;
```

```
Example of $ftp->dir() string:

-r--r--r--   1 ftp      anonymous     1835 May  1 19:37 vertebrate_mammalian.34.rna.fna.gz
```

```
        $ftp->binary;
        if(!$ftp->get($filename))
        {
                print "Cannot download file\n";
                print "$filename DONE\n";
                $ftp->quit;
                return;
        }
        $ftp->quit;

        print "Downloaded\n";
        if ($filename=~/\.gz$/i)
        {
                system ("gunzip $filename");
                $filename=~s/\.gz$//i;
        }
        system ("formatdb -i $filename -p $format_type -o T");
        print "Formatted\n";
        system ("mv $filename $newdir.fasta");
        system ("mv ${filename}* $newdir");
        print "$filename DONE\n";
}
```

Example: Local RefSeq mammalian databases

- Create a local RefSeq RNA mammalian database – one fasta file
- RefSeq available online as a set of files => download and merge them
- download only new files
- store fasta file and formatted files in different directories

Example: Local RefSeq mammalian databases

```
220 FTP Server ready.
Name (ftp.ncbi.nih.gov:jarekp): anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /refseq/release/vertebrate_mammalian/
250 CWD command successful
ftp> ls *.rna.fna.gz
-r--r--r--   1 ftp      anonymous     1835 May  1 19:37 vertebrate_mammalian.34.rna.fna.gz
-r--r--r--   1 ftp      anonymous 27047299 May  1 19:39 vertebrate_mammalian.4.rna.fna.gz
-r--r--r--   1 ftp      anonymous   127191 May  1 19:39 vertebrate_mammalian.41.rna.fna.gz
-r--r--r--   1 ftp      anonymous    42731 May  1 19:39 vertebrate_mammalian.42.rna.fna.gz
-r--r--r--   1 ftp      anonymous     4182 May  1 19:39 vertebrate_mammalian.44.rna.fna.gz
-r--r--r--   1 ftp      anonymous     2964 May  1 19:41 vertebrate_mammalian.45.rna.fna.gz
-r--r--r--   1 ftp      anonymous      920 May  1 19:42 vertebrate_mammalian.46.rna.fna.gz
-r--r--r--   1 ftp      anonymous     2668 May  1 19:42 vertebrate_mammalian.47.rna.fna.gz
-r--r--r--   1 ftp      anonymous     1138 May  1 19:42 vertebrate_mammalian.48.rna.fna.gz
-r--r--r--   1 ftp      anonymous      461 May  1 19:42 vertebrate_mammalian.49.rna.fna.gz
-r--r--r--   1 ftp      anonymous 14136426 May  1 19:42 vertebrate_mammalian.5.rna.fna.gz
-r--r--r--   1 ftp      anonymous  1224316 May  1 19:42 vertebrate_mammalian.50.rna.fna.gz
-r--r--r--   1 ftp      anonymous   801502 May  1 19:42 vertebrate_mammalian.52.rna.fna.gz
-r--r--r--   1 ftp      anonymous  2276671 May  1 19:42 vertebrate_mammalian.53.rna.fna.gz
-r--r--r--   1 ftp      anonymous 22993662 May  1 19:42 vertebrate_mammalian.54.rna.fna.gz
-r--r--r--   1 ftp      anonymous 19960930 May  1 19:42 vertebrate_mammalian.55.rna.fna.gz
-r--r--r--   1 ftp      anonymous  2598827 May  1 19:42 vertebrate_mammalian.56.rna.fna.gz
-r--r--r--   1 ftp      anonymous  8725849 May  1 19:43 vertebrate_mammalian.6.rna.fna.gz
-r--r--r--   1 ftp      anonymous     3072 May  1 19:45 vertebrate_mammalian.9.rna.fna.gz
-r--r--r--   1 ftp      anonymous    29530 May  1 19:45 vertebrate_mammalian.97.rna.fna.gz
226 Transfer complete
```

script3.pl (1)

```
#!/usr/local/bin/perl

use Net::FTP;

my $basedir='/home/jarekp/blastdb/RefSeq_mammals';
my $datadir='/home/jarekp/blastdb/RefSeq_mammals/data';

print "Downloading data\n";
print "================\n";

chdir($basedir);

my %filesize;
my %filedate;
#keep file info in refseq.files
#filename \t size \t date
open fh, "<refseq.files";
my $nloc = 0;
while(my $txt = <fh>)
{
        chomp $txt;
        my ($file, $size, $date) = split /\t/, $txt;
        $filesize{$file} = $size;
        $filedate{$file} = $date;
        $nloc++;
}
close(fh);
print "$nloc local files read\n";
```

script3.pl (2)

```
$ftp = Net::FTP->new("ftp.ncbi.nih.gov", Debug => 0, Passive => 1);
$ftp->login("anonymous",'jp86@cornell.edu');
$ftp->cwd("/refseq/release/vertebrate_mammalian/");

my $i = 0;
my $newfiles = 0;
my @dirlist = $ftp->dir("*.rna.fna.gz");
foreach my $entry (@dirlist)
{
        $i++;
        print "$i ";
        @pub_date = $entry=~/(\S+)/g;
        my $date1=join '_', @pub_date[5, 6, 7];
        my $size1 = $pub_date[4];
        my $filename = $pub_date[8];
        print " $size1 $date1 $filename ";
        #check if the file is new
        if($filesize{$filename} eq $size1 && $filedate{$filename} eq $date1)
        {
                print "same as old\n";
                next;
        }
```

```
        if(! defined $filesize{$filename})
        {
                print "new file\n";
        }
        else
        {
                print "file changed\n";
                system ("rm $datadir/$filename");
        }
        $newfiles++;
        $ftp->binary;
        if(!$ftp->get($filename))
        {
                print "ERROR downloading file $filename\n";
                exit;
        }
        system ("gunzip $filename");
        $filesize{$filename} = $size1;
        $filedate{$filename} = $date1;
        $filename=~s/\.gz$//i;
        system ("mv $filename $datadir");
}
$ftp->quit;
```

```
if($newfiles>0)
{
        print "Formatting the new database now ...\n";
        system("rm RefSeq_mammalian.rna");
        foreach my $file (keys %filesize)
        {
                system("cat $datadir/$file >> RefSeq_mammalian.rna");
        }
        system("formatdb -i RefSeq_mammalian.rna -p F -o T");
        print "Saving new settings ...\n";
        open fh, ">refseq.files";
        foreach my $file (keys %filesize)
        {
                print fh "$file\t$filesize{$file}\t$filedate{$file}\n";
        }
        close(fh);
}

print "DONE\n";
```

Using HTTP in Perl: LWP module

```
#module declaration
use LWP;

#start session (initialize virtual browser)
$ua=LWP::UserAgent->new;

#define browser properties
$ua->agent("MyApp/0.1");

#prepare request
$req = HTTP::Request->new(GET => $url);
```

Using HTTP in Perl: LWP module

```
#set request properties
$req->header(Accept => "text/html, /*;q=0.1");
$req->header(Accept => "application/octet-
                stream, /*;q=0.1");

#execute request
$res = $ua->request($req);

#request status
$res->is_success

#request result (object retrieved)
$res->content
```

Example: automatic check for new software release (Bowtie2)

- Automatically check software website if a new version of the program is released

- Find out how to search the page by examining the web page source http://bowtie-bio.sourceforge.net/bowtie2/index.shtml

- Notify by e-mail if there is a new version

http://bowtie-bio.sourceforge.net/bowtie2/index.shtml

Bowtie 2: fast and sensitive ...

File Edit View Favorites Tools Help

Freeville, NY 10 Day Weat... Freeville, New York (13068... CBSU @ Cornell University Dryden Fish and Game Club Google Maps Ithaca NY Cloud Cover

Bowtie 2

Fast and sensitive read alignment

JOHNS HOPKINS UNIVERSITY

Bowtie 2 is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences. It is particularly good at aligning reads of about 50 up to 100s or 1,000s of characters, and particularly good at aligning to relatively long (e.g. mammalian) genomes. Bowtie 2 indexes the genome with an FM Index to keep its memory footprint small: for the human genome, its memory footprint is typically around 3.2 GB. Bowtie 2 supports gapped, local, and paired-end alignment modes.

OSI certified

Lighter released

- Lighter is an extremely fast and memory-efficient program for correcting sequencing errors in DNA sequencing data. For details on how error correction can help improve the speed and accuracy of downstream analysis tools, see the paper in Genome Biology. Source and software available at GitHub.

Version 2.2.5 - Mar 9, 2015

- Fixed some situations where incorrectly we could detect a Mavericks platform.
- Fixed some manual issues including some HTML bad formating.
- Make sure the wrapper correctly identifies the platform under OSX.
- Fixed --rg/--rg-id options where included spaces were incorrectly treated.
- Various documentation fixes added by contributors.
- Fixed the incorrect behavior where parameter file names may contain spaces.
- Fixed bugs related with the presence of spaces in the path where bowtie binaries are stored.
- Improved exception handling for missformated quality values.
- Improved redundancy checks by correctly account for soft clipping.

Version 2.2.4 - Oct 22, 2014

- Fixed a Mavericks OSX specific bug caused by some linkage ambiguities.
- Added lz4 compression option for the wrapper.
- Fixed the vanishing --no-unal help line.
- Added the static linkage for MinGW builds.
- Added extra seed-hit output.
- Fixed missing 0-length read in fastq --passthrough output.

Site Map

Home
News archive
Manual
Getting started
Frequently Asked Questions
Tools that use Bowtie

Latest Release

Bowtie2 2.2.5 3/9/15

Please cite: Langmead B, Salzberg S. Fast gapped-read alignment with Bowtie 2. *Nature Methods*. 2012, 9:357-359.

Related Tools

Bowtie: Ultrafast short read alignment
Crossbow: Genotyping, cloud computing
Myrna: Cloud, differential gene expression
Tophat: RNA-Seq splice junction mapper
Cufflinks: Isoform assembly, quantitation
Lighter: Fast error correction

Indexes

100%

```
[……]
</ul>

<h2>Version 2.2.5 - Mar 9, 2015</h2>
   <li>Fixed some situations where incorrectly we could detect a Mavericks
platform.</li>
   <li>Fixed some manual issues including some HTML bad formating.</li>
   <li>Make sure the wrapper correctly identifies the platform under OSX.</li>
   <li>Fixed --rg/--rg-id options where included spaces were incorrectly
treated.</li>
   <li>Various documentation fixes added by contributors.</li>
   <li>Fixed the incorrect behavior where parameter file names may contain
spaces.</li>
   <li>Fixed bugs related with the presence of spaces in the path where bowtie
binaries are stored.</li>
   <li>Improved exception handling for missformated quality values.</li>
   <li>Improved redundancy checks by correctly account for soft clipping.</li>
</ul>

<h2>Version 2.2.4 - Oct 22, 2014</h2>
<ul>
   <li>Fixed a Mavericks OSX specific bug caused by some linkage ambiguities.</li>
   <li>Added lz4 compression option for the wrapper.</li>
   <li>Fixed the vanishing --no-unal help line.</li>
   <li>Added the static linkage for MinGW builds.</li>
   <li>Added extra seed-hit output.</li>
   <li>Fixed missing 0-length read in fastq --passthrough output.</li>
   <li>Fixed an issue that would cause different output in -a mode depending on
random seed.</li>
</ul>

<h2>Version 2.2.3 - May 30, 2014</h2>
<ul>
```

script4.pl (1)

```
#!/usr/local/bin/perl
use LWP;

#what is our current version?
open in, "script4.data";
my $version = <in>;
chomp $version;
close(in);

my $ua = LWP::UserAgent->new;
$ua->agent("MyApp/0.1 ");
my $url = "http://bowtie-bio.sourceforge.net/bowtie2/index.shtml";
my $req = HTTP::Request->new(GET => $url);
$req->header(Accept => "text/html, */*;q=0.1");
my $res = $ua->request($req);
my $message = "";
my $subject = "";
if ($res->is_success)
{
        my $n = index($res->content,"<h2>Version");
        if($n == -1)
        {
                $subject = "Bowtie2 version check error";
                $message = "Cannot find version string\n";
        }
        else
        {
```

script4.pl (2)

```
                my $current = substr($res->content, $n+4);
                $current = substr($current, 0, index($current, "</h2>"));
                if($current ne $version)
                {
                        $subject = "New Bowtie2 version available!";
                        $message = "New Bowtie2 found: $current\n";
                        open out, ">script4.data";
                        print out "$current\n";
                        close(out);
                }
        }
}
else
{
        $subject = "Bowtie2 version check error";
        $message = "Cannot open Bowtie2 web page $url\n";
}

if($message ne "")
{
        use Mail::Sendmail;
        %mail = (To => 'jarekpp@yahoo.com',
                From => 'jp86@cornell.edu',
                subject => $subject,
                Message => $message,
                smtp=>'appsmtp.mail.cornell.edu');
        sendmail(%mail);
}
```

This module works like a function library

No objects, just a function

Example: submit a form to retrieve data
get information about Cornell netid owner

- Retrieve basic information for a list of Cornell netids
- Find out how to submit the form by examining the web page source

http://www.cornell.edu/search/people.cfm?netid=rb299

Cornell University | Search ...

File Edit View Favorites Tools Help

Freeville, NY 10 Day Weat... Freeville, New York (13068... CBSU @ Cornell University Dryden Fish and Game Club Google Maps Ithaca NY Cloud Cover

About Cornell Admissions Academics Research Public Engagement Student Life

rb299

ALL PEOPLE EVENTS UNITS LOCATIONS

Robert Bukowski

vCard

General Information

TYPE:	academic
NETID:	rb299
EMAIL:	bukowski@cornell.edu

Phone Numbers

PRIMARY:	607/254-4715

Addresses

CAMPUS:	Frank H T Rhodes Hall, Room 620

Employment Information

SUPERVISORY ORGANIZATION:	RSRCH - Biotechnology Center
JOB TITLE:	Research Associate Sr
BUSINESS TITLE:	Research Associate,Sr

Other Information

CREATED:	9 years ago
MODIFIED:	2 days ago

75%

```
<div id="searchresults">


<div id="peoplename" style="margin-bottom: 1.5em;">
<h3 class="cu-headline" style="display: inline; margin-right: 1em;">Robert
Bukowski</h3>

<a href="vcard.cfm?netid=rb299">vCard</a>

</div>

<div id="peopleprofile">

<div id="general-block">
<h3>General Information</h3>

<table id="generalinfo" class="cu-table">
<tr>
<th>TYPE:</th>
<td>academic</td>
</tr>

<tr>
<th>NETID:</th>
<td>rb299</td>
</tr>
<tr>
<th>EMAIL:</th>
<td><a href="mailto:bukowski@cornell.edu">bukowski@cornell.edu</a></td>
</tr>
```

script5.pl (1)

```
#!/usr/local/bin/perl
use LWP;

print "NetID\tName\tDept\tPhone\n";
open in, "netids.txt";
while($txt = <in>)
{
        chomp $txt;

        $ua=LWP::UserAgent->new;
        $ua->agent("MyApp/0.1 ");
        $adr = "http://www.cornell.edu/search/people.cfm?netid=" . $txt;
        $req = HTTP::Request->new(GET => $adr);
        $req->header(Accept => "text/html, */*;q=0.1");
        my $res = $ua->request($req);
        if ($res->is_success)
        {
          my $dept = get_info($res->content,"<th>SUPERVISORY ORGANIZATION:</th>");
                my $phone = get_info($res->content,"<th>PRIMARY:</th>");
                my $name  = get_name($res->content);
                print "$txt\t$name\t$dept\t$phone\n";
        }
        else
        {
                print "Error reading website\n";
                exit;
        }
         sleep(1); #Important! Do not overload the server
}
close(in);
```

script5.pl (2)

```
sub get_info
{
        my ($page, $tag) = @_;

        my $n = index($page, $tag);
        if($n == -1)
        {
                return "unknown";
        }
        else
        {
                my $txt1 = substr($page, $n);
                my @tab = split /\n/, $txt1;
                my $info = substr($tab[1], index($tab[1],"<td>")+4,
                          index($tab[1],"</td>") - index($tab[1],"<td>") - 4 );
                $info =~ s/\s+$//;
                if($info ne "")
                {
                     if(index($info,"</a>") != -1){
                         $info = substr($info, index($info,">") + 1,
                         index($info,"</a>") - index($info,">") - 1 );}
                      return $info;
                }
                else
                {
                        return "unknown";
                }
        }
}
```

script5.pl (1)

```
sub get_name
{
        my ($page) = @_;

        my $n = index($page, "<h3 class=");
        if($n == -1)
        {
                return "unknown";
        }
        else
        {
                my $txt1 = substr($page, $n);
                $txt1 = substr($txt1,index($txt1,">") + 1,
                    index($txt1,"</h3>") - index($txt1,">") - 1);
                return $txt1;
        }
}
```

Exercises

1. Write a script that checks for new versions of a set of software titles listed in a file. Choose 5 programs from BioHPC Lab software list (http://cbsu.tc.cornell.edu//lab/labsoftware.aspx) and implement them. Check for new versions on the original software websites, NOT BioHPC website.

   HINT: You can use script4.pl as the starting point

   HINT: You will need to put URL and tag delimiters info for each program in the input file.