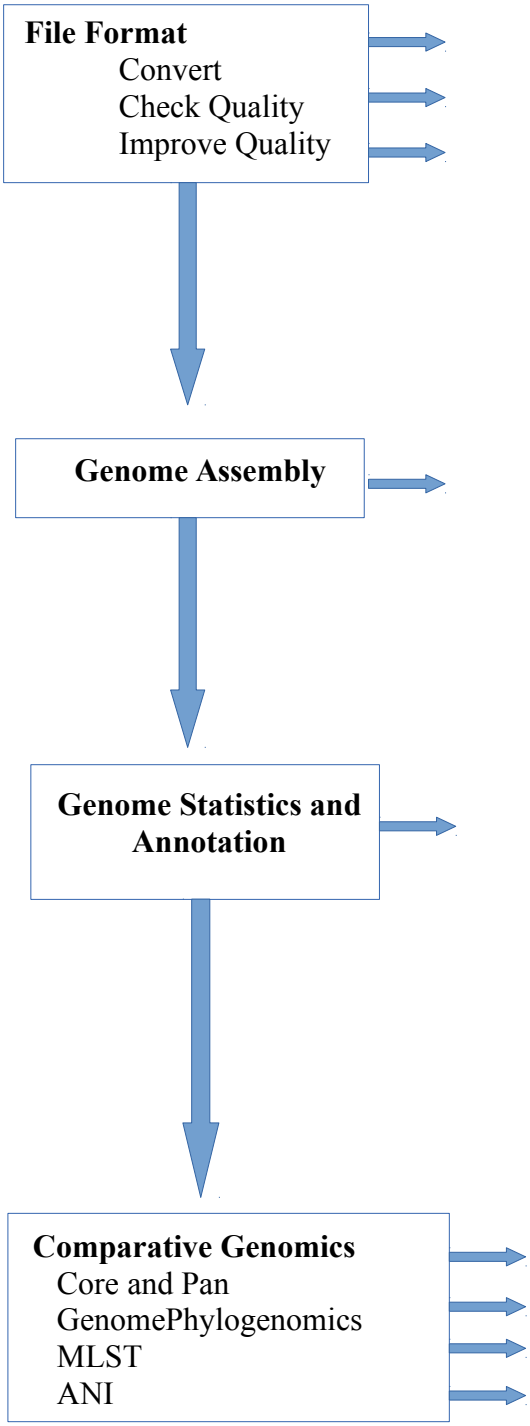


Summary of Steps (LHS) and Software (RHS) Used in *Neisseria memingitidis* Genomics. All analysis was performed using a Linux Environment (Ubuntu)

I. Assessing & Manipulating Read Data & Quality:

1) FASTQC (http://www.bioinformatics.bbsrc.ac.uk/projects/fastqc/). Sequence data is never of equal quality for all reads. You will want to trim/filter some of your reads to enrich for high-quality data. FASTQC is a multi-platform application which will aid in your visualization of the quality of your data.

2) GALAXY and the FASTX Toolkit (http://main.g2.bx.psu.edu/ and http://hannonlab.cshl.edu/fastx_toolkit/). FASTQC should tell you things like how is the sequence quality at the 5' end of my reads. Frequently it will be low and you may want to exclude this sequence from subsequent analysis. Using tools available in GALAXY and the FASTX Toolkit you should be able to filter and trim your data to your hearts content. Both packages are well documented. GALAXY has more functionality than a swiss army knife wielding ninja and I recommend you take a look at the entire package as well as some of the web-tutorials. It offers an ideal platform for an entry level bioinformatician looking to do some work in genomics. A short tutorial on how to do QC on sequence data can be found here: http://www.molecularevolution.org/resources/activities/QC_of_NGS_data_activity along with a variety of other tools/tutorials that might be of interest to you.

3) Another package you may want to consider in order to QC, filter and trim your data is PRINSEQ: http://edwards.sdsu.edu/prinseq_beta/. I recommended it in another thread to someone else new to sequence analysis and it seemed to be a hit.

NOTE:
1. I would suggest that you try to get the raw data either as SFF file or as FASTQ file. From the SFF file, you can extract the sequence and quality data and convert it into FASTQ format using e.g. PRINSEQ (or upload the FASTA and QUAL files directly to its web interface).

II. Assembly:

4) Bowtie/VELVET/NEWBLER/AbYSS/MIRA: These programs should help you to assemble your filtered/trimmed data into something a bit more reasonable to handle. There are a large number of assemblers and mappers that can help you do this task. Assembling next-gen data is an under appreciated and challenging aspect of genomics to many biologists. Each data set has it's unique qualities making it no so easy to cookie cut. As I recommended above, I would use NEWBLER if you have access to it. If not, any of the ones listed here should get you started. Bowtie is easy to use and very fast. If you have a high-quality reference genome this may be the way to go. VELVET takes a lot of memory, but may be an option if you do not have a good reference genome. AbYSS and MIRA can work if you do or do not have a reference genome. Bastien Chevreux has done an excellent job at writing and documenting MIRA. It is worth going through some of his exercises just for the learning experience alone.

Analysis of Read data

Before assembling *de novo*, reads should be assessed for quality.

I. File Formats:

- Ion Torrent reads are available in bam and sff formats; bam formats are provided as a routine but you may need to request the core service for the sff format.
- Sff files can be converted to FASTQ format
  - Sff_extract or sffinfo
- FASTQ files can be converted to FASTA and QUAL files

II. Read quality can be assessed from:

1. sff formats by using
   - online PRINSEQ (http://edwards.sdsu.edu/prinseq_beta/) or
   - offline version installed on your computer.
2. fastQ formats by using
   - Fastqc - http://www.bioinformatics.bbsrc.ac.uk/projects/fastqc/ (see Examples, Galaxy below)
3. Interpretation of FastQC output files is at the URL
   - http://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/

III. Examples:

1. Working on Galaxy with fastqc:

- Click "Get Data" (LHS Panel), click "Upload File" and upload your Ion Torrent sequence data (fastq).
- Click "NGS: QC and manipulation", click "FASTQ Groomer" (with Sanger & Illumina 1.8+) and Advanced Options output FASTA Q score type Sanger (recommended) to change phred quality scores
- Click "NGS: QC and manipulation", click "FastQC"

2. Working with fastqc with command line:

$ fastqc (installed /usr/local/bin/fastqc on bharat's desktop server)

A GUI will load. Follow the options from the GUI

Analysing Assembly Metrics

1. A script for converting the 454NewblerMetrics.txt file to a tab-separated file:
   Software: newblermetrics1.2.pl (ver 1.2, updated by Lex Nederbragt(@bio.uio.no in September 2012)

   Tested on: newbler v 2.3, 2.5.3 and 2.6 (not extensively) on both shotgun, shotgun + paired end and transcriptome assemblies

   Does not work yet on a file from a mapping project (gsMapper, runMaping)

   Local installation:
   Dell Laptop: /usr/bin/newblermetrics1.2.pl
   Desktop: TBA

   Usage:
   newblermetrics1.2.pl newblermetrics_file.txt > output_file_name.txt

2. mira

Readme for sff_extract (revised 1 June 2013)

1. Download ion torrent datafile (sff)from server. sff can also be extracted from a Bam file which is now a standard for Ion Torrent.

2. sff_extract –-version (check - should be version 0.3.0 (older versions do not work). The following may need sudo - I edited a copy (/usr/bin) by replacing the entire content with a copy of the new version using gedit. Alternatively download into /usr/bin and Chmod 755

3. sff_extract (see sffinfo instructions)
   -Q (option -Q gave me an error but file not required)
   -s BC01_Y50_in.iontor.fastq
   -x BC01_Y50_traceinfo_in.iontor.xml
   NOTE: iontor indicates datatype & is required in file naming conventions

4. Using mira
   - Create 3 directories: assembly, data and origdata
   - mv the xml datafile and fastq datafiles to data and the original sff data file to origdata
   - mira command

5. mira command (all in one line line; note spacing between instructions)

mira --project=BC03_run3 --job=denovo,genome,accurate,iontor -GE:not=8 >&log_assembly.txt
(Use tail loga ssembly.tx to view the last few lines to check how the assembly is going)

Explanation: create a directory for the assembly, copy the sequences into it (to make things easier, name the file directly in a format suitable for mira to load it automatically); also copy quality values for the sequences into the same directory if appropriate. Start mira with options:

- the project is named 'bchoc' and hence, input and output files will have this as prefix
- the data is in a FASTA formatted file (if appropriate)
- the data should be assembled *de-novo* as a *genome* at an assembly quality level of *accurate* (draft) and that the reads being assembled were generated using Ion Torrent technology (Sanger, 454, PacBio)
- -GE:not=8 to use 8 threads (maximum in Dell laptop)

Filtering Assemblies

Remove shorter sequences and sequence duplicates should be removed as well. Duplicates do not add new information, but add to the size of the database. In order to reduce the amount of false positive alignments, sequences with too many ambiguous bases should be removed.
The sequences can be easily filtered with programs such as PRINSEQ (http://prinseq.sourceforge.net). The following command will additionally rename the sequence identifiers to ensure unique identifiers in the whole data set and delete the input file (rm command).

```
$ perl prinseq-lite.pl -log -verbose -fasta hs_ref_GRCh37_p2_split.fa -min_len 200 -ns_max_p 10 -derep 12345 -out_good hs_ref_
```

6. **Dell laptop specs**

Checked the processor in “System” and googled the processor to find detailed spec:

Processor Number:	i 7-820QM
# of Cores	4
# of Threads	8

Analysed using 1, 4 and 8 threads:

Start BC01_Y50 with 1 thread

started: Sat Jun 1 19:58:45 2013

completed: Sat Jun 1 21:10:52 2013

Start BC02_86 with 4 processors

started: Sun Jun 2 18:19:19

completed: Sun Jun 2 20:28:12

Sff files

Each sff file starts with a ‘common header’:

```
Common Header:
Magic Number: 0x2E736666
Version: 0001
Index Offset: 110544
Index Length: 3173
# of Reads: 35
Header Length: 840
Key Length: 4
# of Flows: 800
Flowgram Code: 1
Flow Chars:
TACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTAC
Key Sequence: TCAG
```

Magic number: identical for all sff files (0x2E736666, the newbler manual explains that this is ‘the uint32_t encoding of the string „.sff“ ‘)

- Version: also identical for all sff files (0001)
- Index offset and length: has to do with the index of the binary sff file (points to the location of the index in the file)
- # of reads: stored in the sff file
- Header length: looks like it is 440 for GS FLX reads, 840 for GS FLX Titanium reads
- Key length: the length (in bases) of the key sequence that each read starts with, so far always 4
- # of Flows: each flow consists of a base that is flowed over the plate; for GS20, there were 168 flows (42 cycles of all four nucleotides), 400 for GS FLX (100 cycles) and 800 for Titanium (200 cycles)
- Flowgram code: kind of the version of coding the flowgrams (signal strengths); so far, ’1′ for all sff files
- Flow Chars: a string consisting of’ # of flow’ characters (168, 400 or 800) of the bases in flow order (‘TACG’ up to now)

Each read has the following structure

```
>F7K88GK01BMPI0

Run Prefix: R_2009_12_18_15_27_42_

Region #: 1

XY Location: 0551_2346

Run Name: R_2009_12_18_15_27_42_FLX########_Administrator_yourrunname

Analysis Name: D_2009_12_19_01_11_43_XX_fullProcessing

Full Path: /data/R_2009_12_18_15_27_42_FLX########_Administrator_yourrunname/D_2009_12_19_01_11_43_XX_fullProcessing/

Read Header Len: 32

Name Length: 14

# of Bases: 500

Clip Qual Left: 15

Clip Qual Right: 490

Clip Adap Left: 0

Clip Adap Right: 0

Flowgram: 1.03 0.00 1.01 0.02 0.00 0.96 0.00 1.00 0.00 1.04 0.00 0.00 0.97 0.00 0.96 0.02 0.00 1.04 0.01 1.04 0.00 0.97 0.96 0.02 0.00 1.00

0.95 1.04 0.00 0.00 2.04 0.02 0.03 1.05 0.99 0.01 2.84 0.03 0.05 0.97 0.12 0.00 1.01 0.05 0.97 0.01 2.89 0.04 0.09 1.05 0.15 0.00 2.84 0.06

1.00 0.01 0.13 1.01 0.09 0.98 0.01 0.05 1.01 0.06 0.00 1.04 3.72 0.03 0.00 0.96 1.97 0.04 0.01 1.97 0.12 0.98 0.02 0.08 0.95 0.12 ...

Flow Indexes: 1 3 6 8 10 13 15 18 20 22 23 26 27 28 31 31 34 35 37 37 37 40 43 45 47 47 47 50 53 53 53 55 58 60 63 66 67 67 67 67 70 71 71 74

74 76 79 82 83 86 86 88 88 91 93 96 97 99 102 105 ...

Bases:

tcagatcagacacgCCACTTTGCTCCCATTTCAGCACCCCACCAAGCACAAGGCTGTCATCCCAATTGGACGGACAGATATGAGGTTAGCATTGGAAACCAATTCAGTCCCTAATTATTCACGACTGAACCCAGCGACAATTGGACAT

Quality Scores: 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 38 38 38 40 40 40 39 39 39 40 34 34 34 40 40 40 40 39 26 26 26 26 40 40

40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 ...
```

- Key Sequence: the first four bases of reads are either added during library preparation (they are the last bases of the ‘A’ adaptor) or they are a part of the control beads. For example, Titanium sample beads have key sequence TACG (default library protocol) or GACT (rapid library protocol), control beads have CATG or ATGC. Control reads never make it into sff files… Each read has the following structure
- >F7K88GK01BMPI0: this is the read name, or “universal accession number.” ‘F7K88G’ encodes the timestamp of the run, ‘K’ is a random character, ’01′ indicates the region (lane) number on the plate, ‘BMPI0′ encodes the x,y location of the read on the plate.
- Run prefix: A run folder starts with ‘R’ and the time the run started: R_yyyy_mm_dd_hh_min_
- Region #: the region (lane) on the plate the read originated from
- XY location: the location of the read on the plate
- Run name: R_yyyy_mm_dd_hh_min_sec_machineName_userName_yourrunname
- Analysis name: after a run, a subfolder is made with the image/basecalling analysis results, the foldername starts with ‘D’ and the time the analysis started: D_yyyy_mm_dd_hh_min_sec_machineName_analysisType
- Full path: of the analysis results that the sff file originated from (on the GS FLX instrument: /data/R_…/D_…)
- Read header len: 32 for all files as far as I can tell
- Name length: the length of the read name (14), see above
- # of bases: the total number of bases called for the read (before clipping)
- Clip qual left: the position of the first base to be included after clipping. This is usually 5 because of the first four bases that are the key sequence. In this example, the read had an 10 base MID sequence; the example sff file is the result of splitting the original sff file, during splitting the MID sequence is ‘removed’, i.e. the clipping point is set beyond the MID end.
- Clip qual right: position of the last base before the (quality) clipping.
- Clip adap left and right: I actually wouldn’t know what these represent, but perhaps under certain

circumstances, adaptors can be ‘removed’ this way.

- Flowgram: for each flow, the normalized signal strength, or actually, the homopolymer length estimate, as a floating point integer with two digits to the right of the point.
- Flow Indexes: the flows actually used for basecalling (excluding flows considered to be ’0′, i.e. no signal.
- Bases: the determined DNA sequence. Lower case bases are before and after the clipping point
- Quality scores: the phred quality scores cores corresponding to the bases

How to ‘translate’ the flowgram values into bases? The start of the example flowgram has these signals:

```
1.03 0.00 1.01 0.02 0.00 0.96 0.00 1.00 0.00 1.04 0.00 0.00 0.97 0.00 0.96 0.02 0.00 1.04
0.01 1.04 0.00 0.97 0.96 0.02 0.00 1.00 0.95 1.04 0.00 0.00 2.04 0.02 0.03 1.05 0.99 0.01
2.84 0.03
```

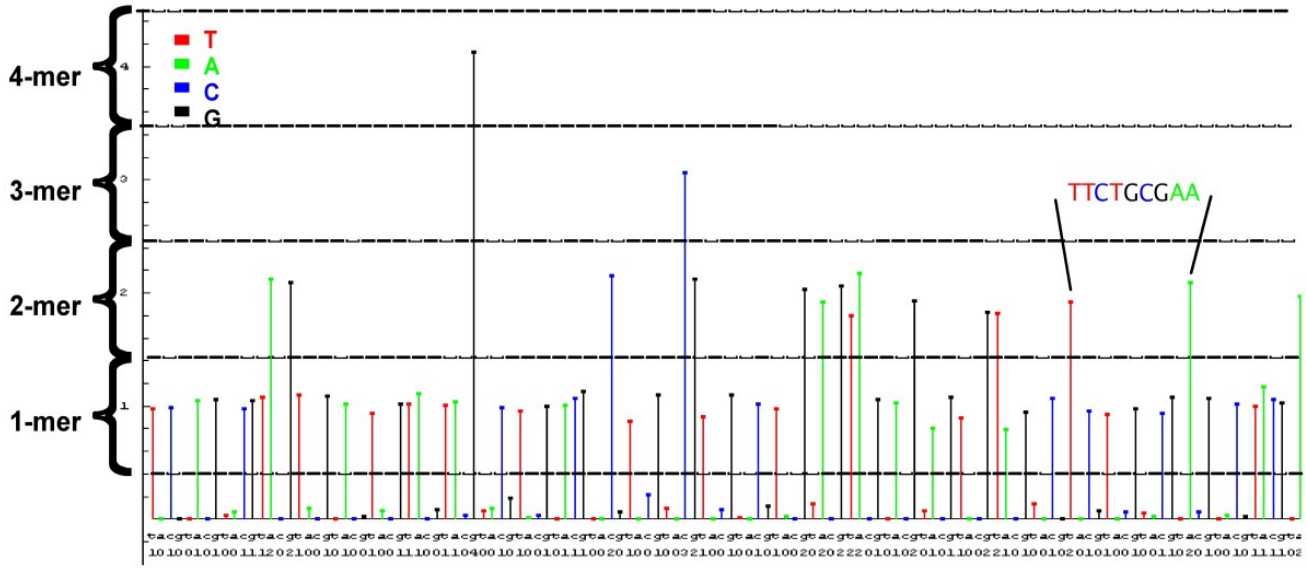
Rounding of the numbers:

```
1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 1 0 0 1 1 1 0 0 2 0 0 1 1 0 3 0
```

With the flow order TACG, this translates into

1 T’s; 0 A’s; 1 C’s; 0 G’s; 0 T’s; 1 A’s etc, or TCAGATCAGACACGCCACTTT

The figure is a graphic representation of the flowgram, with another example of ‘reading’ the sequence from it. Note that for some signals, the intensity is such that it is hard to determine whether for example there are two or three bases at that position. This inherent property of pyrosequencing leads to the well-known homopolymer (over- and undercall) errors.



“# of Flows: each flow consists of a base that is flowed over the plate; for GS20, there were 168 flows (42 cycles of all four nucleotides), 400 for GS FLX (100 cycles) and 800 for Titanium (200 cycles)” Do you know the number of flows values for the “GS Junior” and for the new “GS FLX+”?

Since the GS Junior is running GS FLX Titanium chemistry, the number of flows is 800. GS FLX+ is double that, 1600 flows (400 cycles).

Sff_info

Sff files are binary files, meaning that they can not be accessed by regular text-based tools. 454 has its own scripts to manipulate sffiles and extract information from them (sfffile, sffinfo), but other programs/scripts can also be used to extract information from them. Example programs are sff_extract, flower, sff2fasta, or use the biopython parser, nothing for bioperl yet (I have not tested any of these – use at your own discretion…).

When one uses 454′s sffinfo command on an sff file without parameters, all information contained in the file is reported in text format.

Note: Use sffinfo with options but stick to the naming convention as described in 3, above.

```
sffinfo -s file.sff > file.fasta
sffinfo -q file.sff > file.qual
sffinfo -m file.sff > file.manifest.xml
```

Here is the usage:

```
Usage: sffinfo [options...] [- | sfffile] [accno...]
Options:
-s or -seq Output just the sequences
-q or -qual Output just the quality scores
-f or -flow Output just the flowgrams
-t or -tab Output the seq/qual/flow as tab-delimited lines
-n or -notrim Output the untrimmed sequence or quality scores
-m or -mft Output the manifest text
```

Newbler Metrics

The following will extract the main metrics data from 454NewblerMetrics newblermetrics1.2.pl. This perl script is in /usr/bin

Mapping

Scripture and Cufflinks should be on your list for reference-guided / mapping assembly.

TMAP – TS from Ion Torrent:
cd /opt
sudo git clone https://github.com/iontorrent/TS.git

Need more information

Installing Newbler 2.8 – Ubuntu 12.04
Installing 2.9 – Ubuntu 12.04 (15 Aug 2013)

1. Fill in the application form and Roche will get back to you with instructions for download.

To do assembly or mapping using Newbler you will need to download item1.1; 1.2, 1.3 and 1.4 are pdf files on howto

1.1 DataAnalysis_2.8_All.tgz = Newbler assembler, mapper, Amplicon variant analyser, and sffinfo/sfffile software. Runs on Linux ONLY.

1.2 Manual-v2.8_Overview_Oct2012.pdf = Overview of roche software, including sffinfo, sfffile

1.3 Manual-v2.8_PartC_Assembly_and_Mapping_Oct2012.pdf = PDF file of Manual for Newbler Assembly and Mapping programs.

1.4 Manual-v2.8_PartD_AmpliconVariantAnalyzer_Oct2012.pdf = PDF file of manual for the Amplican Variant Analyser

2. Installing Newbler 2.8:
There are 3 work arounds for installing Newbler 2.8 (BioLinux 7). Out of these 3, option number 2.1 worked for my Dell Laptop.

2.1 Local install and once installed move directories / files to System wide:
This requires that you install software locally and not system wide, that is not as root. Once the installation is complete, you can move it to a system wide location eg /opt/454.

Change dash to bash:

```
sudo rm /bin/sh
sudo ln -s bash /bin/sh
```

Uncompress the downloaded Newbler file:

```
tar -zxvf DataAnalysis_2.8_All_20120731_2108.tgz
```

cd to the uncompressed directory:

```
DataAnalysis_2.8_All/
```

Use the install script to install:

```
./setup.sh
```

Newbler will be installed in your local home directory:

```
eg /home/yourname/454
```

Copying local to system wide:

```
cd to /opt
mkdir 454
cp all directories from local to /opt/454
  eg cp -a /source/. /dest/;  -a option is an improved recursive option, that preserves all file
      attributes, and also preserves symlinks.
make all files executable by sudo chmod +x * (for all files)
```

Remember to change bash to dash again:

```
sudo rm /bin/sh
sudo ln -s dash /bin/sh
```

Edit .zshrc:

gedit or vi .zshrc (in your home) to include the path for newbler programs:

export PATH=$PATH:/opt/454/bin

(OR export PATH=$PATH:/yourhome/454/bin)

2.2 Rpm to deb and than install:

- Install ia32-libs and alien using Synaptic Packet Manager
- Make debian files by: sudo alien *.rpm –scripts
- Install Debian files by sudo dpkg -i *.deb
- You'll have to manually create the gsAmplicons menu entry. The icon and executable will be in **/opt/454.**

For some reason jre rpm does not get packaged as debian and hence is not installed.

2.3 Using Nixinstaller:

By default, the software will be installed in either:

/opt/454 – if you select root install (ie. Used by all users):

home/yourhome/454- if you install as non-root user for yourself in your home directory.

(a) In Ubuntu, the "Nixstaller" installer used in the installation package may probably stop with the following error:

Cannot execute command: type rocks2>&1

A work-around is to create a link so rocks always returns true:

ln -s /bin/true ~/bin/rocks

sudo ./setup.sh (sudo is only required for system wide install but not for your own home)

(b) For the Newbler assembler itself there are both 64 bit and 32-bit versions in the package, but the GUI is only 32-bit, so the installation may complain that the following libraries were not found:

libraries not found : zlib.i386, libXi.i386, libXtst.i386, libXaw.i386

For ubuntu install the 32-bit libraries:

sudo apt-get install ia32-libs (or use the Synaptic Package Manager if root)

For Fedora, CentOS, or Readhat linux, use:

sudo yum install glibc.i686

(http://stackoverflow.com/questions/8328250/centos-64-bit-bad-elf-interpreter)

3. Running Newbler:

To run the Newbler assembler from the command-line, typically use:

(a) For Genome data:

runAssembly -urt -het -o assembly1 reads.sff where:

"-urt" means "use read tips" to extend contigs across low coverage regions,

"-het" is for reads from heterogenous sample -eg. from several individuals or diploid.

(b) For Transcriptome data:

runAssembly -urt -cdna -vt AdaptersToTrimFromReads.fna -o assembly1 reads.sff where

"-urt" means "use read tips" to extend contigs across low coverage regions,

"-cdna" indicates reads are cDNA, so consider alternative splicing, and large reange of coverage.

"-vt .." to trim adapters, such as MINT or SMART adapters used to synthesis the cDNA, from the

reads before assembly.

Several additional options can be used for genome and transcriptome data, such as:

"-vs ScreeningFile.fna" for screening out contaminants.

"-p matepaireads.sff" for mate-pair reads.

See the Part-C user manual for more details.

Installing mira 3.9.1.7

1. Uninstall mira 3.4 from Bio-Linux (use Synaptic Manager, search for mira and deselect to uninstall) – this is an old version.

2. Download: mira_3.9.17_linux-gnu_x86_64_static.tar.bz2 (from

3 . Change to execute permission: sudo chmod a+x

4. mira_3.9.17_linux-gnu_x86_64_static.tar.bz2

5. Move directory: sudo mv mira_3.9.17_linux-gnu_x86_64_static.tar.bz2 /opt

6. Change owner: sudo chown root mira_3.9.17_linux-gnu_x86_64_static.tar.bz2e

7. Unbzip: sudo tar xjfv mira_3.9.17_linux-gnu_x86_64_static.tar.bz2

8. gedit .zshrc, add: exportPATH=$PATH:/opt/mira_3.9.17_linux-gnu_x86_64_static/bin

Note:
Running mira version 3.9.18 from 18 July 2013.

Too much coverage - Mira

```
On Jun 16, 2012, at 17:25 , Shankar Manoharan wrote:
> Thank you Bastien. Is there any way I can throw away low quality/low read
> length parts of the sff file and proceed with the rest of the data ?

You can do so by using "-c" in sff_extract. However, I do not recommend doing
this as MIRA uses low-quality parts of Ion reads to look for irregularities and
eventually clip the read further.

In case you are trying to save as much memory as possible, there's one mean
trick: tell MIRA to stop after data preprocessing ("-AS:nop=0"). Then convert
the checkpoint file of MIRA to clipped FASTQ ("convert_project -f maf -t fastq
-C readpool.maf yournewinput"). The resulting "newinput.fastq" will represent
what MIRA thinks is valid sequence with everything else removed.

Start a regular assembly with that, remember to switch off all clipping for
that as it has been already done: "--noclipping -CL:pec=no"

> On Jun 16, 2012, at 15:16 , Shankar Manoharan wrote:

>
> *      This is regarding assembly of Ion torrent data with MIRA. We
> sequenced a ~3 Mb genome on an Ion 316 chip which generated around 730 Mb
> of data to our surprise. We didn't expect SO much data out of a 316 chip
> (their claim is >100 Mb). Ion torrent's torrent server which uses MIRA as
> the de novo assembler failed with the TmAlloc error). I was asked by the
> bioinformaticians at Ion torrent support to run MIRA via command line on
> the torrent server (RAM: 48 Gb, Disk: 12 Tb) which ran out of memory
> (Displays a TmAlloc error and MIRA ends abruptly before creation of contig
> fasta files. I can however see the assembly statistics in the info folder).
> Assembly statistics show ~15000 contigs and a total consensus of 9.8 Mb.
> I'm completely flabbergasted first at the amount of data generated, and now
```

> at number of contigs generated :D. Can someone help with this ? Any similar
> experiences ? The next thing I am planning to do is to extract reads of
> good quality and reads longer than 200 bp (Average read length is 245 bp
> using their 200 bp chemistry) and attempt assembling the down sized data.
> Any inputs and suggestions would be greatly appreciated.*
>
>
> I think you might want to read "A word or two on coverage ..." in the MIRA
> docs:
>
>
http://mira-assembler.sourceforge.net/docs/DefinitiveGuideToMIRA.html#sect_seqadv_a_word_or_two_on_coverage
>
> Especially the warning regarding too much coverage at the end of the
> section.
>
> Hope that helps,
>   Bastien

Hi Bastien,

I was trying to do a mapping assembly using Ion torrent data. Little was mentioned in the manual for ion torrent data but found the following from 'Mira_talk' as you suggested:

```
mira -project=readData -job=mapping,genome.accurate,iontor
  -SB:bsn=ThisIsTheNameOfTheStrainServingAsBackboneAlsoCalledReference
  IONTOR_SETTINGS -LR:dsn=MyReadsComeFromStrainXYZAndThatsWhyIPutItHere
  >&log_assembly.txt
```

Here is what I did:

1)downloaded a reference genome from NCBI: NC_012967.gbk and renamed it to A15_backbone_in.gbf

2) ion torrent fastq data: A15.fastq and renamed to A15_in.iontor.fastq

Then I ran:

```
--project=A15mapping --job=mapping,genome,accurate,iontor
SB:bsn=A15_backbone_in.gbf IONTOR_SETTINGS -LR:dsn=A15_in.iontor.fastq
>&log_assembly.txt
```

It didn't work.

Then I used A15_in.iontor.fastq generated from sff_extract (together I put A15_traceinfo_in.iontor.xml into the same dir), but it still didn't work.

Can you please help or guide me through where went wrong? Or something I didn't understand or didn't include?

Thanks a lot

Austen

Manifest file from mira documents

 # A manifest file can contain comment lines, these start with the #-character

```
# First part of a manifest: defining some basic things

# In this example, we just give a name to the assembly
#  and tell MIRA it should assemble a genome de-novo in accurate mode
# As special parameter, we want to use 4 threads in parallel (where possible)

project = dh10b
job = genome,denovo,accurate
parameters = -GE:not=4

# The second part defines the sequencing data MIRA should load and assemble
# The data is logically divided into "readgroups", for more information
#  please consult the MIRA manual, chapter "Reference"

readgroup = SomeUnpairedIonTorrentReadsIGotFromTheLab
technology = iontor
data = ../../data/dh10b*

# note the wildcard "dh10b*" part in the 'data' line above:
#  if you followed the walkthrough and have the FASTQ and XML file,
#  this will automatically load both files (which is what we want)
```

Manifest files – Example 1

Ion Torrent Assembly (with illumina):

```
project = bac_ill_ion
job = genome, denovo, accurate
parameters = COMMON_SETTINGS -GE:not=30 \
        -MI:sonfs=no \
        --noclipping \
        IONTOR_SETTINGS -AS:mrpc=50\
        # SOLEXA_SETTINGS -AS:mrpc=100

readgroup = torrent_single_end
technology = iontor
data = iontorrent.fastq

readgroup = illumina_paired_end
data = illumina_1.fastq illumina_2.fastq
technology = solexa
templatesize = 200 600
segmentplacement = FR
segmentnaming = solexa
```

Ion Torrent with mapping:

```
mira -project=readData -job=mapping,genome.accurate,iontor
  -SB:bsn=ThisIsTheNameOfTheStrainServingAsBackboneAlsoCalledReference
  IONTOR_SETTINGS -LR:dsn=MyReadsComeFromStrainXYZAndThatsWhyIPutItHere
  >&log_assembly.txt
```

NOTE: Setting quality and length:

How would one set a minimum read length of 80 and a base default quality of 10 for 454 reads, but for Solexa reads a minimum read length of 30 with a base default quality of 15?

The answer:

```
mira -job=denovo,genome,draft,454,solexa  -fasta
  454_SETTINGS -AS:mrl=80:bdq=10 SOLEXA_SETTINGS -AS:mrl=30:bdq=15
```

Manifest file – example 2 – (tsucheta@gmail.com)

```
project = Mastigocladus
job = genome,denovo,accurate

readgroup = fragment reads
data = /home/mastigocladus_filtered_reads_in_iontor.fastq
technology = iontor

parameters = COMMON_SETTINGS -GE:not=4 COMMON_SETTINGS -AS:nop=2
parameters = IONTOR_SETTINGS -AS:mrpc=20 -AS:mrl=60:bdq=15
parameters = IONTOR_SETTINGS -CL:msvssfc=10 -CL:msvssec=3 -CL:cpat=yes
~
```

Manifest file – example 2 – (tsucheta@gmail.com)

The following is included for mapping:

readgroup
as_reference
data = something.gff3

Manifest file – Mira

The manifest file is divided into 2 parts.

Part 1: Part 1 is comprised of three entries, namely:

1. **project**
2. **job**
3. **parameters**

1. project = tells the assembler the name you wish to give the assembly. DONOT use slashes or back slashes eg MyFirstAssembly

2. job = tells the assembler what kind of data it should expect and how it should assemble it. The choice is made in 3 steps:

- denovo OR mapping -- leaving this blank defaults to denovo.
- genome (large fragments) OR EST (small fragments) -- leaving this blank will default to genome.
- draft (quick and dirty) OR accurate -- leaving this blank will default to accurate

NOTE: For *denovo* assemblies of genomes, the above switches are optimised for decent coverage eg x7 for Sanger, x>25 for 454 FLX / Titanium, x25 for 454 GS20, x>30 for Solexa.

Once you're done concatenate everything with commas and you're completed the **"job"** Entry. For example:

- job=mapping,genome,draft, will give you a mapping assembly of a genome in draft quality.
- job=genome,denovo,accurate (assemble genome denovo and in accurate mode)

3. parameters is selected from any of the 150+ selectable switches. For example:

- parameters = -GE:not=4 (Use 4 threads in parallel, if possible)

Part 2. Part 2 of the manifest file tells MIRA which files it needs to load, (i) which sequencing technology generated the data eg 454, Illumina, Ion Torrent, PacBio, (ii) whether there are DNA template constraints it can use during the assembly process and (iii) a couple of other things.

1. readgroup: Reads from the same technology and same library preparation are pulled together in a read group:(= group name) is the keyword which tells MIRA that you are going to define a new read group. You can optionally name that group. For example:

- readgroup = SomeUNnpairedIlluminaReadsIGotFromTheLab
- readgroup = SomeUNnpaired454ReadsIGotFromTheLab
- readgroup = torrent_single_e
- readgroup = SomePairedEndIlluminaReadsIGotFromTheLab

2. data: Sets the location and file of the data [filepath [filepath ...] and the type of data format from where the sequences should be loaded.

2.1 A file path can contain just the name of one (or several) files or it can contain the *path*, i.e., the directory (absolute or relative) including the file name.

2.2 MIRA automatically recognises what type the sequence data is by looking at the postfix of files. Currently allowed file types are:

- .fasta for sequences formatted in FASTA format and an additional .fasta.qual file which contains quality data. If the file with quality data is missing, this is interpreted as error and MIRA will abort.

- .fna, also for sequences formatted in FASTA format. The difference to .fasta lies in the way MIRA treats a missing quality file (called .fna.qual): it does not see that as critical error and continues.
- .fastq for files in FASTQ format
- .fofnexp for a *file of EXP filenames* which point to file in the Staden EXP format.
- .gff3 or .gff for files in GFF3 format. Note that MIRA will load all sequences and annotations contained in this file.
- .gbk, .gbf, .gbff or .gb for files formatted in GenBank format. Note that the MIRA GenBank loader does not understand intron/exon or other multiple-locus structures in this format, use GFF3 instead!
- .caf for files in the CAF format (from Sanger Centre)
- .maf for files in the MIRA MAF format
- .xml, .ssaha2 and .smalt for ancillary data in NCBI TRACEINFO, SSAHA2 or SMALT format respectively.

Notes:
Multiple 'data' lines and multiple entries per line (even different formats) are allowed. For example:
data = file1.fastq file2.fastq file3.fasta file4.gbk
data = myscreenings.smalt

You can also use wildcards and/or directory names. For example:
loading all file types MIRA understands from a given directory mydir:

```
data = mydir
```

OR
loading all files starting with mydata and ending with fastq.

```
data = mydata*fastq
```

OR
Loading all files in directory mydir starting with mydata and ending with fastq:

```
data = mydir/mydata*fastq
```

OR
loading all FASTQ files in all directories starting with mydir:

```
data = mydir*/*fastq
```

Note

Giving a directory like in mydir is equivalent to mydir/* (saying: give me all files in the firectory mydir), however the first version should be preferred when the directory contains thousands of files.

Note

GenBank and GFF3 files may or may not contain embedded sequences. If annotations are present in these files for which no sequence is present in the same file, MIRA will look for reads of the same name which it already loaded in this or previously defined read groups and add the annotations there.
As security measure, annotations in GenBank and GFF3 files for which absolutely no sequence or read has been defined are treated as error.

- **default_qual**= *quality_value* is meant to be used as default fallback quality value for sequences where the data files given above do not contain quality values. E.g., GFF3 or GenBank formats,

eventually also FASTA files where quality data files is missing.

- **technology**= *technology* which names the technology with which the sequences were produced.
- Allowed technologies are: *sanger, 454, solexa, iontor, pcbiolq, pcbiohq, text*.

The *text* technology is not a technology per se, but should be used for sequences which are not coming from sequencing machines like, e.g., database entries, consensus sequences, artificial reads (which do not comply to normal behaviour of 'normal' sequencing data), etc.pp

- **as_reference** This keyword indicates to MIRA that the sequences in this readgroup should not be assembled, but should be used as reference backbone for a mapping assembly. That is, sequencing reads are then placed/mapped onto these reference reads.
- **templatesize** = *min_size max_size [infoonly|exclusion_criterion]*. Defines the minimum and maximum size of "good" DNA templates in the library prep for this read group.

If the term *infoonly* is present, then MIRA will pass the information on template sizes in result files, but will not use it for any decision making during de-novo assembly or mapping assembly. The term *exclusion_criterion* makes MIRA use the information for decision making.
If *infoonly* or *exclusion_criterion* are missing, then MIRA assumes *exclusion_criterion* for denovo assemblies and *infoonly* for mapping assemblies.

Note: The *templatesize* line in the manifest file replaces the parameters -GE:uti:tismin:tismax of earlier versions of MIRA (3.4.x and below)

For *mapping* assemblies with MIRA, you usually will want to use *infoonly* as else - in case of genome re-arrangements, larger deleteions or insertions - MIRA would probably reject one read of every read pair (as it would not be at the expected distance and/or orientation) and you would not be able to simply find the re-arrangement in downstream analysis.
For *de-novo* assemblies however you *should not* use *infoonly* except in very rare cases where you know what to do. at you do.

Some examples:
readgroup = SomeUNnpaired454ReadsIGotFromTheLab
data = This can contain one or more files (eg TCMAXXXX.fastq TCMBXXXXXXX.fastq) OR can contain the path to the files(eg /bharat/Downloads/Desktop/genome_new/TCMAXXXXX.fastq)
technology = 454

readgroup = SomePairedEndIlluminaReadsIGotFromTheLab
data = TCMCXXXX.fastq TCMDXXXXXXX.fastq
/bharat/Downloads/Desktop/genome_new/TCMYXXXXX.fastq
technology = Solexa

readgroup = SomeUNnpairedIlluminaReadsIGotFromTheLab
data = TCMCXXXX.fastq TCMDXXXXXXX.fastq
technology = Solexa

r**eadgroup** = torrent_single_end
data = iontorrent1.fastq iontorrent2.fastq
technology = iontor

project = bac_ill_ion

job = genome, denovo, accurate
parameters = COMMON_SETTINGS -GE:not=30 \
-MI:sonfs=no \
--noclipping \
IONTOR_SETTINGS -AS:mrpc=50\
SOLEXA_SETTINGS -AS:mrpc=100

readgroup = torrent_single_end
data = iontorrent.fastq
technology = iontor

BWA tutorial

Contents

1. Introduction

BWA (Burrows-Wheeler Aligner)is a program that aligns short deep-sequencing reads to long reference sequences. Here is a short tutorial on the installation and steps needed to perform alignments. You can align the short reads to the genome or the transcriptome depending on the experiment/application.

2. Installation:

2.1 Download: http://sourceforge.net/projects/bio-bwa/files/ (**bwa-0.7.5a.tar.bz2**)

Install:

```
sudo chmod a+rwx
sudo cp /opt
sudo bunzip2 bwa-0.7.5a.tar.bz2
sudo tar xvf  bwa-0.7.5a.tar.bz2
sudo cd bwa-0.7.5a
sudo make
```

Set Path: Add bwa to your PATH by editing ~/.zshrc and adding

```
export PATH=$PATH:/opt//bwa-0.7.5a
```

Alternatively cp bwa, qualfa2fq.pl and xa2multi.pl to /usr/bin

2.2 Download the reference genome

```
wget http://hgdownload.cse.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz
```

Unzip it and concatenate the chromosome files

```
tar zvfx chromFa.tar.gz
cat *.fa > wg.fa
```

Then delete chromosome files:

```
rm chr*.fa
```

2.3 Download the mRNA sequences (RefSeq)

- Download SNVseeqer's http://physiology.med.cornell.edu/faculty/elemento/lab/files/refGene.txt.07Jun2010.fa (they represent the genomic counterparts of RefSeq mRNAs, i.e. transcription start site to end site with all introns removed). Pleas do not use the mRNA transcripts from the RefSeq Web site directly.
- The latest RefGene FASTA file can be generated from the RefSeq definition file using SNVseeqer/Adding a new annotation database.

3. Index the references

3.1 Create the index for the reference genome (assuming your reference sequences are in wg.fa)

```
bwa index -p hg19bwaidx -a bwtsw wg.fa (for mammalian genomes)
bwa index -p hg19bwaidx -a is wg.fa (for genomes < 2GB)
bwa index -p RefSeqbwaidx -a bwtsw refGene.txt.07Jun2010.fa (for transcripts)
```

where -p = name_of_output_file with idx indicating index
-a is or bwtsw for genomes < 2MB and mammalian respectively followed by reference sequence

Note index creation only needs to be performed once (the index does not have to be recreated for every alignment job).

4. Alignment of short reads

4.1 Mapping short reads to the reference genome, eg hg19

1. Align sequences using multiple threads (eg 4 CPUs). We assume your short reads are in the s_3_sequence.txt.gz file.

```
bwa aln -t 4 hg19bwaidx s_3_sequence.txt.gz >  s_3_sequence.txt.bwa
```

Notes:
(1) BWA can also take a compressed read file as input. So you can leave your read files compressed to save disk space.
(2) Problematic SAM output has been observed when aligning with more than 10 CPUs.

2. Create alignment in the SAM format (a generic format for storing large nucleotide sequence alignments):

```
bwa samse hg19bwaidx s_3_sequence.txt.bwa s_3_sequence.txt.gz > s_3_sequence.txt.sam
```

Note 1: BWA is capable of aligning reads stored in the compressed format (*.gz). You can gzip your reads to save disk space.

Note 2: for paired end reads, you need to align each end (R1 and R2) separarely:

```
bwa aln -t 4 hg19bwaidx s_3_1_sequence.txt.gz >  s_3_1_sequence.txt.bwa
bwa aln -t 4 hg19bwaidx s_3_2_sequence.txt.gz >  s_3_2_sequence.txt.bwa
bwa sampe hg19bwaidx s_3_1_sequence.txt.bwa s_3_2_sequence.txt.bwa
s_3_1_sequence.txt.gz s_3_2_sequence.txt.gz > s_3_sequence.txt.sam
```

Typically, after this step, you can split the reads using our split_samfile tool, or convert SAM to BAM

4.2 Mapping short reads to RefSeq mRNAs

1. Align sequences using multiple threads (eg 4). We assume your short reads are in the s_3_sequence.txt file.

```
bwa aln -t 4 RefSeqbwaidx s_3_sequence.txt >  s_3_sequence.txt.bwa
```

2. Create alignment in the SAM format (a generic format for storing large nucleotide sequence alignments):

```
bwa samse RefSeqbwaidx s_3_sequence.txt.bwa s_3_sequence.txt > s_3_sequence.txt.sam
```

4.3 Mapping long reads (454)

You can align 454 long reads using the bwasw command:

```
bwa bwasw hg19bwaidx 454seqs.txt > 454seqs.sam
```

5. Misc.

- **The SAMtools (http://samtools.sourceforge.net/samtools.shtml) can be used to convert the BWA/SAM reads into a so-called pile-up:**

First convet SAM to BAM and sort and index the BAM

```
samtools faidx wg.fa
      # creates index file wg.fa.fai

samtools import wg.fa.fai s_3_sequence.txt.sam s_3_sequence.txt.bam
      # bam= binary alignment/map format

samtools sort s_3_sequence.txt.bam s_3_sequence.txt.srt
      # sort by coordinate to streamline data processing

samtools index s_3_sequence.txt.srt.bam # a position-sorted BAM file can also be indexed
```

Then create the pileup. Note that pileups are not necessary to run DINDEL.

```
samtools pileup -f wg.fa s_3_sequence.txt.srt.bam
      # generate pilup pileup output
```

NOTE:

1. Convert sam to bam:

   samtools view -Sb file.sam > file.bam (-S input is in SAM format and "b" indicates that you'd like BAM output)

2. Than view bam file

   samtools tview will read alignments in bam format

   Usage: bamtk tview <aln.bam> [ref.fasta]

BWA requires different approaches depending on the type of input data. See the BWA Manual Reference Pages for further details.

Examples:

Common to all approaches is creation of the BWA index:

#it is more nicely organized if this is kept in it's own folder:

```
mkdir ref-index
cd ref-indx
ln -s /gnomes/ref-genome.fasta ref-genome.fasta
bwa index -p ref-genome ref-genome.fasta
```

Single end fastq with Illumina qualities:

(-I = Illumina qualities, -t 3 = use 3 processors)

```
bwa aln -I -t 3 ./ref-index/ref-genome ../Trimmed_reads/s_1_trimmed.fastq > s_1.aln.sai

bwa samse ./ref-index/ref-genome s_1.aln.sai ../Trimmed_reads/s_1_trimmed.fastq | gzip > s_1.sam.gz
```

#sort alignments and convert to BAM:

```
samtools view -uS s_1.sam.gz | samtools sort - s_1
```

Single end 454 (long) reads:

```
bwa bwasw -t 3 ./ref-index/ref-genome ../Trimmed_reads/454_trimmed.fastq | gzip > 454.sam.gz
```

#call Mismatch (MD) tag, sort, and convert to BAM:

```
samtools calmd -uS 454.sam.gz  ./ref-index/ref-genome.fasta | samtools sort - 454
```

Paired end short reads:

(align each side of the pair, then combine..)

```
bwa aln -t 3 ./ref-index/ref-genome ../Trimmed_reads/s_1_PE1.fastq > s_1_PE1.sai
bwa aln -t 3 ./ref-index/ref-genome ../Trimmed_reads/s_1_PE2.fastq > s_1_PE2.sai
bwa sampe ./ref-index/ref-genome s_1_PE1.sai
s_1_PE2.sai ../Trimmed_reads/s_1_PE1.fastq ../Trimmed_reads/s_1_PE2.fastq | gzip > s_1_PE12.sam.gz
```

#sort alignments and convert to BAM:

```
samtools view -uS s_1_PE12.sam.gz | samtools sort - s_1_PE12
```

(howto) Prepare a reference for use with BWA and GATK

 Geraldine_VdAuwera Posts: **3,620**Administrator, GSA Official Member admin
edited July 3 in Tutorials

Objective

Prepare a reference sequence so that it is suitable for use with BWA and GATK.

Prerequisites

- Installed BWA
- Installed SAMTools
- Installed Picard

Steps

1. Generate the BWA index
2. Generate the Fasta file index
3. Generate the sequence dictionary

---

1. Generate the BWA index

Action

Run the following BWA command:

```
bwa index -a bwtsw reference.fa
```

where `-a bwtsw` specifies that we want to use the indexing algorithm that is capable of handling the whole human genome.

Expected Result

This creates a collection of files used by BWA to perform the alignment.

---

2. Generate the fasta file index

Action

Run the following SAMtools command:

```
samtools faidx reference.fa
```

Expected Result

This creates a file called `reference.fa.fai`, with one record per line for each of the contigs in the FASTA reference file. Each record is composed of the contig name, size, location, basesPerLine and bytesPerLine.

---

3. Generate the sequence dictionary

Action

Run the following Picard command:

```
java -jar CreateSequenceDictionary.jar \
    REFERENCE=reference.fa \
    OUTPUT=reference.dict
```

Expected Result

This creates a file called `reference.dict` formatted like a SAM header, describing the contents of your reference FASTA file.

Post edited by Geraldine_VdAuwera on
- See more at:
http://gatkforums.broadinstitute.org/discussion/2798/howto-prepare-a-reference-for-use-with-bwa-and-gatk#sthash.MtHG3koG.dpuf

BWA Notes (from http://www.csc.fi/english/research/sciences/bioscience/programs/BWA)

1. Reference genome indexing

1.1 The fist step in creating alignment with BWA is downloading the reference genome and indexing it. You can use for example command ftp or wget to download a reference genome.

1.2 Use tar zvfx filename.tar.gz to uncompress and concatenate into one file using eg cat *.fa > wg.fa (wheer * is the wild card for the sequence files)

1.3 The $HOME directory is often too small for working with complete genomes in which case should do the analysis in temporary directories like $WRKDIR.Organise so that all the sequences are kept in it's own folder:

mkdir ref-index
cd ref-indx
ln -s /gnomes/ref-genome ref-genome.fasta
than bwa index -p -a is ref-genome ref-genome.fasta (to calculate the BWA indexes for this file

where: bwa index is the command
-p is the Prefix of the output database [same as db filename]
-a with option
is (if genome reference is < 2GB) (OR **bwtsw for human genome)**

2. Single-end alignment

Once the indexing is ready carry out the alignment for singe-end reads with command:

```
bwa mem filename.fa reads.fastq > aln.sam
```

WHAT I HAVE BEEN RUNNING:

1. Indexing

$ bwa index -p NC_008593idx -a is NC_008593.fasta

```
where -p is the prefix of the db / sequence
-a with option is is used for for genomes 2Gb
seq_file (is in fasta format)
```

2. aligning:

$ bwa mem NC_008593idx MGRRF-AeB_fervidicella-130713.fastq > aln.sam

where NC_008593idx is the index files created using 1, above

SAMtools

SAMtools is a set of utilities for interacting with and post-processing short DNA sequence read alignments in the SAM/BAM format, written by Heng Li. These files are generated as output by short read aligners like BWA. Both simple and advanced tools are provided, supporting complex tasks like variant calling and alignment viewing as well as sorting, indexing, data extraction and format conversion. [2] SAM files can be very large (10s of Gigabytes is common), so compression is used to save space. SAM files are human-readable text files, while BAM files are simply the binary equivalent. BAM files are typically compressed and more efficient for software to work with. SAMtools makes it possible to work directly with a compressed BAM file, without having to uncompress the whole file. Additionally, since the format for a SAM/BAM file is somewhat complex - containing reads, references, alignments, quality information, and user-specified annotations - SAMtools reduces the effort needed to use SAM/BAM files by hiding low-level details.

Usage and commands

Like many Unix commands, SAMtool commands follow a stream model, where data runs through each command as if carried on a conveyor belt. This allows combining multiple commands into a data processing pipeline. Although the final output can be very complex, only a limited number of simple commands are needed to produce it. If not specified, the standard streams (stdin, stdout, and stderr) are assumed. Data sent to stdout are printed to the screen by default but are easily redirected to another file using the normal Unix redirectors (> and >>), or to another command via a pipe (|).

SAMtools commands

SAMtools provides the following commands, each invoked as "`samtools` *`some_command`*".

view
: The `view` command filters SAM or BAM formatted data. Using options and arguments it understands what data to select (possibly all of it) and passes only that data through. Input is usually a sam or bam file specified as an argument, but could be sam or bam data piped from any other command. Possible uses include extracting a subset of data into a new file, converting between BAM and SAM formats, and just looking at the raw file contents. The order of extracted reads is preserved.

sort
: The `sort` command sorts a BAM file based on its position in the reference, as determined by its alignment. The element + coordinate in the reference that the first matched base in the read aligns to is used as the key to order it by. [TODO: verify]. The sorted output is dumped to a new file by default, although it can be directed to stdout (using the -o option). As sorting is memory intensive and BAM files can be large, this command supports a sectioning mode (with the -m options) to use at most a given amount of memory and generate multiple output file. These files can then be merged to produce a complete sorted BAM file [TODO - investigate the details of this more carefully].

index
: The `index` command creates a new index file that allows fast look-up of data in a (sorted) SAM or BAM. Like an index on a database, the generated </t>*.sam.sai or `*.bam.bai` file allows programs that can read it to more efficiently work with the data in the associated files.

tview
: The `tview` command starts an interactive ascii-based viewer that can be used to visualize how reads are aligned to specified small regions of the reference genome. Compared to a graphics based viewer like IGV,[3] it has few features. Within the view, it is possible to jumping to different positions along reference elements (using 'g') and display help information ('?').

mpileup
: The `mpileup` command produces a pileup format (or BCF) file giving, for each genomic coordinate, the overlapping read bases and indels at that position in the input BAM files(s). This can be used for SNP calling for example.

Examples

view

```
samtools view sample.bam > sample.sam
```

Convert a bam file into a sam file.

```
samtools view -bS sample.sam > sample.bam
```

Convert a sam file into a bam file. The `-b` option compresses or leaves compressed input data.

```
samtools view sample_sorted.bam "chr1:10-13"
```

Extract all the reads aligned to the range specified, which are those that are aligned to the reference element named *chr1* and cover its 10th, 11th, 12th or 13th base. The results is saved to a BAM file including the header. An index of the input file is required for extracting reads according to their mapping position in the reference genome, as created by *samtools index*.

```
samtools view -h -b sample_sorted.bam "chr1:10-13 > tiny_sorted.bam"
```

Extract the same reads as above, but instead of displaying them, writes them to a new bam file, *tiny.bam*. The `-b` option makes the output compressed and the `-h` option causes the SAM headers to be output also. These headers include a description of the reference that the reads in *sample.bam* were aligned to and will be needed if the *tiny.bam* file is to be used with some of the more advanced SAMtools commands. The order of extracted reads is preserved.

tview

```
samtools tview sample_sorted.bam
```

Start an interactive viewer to visualize a small region of the reference, the reads aligned, and mismatches. Within the view, can jump to a new location by typing g: and a location, like `g:chr1:10,000,000`. If the reference element name and following colon is replaced with `=`, the current reference element is used, i.e. if `g:=10,000,200` is typed after the previous "goto" command, the viewer jumps to the region 200 base pairs down on *chr1*. Typing `?` brings up help information.

sort

```
samtools sort unsorted_in.bam sorted_out
```

Read the specified *unsorted_in.bam* as input, sort it by aligned read position, and write it out to *sorted_out.bam*, the bam file whose name (without extension) was specified.

```
samtools sort -m 5000000 unsorted_in.bam sorted_out
```

Read the specified *unsorted_in.bam* as input, sort it in blocks up to 5 million k (5 Gb) [TODO: verify units here, this could be wrong] and write output to a series of bam files named *sorted_out.0000.bam*, *sorted_out.0001.bam*, etc., where all bam 0 reads come before any bam 1 read, etc. [TODO: verify this is correct].

index

```
samtools index sorted.bam
```

Creates an index file, *sorted.bam.bai* for the *sorted.bam* file.

Tablet

Tablet is a lightweight, high-performance graphical viewer for next generation sequence assemblies and alignments.

- High-performance visualization and data navigation.
- Display of reads in both packed and stacked formats.
- Paired end visualization support.
- File format support for ACE, AFG, MAQ, SOAP2, SAM and BAM.
- Import GFF3 features and quickly find/highlight/display them.
- Search and locate reads by name across entire data sets.
- Entire-contig overviews, showing data layout or coverage information.
- Simple install routine via auto-updating graphical installers.
- Support for Windows, Apple Mac OS X, Linux and Solaris, in 32 and 64-bit.

Installing Tablet:

1. Download the Tablet (linux 64 bit version) genome browser from http://bioinf.scri.ac.uk/tablet/

2. sudo cp the file from Downloads to /opt

3. sudo sh tablet to install (will be installed in /opt with symlink to /usr/local/bin

4. To run tablet, type: tablet at the comand prompt

Using tablet

1. Downloading of mapping results and reference fasta files on Detail view screen

   (1) Download a reference fasta file

   (2) Download a mapping result file (.sam format)

   Alignment files from Maq, SOAP, Bowtie, TopHat can download .sam format.

   Maq)
   command : maq2sam out_all.map > out.sam
   download file : out.sam

   bwa)
   command : bwa samse refgenome.fasta in.sai query1.fastq > out.sam
   bwa sampe refgenome.fasta in1.sai in2.sai query1.fastq query2.fastq > out.sam

   download file : out.sam

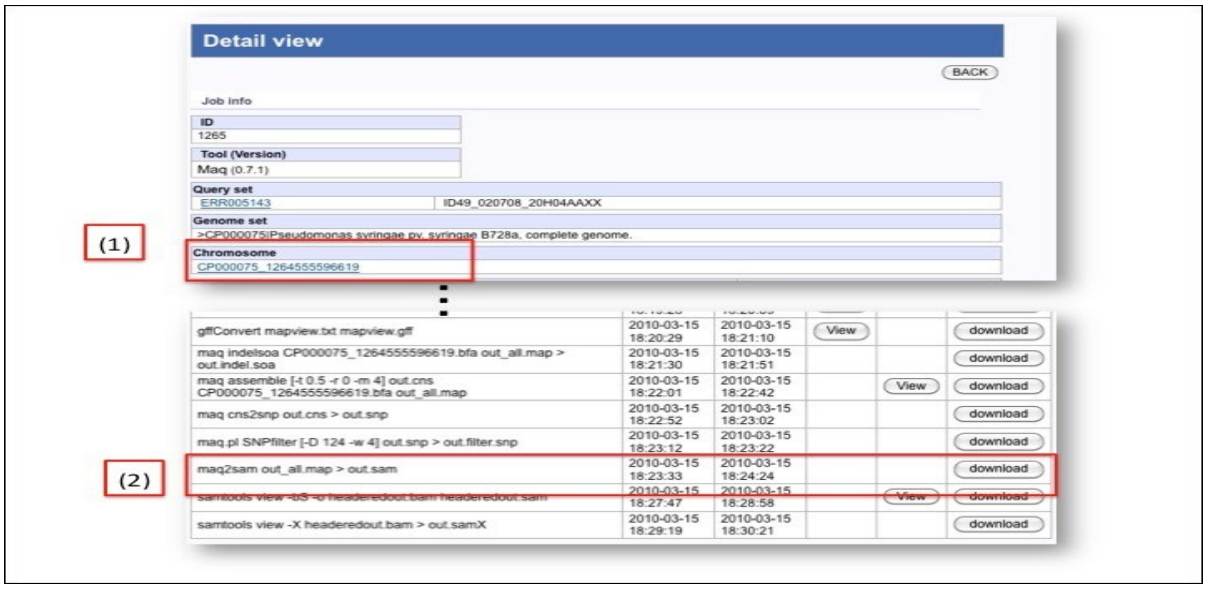
   SOAP)
   command : soap2sam paired_out.map > out.sam
   download file : out.sam

   Bowtie)
   command : samtools view out2.bam > out.sam

download file : out.sam

TopHat)
command : samtools view out2.bam > out.sam
download file : out.sam

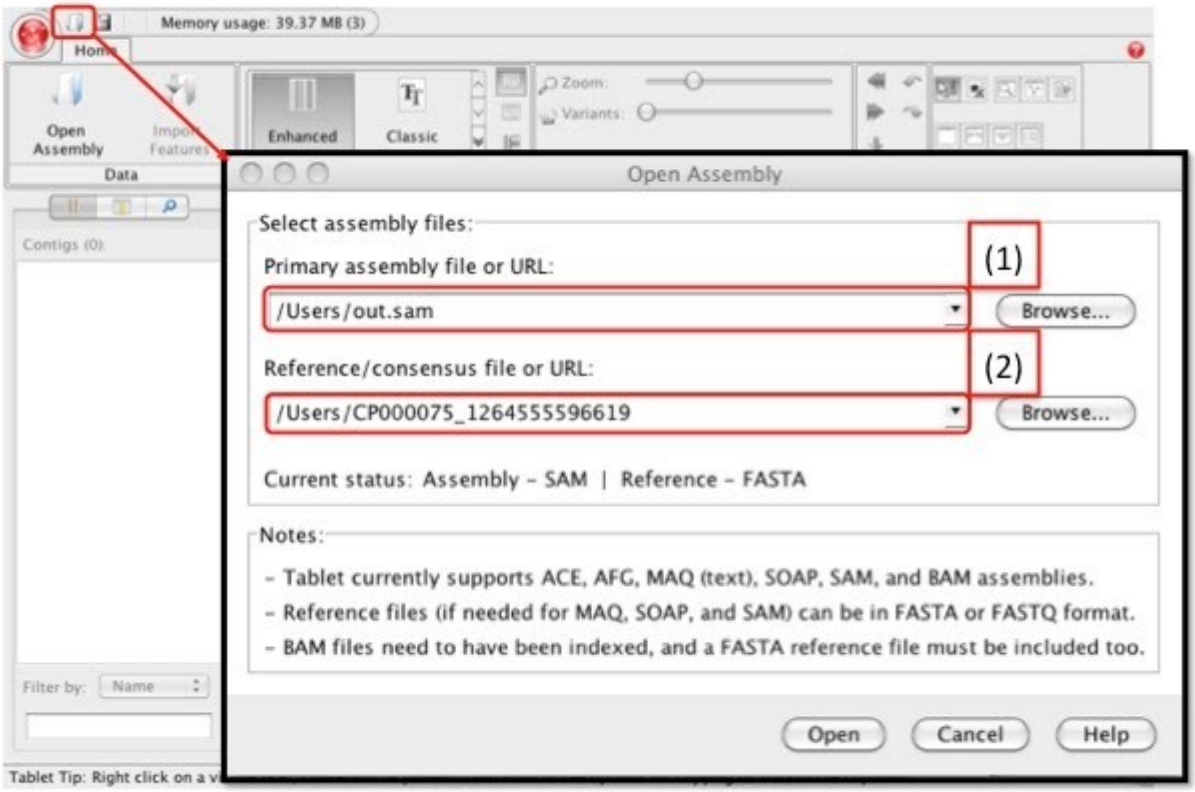
Fig. 1. Download of mapping results and reference fasta files (case of Maq)



2. Run Tablet

3. Open a new assembly in Tablet from a file on disk.
(1)Select a mapping result file(.sam format) on disk.
(2)Select a reference fasta file on disk.

Fig. 2. Open a new assembly in Tablet from a file on disk



4. Select a contig
   Select a contig to begin visualization.

Fig. 3. Select a contig

NGS File Formats

sff

- fastQ – quality
- fastA – no quality

File format converters:

- sff_extract
  - to fastq, fasta and xml
- fastaQual2fastq.pl
  - pass a fasta and a fasta-quality file

ContigScape
Requires Java and Cytoscape

1. Cytoscape
   - Download Cytoscape 2.8.3 installation file for Linux from www.cytoscape.org
   - Change the execution file property : sudo chmod a+x Cytoscape_2.8.3_unix.sh
   - Install: sudo sh Cytoscape_2_8_3_unix.sh (installs in /opt)
   - run command: Cytoscape &

2. ContigScape1.0:
   - Download ContigScape.jar from http://sourceforge.net/projects/contigscape/?source=directory
   - sudo mv ContigScape.jar to /opt/Cytoscape_v2.8.3/plugins
   - gedit .zshrc and add: export PATH=$PATH:/opt/Cytoscape_v2.8.3/Cytoscape (there are two versions, **c**ytoscape (version) 2.7 and **C**ytoscape (version 2.8.3) – differentiated on the basis of the small and Capital "c"
   - In terminal, run command: Cytoscape &.
   - Follow instructions from the Contigspace document file.

SINA

Sina:

1. Download SINA (v 1.2.11, 4 bit, ubuntu 12.04) from: http://www.arb-silva.de/aligner/sina-download/

2. Unpack the archive

```
$ tar -xvzf sina-1.2.10.tgz
```

3. Change into the unpacked directory and check that SINA is working

```
$ cd sina-1.2.10
$ ./sina --version
SINA v1.2.10 (svn-20432)
```

4. Get yourself a suitable reference alignment. For SSU and LSU, the SILVA NR datasets are a good starting point. Alternatively, if you have some other reference alignment you want to use in multi-FASTA format, convert it to ARB format like this:

```
$ ./sina -i reference.fasta -o reference.arb –prealigned
```

5. Make sure it's valid multi-FASTA though. The first word of each header must be unique for each sequence!

6. Try aligning some sequences:

```
$ ./sina -i mysequences.fasta -o aligned.fasta --ptdb reference.arb
```

The first time you do that, the ARB PT server used by SINA to quickly find reference sequences for alignment will build it's index. This may take a while, but you will only have to repeat this if you change the reference alignment. The PT server will also continue to run. Use `"killall arb_pt_server"` to stop all your running PT servers.

7. If you used the SILVA NR dataset, you can classify your sequences like this:

```
$ ./sina -i mysequences.fasta -o aligned.arb --ptdb reference.arb \
 --search --search-db reference.arb \
 --lca-fields tax_slv
```

8. Check the manual to find out about the rest of the options.

```
$ ./sina manual
```

9. Have fun, find out something great and cite us when you publish (Elmar Pruesse; Jörg Peplies; Frank Oliver Glöckner (2012). SINA: accurate high throughput multiple sequence alignment of ribosomal RNA genes.Bioinformatics 2012; doi: 10.1093/bioinformatics/bts252)

Krona

1. Download krona from http://sourceforge.net/projects/krona/. This will be downloaded in the directory Download.
2. Unpack the archive.
3. cd to the resulting directory
4. sudo run ./install.pl and will install by default in /usr/local
5. To install in another directory use the following:
   - "--prefix \<path>" - scripts will be installed in the bin directory within this path. (The default is /usr/local/ as described in 4, above
   - "--taxonomy \<path>" - taxonomy files will be stored in this directory when updateTaxonomy.sh is run. The default is taxonomy/ within the unpacked Krona Tools directory. If the taxonomy database was installed in a previous version of Krona Tools, it can be reused by moving it to the to new Krona Tools folder or by pointing to it with this option.

Taxonomy database

To be able to use import scripts that rely on NCBI taxonomy, **updateTaxonomy.sh** must be run after installing. This will install the local taxonomy database, which uses about 1.5 GB of space and requires an additional 4GB of scratch space during installation. It can also be run later to keep the local database up to date with NCBI.

For installations with no internet connection:

1. Download these files from ftp://ftp.ncbi.nih.gov/pub/taxonomy:
   - gi_taxid_nucl.dmp.gz
     (wget ftp://ftp.ncbi.nih.gov/pub/taxonomy/gi_taxid_nucl.dmp.gz)
   - gi_taxid_prot.dmp.gz
     (wget ftp://ftp.ncbi.nih.gov/pub/taxonomy/gi_taxid_prot.dmp.gz)
   - taxdump.tar.gz
     (wget ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz)
2. Transfer the files to the **taxonomy** folder in the standalone KronaTools installation.
3. Run "**./updateTaxonomy.sh --local**".

Note 1: Installation error and fix
Error-- I've tried to install Krona-2.3 on a Linux x86_64, but it generates the error prompt below. Can you help resolve this?

Use of qw(...) as parentheses is deprecated at ./install.pl line 56.
ERROR: does not exist and couldn't create

Error Fix – I figured out the fix. After the for loop in the install.pl file qw(should actually be in parentheses, like below (qw (ClassifyBLAST ....))

Note 2:
The error in note 1, above, does not occur with Krona-2.4. Installed as per the instructions to /usr/local/bin

Using krona:
RDP

- Taxonomic classifications of the Ribosomal Database Project 16S Classifier can be imported

using `ktImportRDP`. To download classifications:

1. Run the classification tool.
2. Click "show assignment detail" for the root of the hierarchy.
3. Click "download allrank result" or "download fixrank result".

- Comparisons from the RDP Library Compare tool can be imported using `ktImportRDPComparison`. To download comparison results:
    1. Run the comparison tool.
    2. Click "download comparison result as text".

PhymmBL

- A taxonomical profile of PhymmBL classifications can be created using `ktImportPhymmBL`.
- Example:
    - `ktImportPhymmBL phymmbl/results.03.*`

BLAST

- A taxonomic profile of BLAST results can be created with `ktImportBLAST`. Results must in tabular format (use "Hit table [text]" when downloading results from NCBI). GI numbers from the top hits are used to obtain NCBI taxonomy IDs, which are used to create a tree. Ties for top hits can be broken by going up to the lowest common ancestor (default) or by picking one at random. The Krona Tools installation must have local taxonomy information installed using `updateTaxonomy.sh` (see [Installing]).

MG-RAST

- Taxonomic or functional classifications can be imported from the MG-RAST metagenomics server using `ktImportMG-RAST`.
- Downloading data from MG-RAST:
    1. From the home page, click Browse Metagenomes.
    2. Select a project, then select a metagenome from that project (e.g. 4441138.3).
    3. Click on the small bar chart icon to analyze the metagenome.
    4. Under Analysis Views, choose Hierarchical Classification (organism or functional) or Lowest Common Ancestor.
    5. Select "table" under Data Visualization and click the "generate" button .
    6. Adjust the table options as desired and click "download data matching current filter".

METAREP

- A taxonomical profile of a METAREP data set can be created with `ktImportMETAREP`. NCBI taxonomy IDs from the top hits in the blast.tab file of the data set are used to create a tree. Query lengths are use for magnitudes. Ties for top hits can be broken by going up to the lowest common ancestor (default) or by picking one at random. The Krona Tools installation must have local taxonomy information installed using `updateTaxonomy.sh` (see [Installing]).
- To download and import data from METAREP:
    1. From the home page, log in or choose "Try It".
    2. View a project.
    3. Choose "Download" for desired libraries in the project or choose "Download All Libraries" and unpack the downloaded file.
    4. Unpack each library to be imported.
    5. Unzip blast.tab.gz in each of the library folders to be imported.
    6. Run `ktImportMETAREP` with the desired library folders.
        - Example:
            - `ktImportMETAREP HOT01-0010M HOT02-0070M`

MEGAN

- MEGAN taxonomic classifications can be imported using `ktImportTaxonomy`.
- To export classifications:
    1. From the menu, choose File -> Export -> Assignments to CSV.
    2. In "Choose format", choose "taxon-id, count(s)".
    3. In "Choose separator", choose "tab".

NCBI workbench

1. Download the NCBI debian package:

ftp://ftp.ncbi.nlm.nih.gov/toolbox/gbench/ver-2.7.6/gbench-linux-Ubuntu-precise_amd64-2.7.6.deb

2. Install: sudo dpkg -i gbench-linux-Ubuntu-precise_amd64-2.7.6.deb

Note: This should remove the older version of the workbench and replace with the new version

3. By default the program is installed in /opt and in the directory /ncbi

4. Set path in .zshrc:

export PATH=$PATH:/opt/path/ncbi/gbench-2.7.6 (2.7.6 is the latest version installed on 18 July 2013; change path to reflect any new vesrions)

5. Go to dashboard and search for “gbench”. Click on “NCBI Genome Workbench” and a GUI will open **OR** run from a command line: /opt/ncbi/gbench-2.7.6/bin/gbench **OR** install in the Desktop panel and click to run.

How to proceed from Assembly to Draft Genome

This is not as straightforward as you might think and the process of moving from contigs to a draft genome is difficult. This can vary on the type of organism, the size of the genome, the frequency of gene space, the depth of sequencing, etc. You didn't give us any of this information so it's hard to guide you. There are numerous ways to close contigs or scaffolds but much of this depends on how similar your sequenced contigs are to those from already sequenced genomes.

If you have a relatively small genome (bacterial or very small eukaryote less than ~40Mb), I have had good luck using the PAGIT pipeline (Post Assembly Genome Improvement Tool) (see these links for the pipeline – http://www.sanger.ac.uk/resources/software/pagit/ and paper –http://www.nature.com/nprot/journal/v7/n7/full/nprot.2012.068.html). This pipeline (as well as others like this) require a lot of memory, so you won't be able to do this on a laptop or desktop computer.

Calculating coverage

1. Data generated on a 454Titanium at the University of Florida:

	Raw Data		Analysed data - Newbler		
	No of reads	No of bases	Inferred read error	No aligned reads	No alig
Fragment library:					
RC3_F031EKX04.sff	81678	29099796	1.07%	81039 (99.2%)	28790
RC3_GBS6S5403.sff	115058	42950289	0.80%	114303 (99.3%)	42625
8kb paired end library:					
RC3-PE_GBS6S5404.sff	81599	14481102	1.09%	86582 (83%)	93424
F7ETK1T05.sff	5246	911539	1.20%	4720 (88.7%)	54605

2. Coverage:

$$C = \frac{\text{Length of reads x No of reads}}{\text{Genome Size}}$$

$$= \frac{28790686 + 42625079 + 9342499 + 546052}{3{,}000{,}0000}$$

$$= \frac{81304316}{3000000}$$

$$= \sim 27$$

Assembly

GS Junior:
Throughput 100,000 reads per 10 hour run > 35 million filtered bases
Read length: Modal 500 bases, average 400 bases
Accuracy: Q20 (99%) at 400 bases
Comes with desktop computer and software to do some standard processing
Cost will be approx $100K to $125K
Basic sequencing Chemistry - the same as 454.

QUESTION:
Im new to 454 data analysis. I have a 6mbp genome that has been sequenced at>10X. this is an emerging bacterial pathogen.
what sort of post 454 data analysis should one do? does anyone have any standard operating procedures? As of now, I have a broad idea of what I would like to do. I have listed some of them below. I would appreciate any comments/suggestions.
1)with the 400 or so contigs, I intend to align it with reference genomes(mauve? - will this help me in judging the quality of assembly?).
2)carry out a rast server based annotation
3)extract nucleotide and protein sequences from the gbk output of rast. (what is the best way to do it such that protein sequences are obtained as fastA with the product name as header rather than contig number?)
4)carry out multiway blast searches against related genomes
5)look out for presence of plasmids in the contigs by searching for regions of high coverage?
6)look for regions of atypical nucleotide composition ?

ANSWER
I think I can plug xBASE services here.

I suggest you use my-xBASE to align your data to a reference, also try de-novo assembly. Then use the annotation service to generate a preliminary annotation.

xBASE main page - my-xBASE and annotation services linked in the right column.

for (5), I plot a graph of contig length vs. reads/length. Plasmid contigs stick out from the vast majority of large contigs at a non-integral multiple of the read-density.

Error for jango settings. Did the following:
cd to /usr/share/pyshared/django/conf
sudo vi global_settings.py
changed Debug to True
add ALLOWED_HOSTS = [http://ng.xbase.ac.uk]
See notes below

Django 1.5 introduced the allowed hosts setting that is required for security reasons. A settings file created with Django 1.5 has this new section which you need to add:

```
# Hosts/domain names that are valid for this site; required if DEBUG is False
# See https://docs.djangoproject.com/en/1.5/ref/settings/#allowed-hosts
ALLOWED_HOSTS = []
```

Add your host here like `['www.beta800.net']` or `['*']` for a quick test, but don't use `['*']` for production.

MicrobeDB

1. Main Features

- Centralized storage and access to completed archaeal and bacterial genomes
  - Genomes obtained from NCBI RefSeq:
    http://www.ncbi.nlm.nih.gov/genomes/lproks.cgi
  - Genome/Flat files are stored in one central location
  - Including files .gbk, .gff, .fna, .faa, etc.
  - Unpublished genomes can be added as well

- Information at the genome project, chromosome, and gene levelare parsed and stored in a MySQL database

- A Perl MicrobeDB API provides non-MySQL interface with the database.

2. Main MicrobeDB Tables

- Version
  - Each download of genomes from NCBI is given a new version number
  - Data will not change if you always use the same version number of microbedb
  - Version date can be cited for any method publications
  - A version can be saved by users so not automatically deleted.

- Genome Project
  - Contains information about the genome project and the organism that was sequenced
- Each genome project contains one or more replicons

- Replicon
  - Chromosome, plasmids, or contigs
  - Each replicon contains one or more genes

- Gene
  - Contains gene annotations and also the DNA and protein sequences (if protein coding gene)

3. Accessing MicrobeDB

• Any traditional MySQL programs
  – phpMyAdmin:
    – Web-based
    – http://phpmyadmin.net
  – MySQL Workbench
    – Local desktop client
    – http://www.mysql.com/products/workbench/

• MicrobeDB Perl API
  – Allows interaction with database directly from within a Perl script
  – Requires no knowledge of SQL

mysql configuration file: .my.cnf

This can be found at:

/etc/mysql/my.cnf
/etc/mysql/my.cnf.dpkg-old
/home/bharat/.my.cnf_old

The problem is likely your .my.cnf file. It appears to have two errors. The first line should be '[client]'

and the username should probably just be 'microbedb'.

It should look like this:

[client]
host=localhost
user=microbedb
password=patel

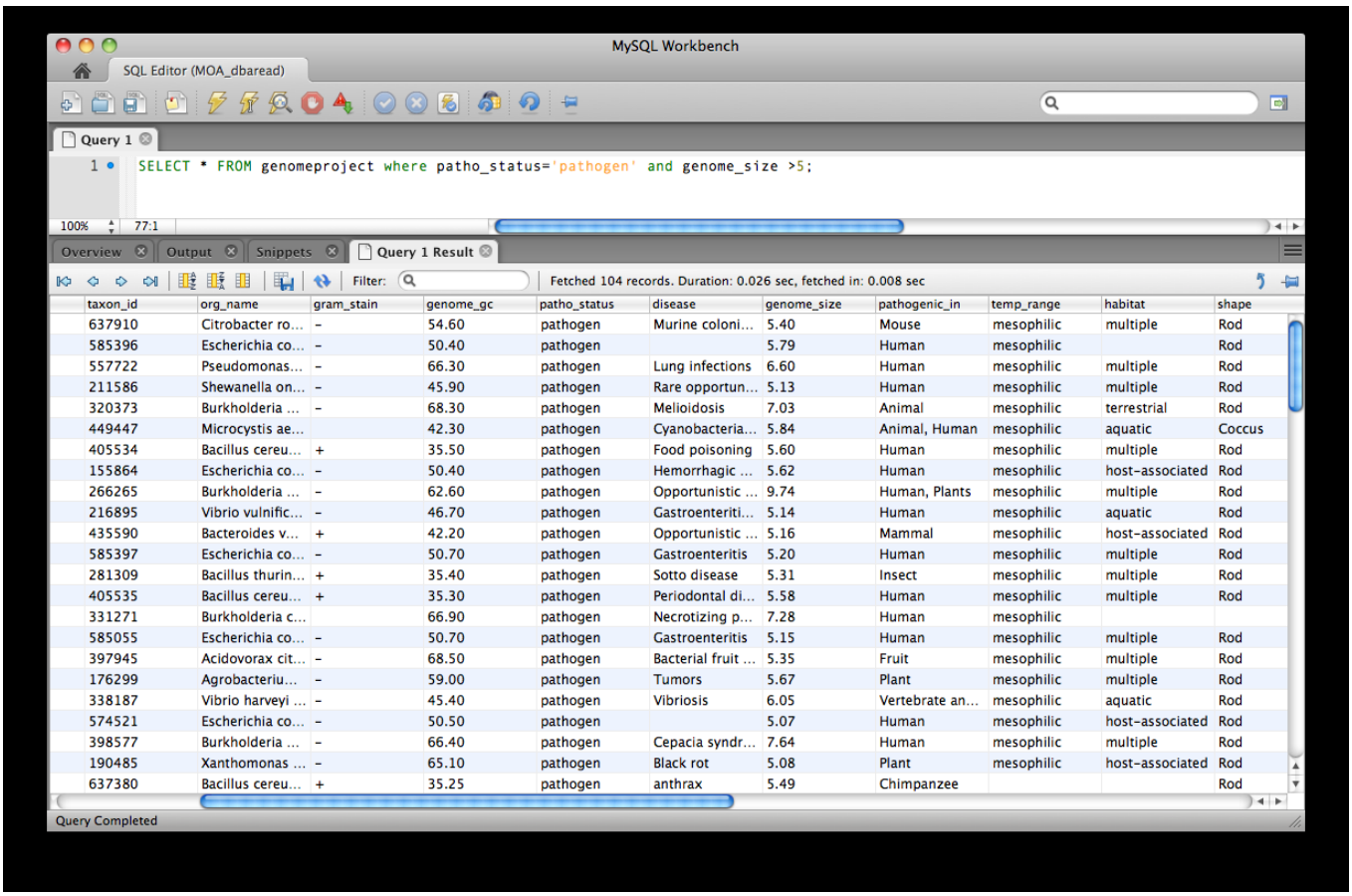
To test if it works correctly, you should be able to simply type the command 'mysql' from a terminal/console and get access to a mysql prompt. If you get an error then there is something wrong, and you need to check that you created the user 'microbedb' with the password 'patel'.

MicrobeDB Annotations

Table/Object*	Field Descriptions*	Example
Genome Project	Organism Name	Pseudomonas aeruginosa LESB58
	NCBI Taxon ID	557722
	Genome Size (Mb)	6.6
	Pathogenic In	Human
	GC %	66.3
	Oxygen Requirements	aerobic
	Sequencing Centre	Wellcome Trust Sanger Institute
Replicon	Replicon Type	Chromosome
	Accession (RefSeq)	NC_011770
	Replicon Size (bp)	6601757
	Number of Genes	6027
	Replicon Sequence	TTTAAAGAG…
Gene	Gene Type	CDS
	Locus ID	PLES_00001
	Start Position	483
	End Position	2027
	Gene Name	dnaA
	Product	chromosomal replication initiation
	DNA Sequence	GTGTCCGT…
	Protein Sequence	MSVELWQQ...
Version	Download Date	2011-12-17
	Flat File Directory	/share/genomes/2011-12-17/
	Used By	Morgan, Matthew

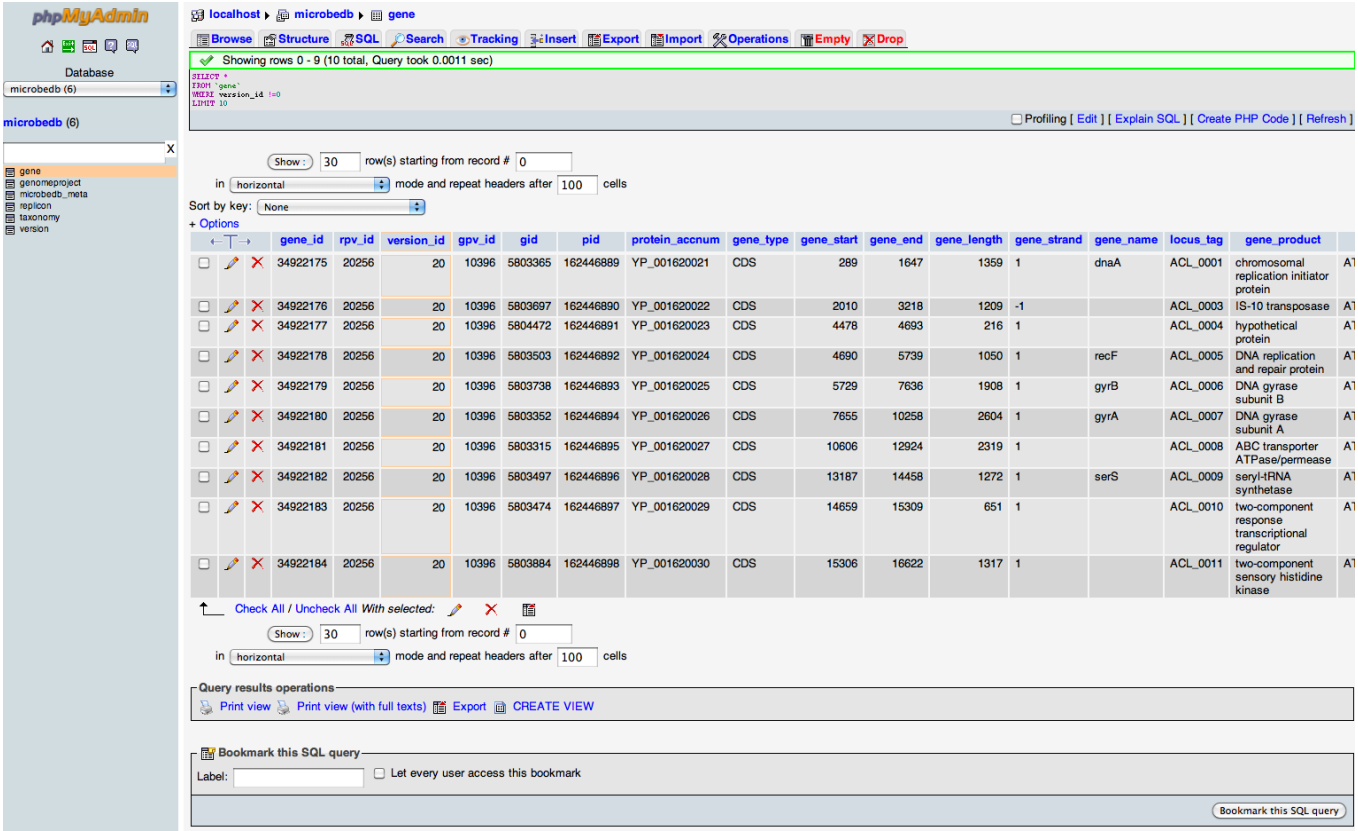
*Not all fields and tables in MicrobeDB are listed.

Mysql work bench



taxon_id	org_name	gram_stain	genome_gc	patho_status	disease	genome_size	pathogenic_in	temp_range	habitat	shape
637910	Citrobacter ro...	-	54.60	pathogen	Murine coloni...	5.40	Mouse	mesophilic	multiple	Rod
585396	Escherichia co...	-	50.40	pathogen		5.79	Human	mesophilic		Rod
557722	Pseudomonas...	-	66.30	pathogen	Lung infections	6.60	Human	mesophilic	multiple	Rod
211586	Shewanella on...	-	45.90	pathogen	Rare opportun...	5.13	Human	mesophilic	multiple	Rod
320373	Burkholderia ...	-	68.30	pathogen	Melioidosis	7.03	Animal	mesophilic	terrestrial	Rod
449447	Microcystis ae...		42.30	pathogen	Cyanobacteria...	5.84	Animal, Human	mesophilic	aquatic	Coccus
405534	Bacillus cereu...	+	35.50	pathogen	Food poisoning	5.60	Human	mesophilic	multiple	Rod
155864	Escherichia co...	-	50.40	pathogen	Hemorrhagic ...	5.62	Human	mesophilic	host-associated	Rod
266265	Burkholderia ...	-	62.60	pathogen	Opportunistic ...	9.74	Human, Plants	mesophilic	multiple	Rod
216895	Vibrio vulnific...	-	46.70	pathogen	Gastroenteriti...	5.14	Human	mesophilic	aquatic	Rod
435590	Bacteroides v...	+	42.20	pathogen	Opportunistic ...	5.16	Mammal	mesophilic	host-associated	Rod
585397	Escherichia co...	-	50.70	pathogen	Gastroenteritis	5.20	Human	mesophilic	multiple	Rod
281309	Bacillus thurin...	+	35.40	pathogen	Sotto disease	5.31	Insect	mesophilic	multiple	Rod
405535	Bacillus cereu...	+	35.30	pathogen	Periodontal di...	5.58	Human	mesophilic	multiple	Rod
331271	Burkholderia c...		66.90	pathogen	Necrotizing p...	7.28	Human	mesophilic		
585055	Escherichia co...	-	50.70	pathogen	Gastroenteritis	5.15	Human	mesophilic	multiple	Rod
397945	Acidovorax cit...	-	68.50	pathogen	Bacterial fruit ...	5.35	Fruit	mesophilic	multiple	Rod
176299	Agrobacteriu...	-	59.00	pathogen	Tumors	5.67	Plant	mesophilic	multiple	Rod
338187	Vibrio harveyi ...	-	45.40	pathogen	Vibriosis	6.05	Vertebrate an...	mesophilic	aquatic	Rod
574521	Escherichia co...	-	50.50	pathogen		5.07	Human	mesophilic	host-associated	Rod
398577	Burkholderia ...	-	66.40	pathogen	Cepacia syndr...	7.64	Human	mesophilic	multiple	Rod
190485	Xanthomonas ...	-	65.10	pathogen	Black rot	5.08	Plant	mesophilic	host-associated	Rod
637380	Bacillus cereu...	+	35.25	pathogen	anthrax	5.49	Chimpanzee			Rod

PhpMyAdmin



gene_id	rpv_id	version_id	gpv_id	gid	pid	protein_accnum	gene_type	gene_start	gene_end	gene_length	gene_strand	gene_name	locus_tag	gene_product
34922175	20256	20	10396	5803365	162446889	YP_001620021	CDS	289	1647	1359	1	dnaA	ACL_0001	chromosomal replication initiator protein
34922176	20256	20	10396	5803697	162446890	YP_001620022	CDS	2010	3218	1209	-1		ACL_0003	IS-10 transposase
34922177	20256	20	10396	5804472	162446891	YP_001620023	CDS	4478	4693	216	1		ACL_0004	hypothetical protein
34922178	20256	20	10396	5803503	162446892	YP_001620024	CDS	4690	5739	1050	1	recF	ACL_0005	DNA replication and repair protein
34922179	20256	20	10396	5803738	162446893	YP_001620025	CDS	5729	7636	1908	1	gyrB	ACL_0006	DNA gyrase subunit B
34922180	20256	20	10396	5803352	162446894	YP_001620026	CDS	7655	10258	2604	1	gyrA	ACL_0007	DNA gyrase subunit A
34922181	20256	20	10396	5803315	162446895	YP_001620027	CDS	10606	12924	2319	1		ACL_0008	ABC transporter ATPase/permease
34922182	20256	20	10396	5803497	162446896	YP_001620028	CDS	13187	14458	1272	1	serS	ACL_0009	seryl-tRNA synthetase
34922183	20256	20	10396	5803474	162446897	YP_001620029	CDS	14659	15309	651	1		ACL_0010	two-component response transcriptional regulator
34922184	20256	20	10396	5803884	162446898	YP_001620030	CDS	15306	16622	1317	1		ACL_0011	two-component sensory histidine kinase

Programming with the MicrobeDB API

If you know how to program in Perl you can use the MicrobeDB Perl API which allows you to retrieve data without constructing MySQL queries.

1. Example 1 – search genomes from pathogens:

```
#Use the MicrobeDB Search library
use MicrobeDB::Search;

#create the search object
my $search_obj= new MicorbeDB::Search();

#Create an object with certain features that we want (i.e. only pathogens)
my $obj = new GenomeProject( version_id => '1', patho_status => 'pathogen' );

#This does the actual search and returns a list of all genome projects that match search parameters
my @result_objs = $search_obj->object_search($obj);

#Now we can iterate through each genome project
foreach my $gp_obj (@result_objs) {

        #get the name of the genome
        $gp_obj->org_name()
foreach my $gene_obj ($gp_obj->genes()){

                if($gene_obj->gene_type() eq 'tRNA'){
                        #write the genes in fasta format with gid as the identifier
                        print '>',$gene_obj->gid,"\n",$gene_obj->gene_seq();

}}}
```

Example 2 – search for recA genes and print in fasta format

- Example of a simple perl script using the MicrobeDB API that searches for all 'recA' genes and prints them in 'Fasta' format:

```
#Import the MicrobeDB API
use lib '/your/path/to/MicrobeDB';
use MicrobeDB::Search;

#intialize the search object
$search_obj= MicrobeDB::Search();

#create the object that has properties that must match in the database
$gene_obj= MicrobeDB::Gene(gene_name => 'recA');

#do the actual search
@genes = $search_obj->object_search($gene_obj);

#loop through each gene we found and print in FASTA format
foreach my $gene (@genes){
print'>',$gene->gid(),"\n",$gene->gene_seq(),"\n";
}
```

Example 3. retrieves all annotated 16s genes and outputs them in fasta file format.

```
!/usr/bin/env perl
```

```
#Copyright (C) 2011 Morgan G.I. Langille
#Author contact: morgan.g.i.langille@gmail.com

#This file is part of MicrobeDB.

#MicrobeDB is free software: you can redistribute it and/or modify
#it under the terms of the GNU General Public License as published by
#the Free Software Foundation, either version 3 of the License, or
#(at your option) any later version.

#MicrobeDB is distributed in the hope that it will be useful,
#but WITHOUT ANY WARRANTY; without even the implied warranty of
#MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#GNU General Public License for more details.

#You should have received a copy of the GNU General Public License
#along with MicrobeDB.  If not, see <http://www.gnu.org/licenses/>.

#Example of how to use the search api to get information from microbedb using an
object as the search field
#Searchable objects are:
#GenomeProject, Replicon, Gene, Version, or UpdateLog

#See table_search_example.pl, if you want to do a simple search on a mysql db table
that is not part of the microbedb api

#This script retrieves all annotated 16s genes and outputs them in fasta file
format.

use warnings;
use strict;

use lib "../../../";

#we need to use the Search library (this also imports GenomeProject,Replicon, and
Gene libs)
use MicrobeDB::Search;

warn "What version?\n";
my $version_id=<STDIN>;
chomp($version_id);
#Create an object with certain features that we want (e.g rep_type='chromosome')
my $rep_obj = new MicrobeDB::Replicon( version_id => $version_id);

#Create the search object.
my $search_obj = new MicrobeDB::Search();

#do the actual search using the replicon object to set the search parameters
#all objects that match the search criteria are returned as an array of the same
type of objects
my @result_objs = $search_obj->object_search($rep_obj);

#iterate through each replicon object that was returned
foreach my $curr_rep_obj (@result_objs) {

        #get the name of the replicon
   my $rep_name = $curr_rep_obj->definition();

        #get the replicon accesion
      my $rep_accnum = $curr_rep_obj->rep_accnum();

        #get all genes associated with this chromosome
```

```
  my $genes = $curr_rep_obj->genes();

        foreach my $curr_gene (@$genes){
            #check to see if the gene is annotated as a 16s rRNA
            if($curr_gene->gene_type() eq 'rRNA'){
              my $rna_product = $curr_gene->gene_product();
           if(defined($rna_product) && $curr_gene->gene_product =~ /16s/i){
                my $gid = $curr_gene->gid();
            my $start = $curr_gene->gene_start();
           my $end = $curr_gene->gene_end();
               my $seq = $curr_gene->gene_seq();

                #print out the gene in fasta format
             print ">$rep_accnum|gi:$gid|$start - $end|$rna_product\n";
              print $seq . "\n";
              }
           }
   }
}
```

Example 4 - prints out a fasta formatted list of 'recA' genes from genomes that are described as pathogens.

```
#!/usr/bin/env perl
#Copyright (C) 2011 Morgan G.I. Langille
#Author contact: morgan.g.i.langille@gmail.com

#This file is part of MicrobeDB.

#MicrobeDB is free software: you can redistribute it and/or modify
#it under the terms of the GNU General Public License as published by
#the Free Software Foundation, either version 3 of the License, or
#(at your option) any later version.

#MicrobeDB is distributed in the hope that it will be useful,
#but WITHOUT ANY WARRANTY; without even the implied warranty of
#MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#GNU General Public License for more details.

#You should have received a copy of the GNU General Public License
#along with MicrobeDB.  If not, see <http://www.gnu.org/licenses/>.

#This script prints out a fasta formatted list of 'recA' genes from genomes that are
described as pathogens.

use strict;
use warnings;

#Import the MicrobeDB API
use lib '../../../';
use MicrobeDB::Search;

#intialize the search object
my $search_obj= new MicrobeDB::Search();


#create the object that has properties that must match in the database
my $gene_obj= new MicrobeDB::Gene(gene_name => 'recA');

#do the actual search
my @genes = $search_obj->object_search($gene_obj);
```

```
#loop through each gene we found
foreach my $gene (@genes){

    #get genome associated with this gene
    my $genome=$gene->genomeproject();

    #only interested in 'pathogen' genomes
    if(defined($genome->patho_status()) && $genome->patho_status() eq 'pathogen'){

        #print out the fasta header line using information from the genome and from
the gene
    print '>',$genome->org_name(),'|',$gene->gid,'|',$gene->gene_name(),"\n";

        #print out the DNA sequence
     print $gene->gene_seq(),"\n";
    }
}
```

3. Example 3 – search for genomes with habitat listed as aquatic, and then prints out their genome sizes and GC%.

```
#!/usr/bin/env perl
#Copyright (C) 2011 Morgan G.I. Langille
#Author contact: morgan.g.i.langille@gmail.com

#This file is part of MicrobeDB.

#MicrobeDB is free software: you can redistribute it and/or modify
#it under the terms of the GNU General Public License as published by
#the Free Software Foundation, either version 3 of the License, or
#(at your option) any later version.

#MicrobeDB is distributed in the hope that it will be useful,
#but WITHOUT ANY WARRANTY; without even the implied warranty of
#MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#GNU General Public License for more details.

#You should have received a copy of the GNU General Public License
#along with MicrobeDB.  If not, see <http://www.gnu.org/licenses/>.

#This script looks for genomes with habitat listed as aquatic, and then prints out their genome sizes and GC%.

use strict;
use warnings;

#Import the MicrobeDB API
use lib '../../../';
use MicrobeDB::Search;

#intialize the search object
my $search_obj= new MicrobeDB::Search();

#create the object that has properties that must match in the database
my $gp_obj= new MicrobeDB::GenomeProject(habitat => 'aquatic');

#do the actual search
my @genomes = $search_obj->object_search($gp_obj);

#loop through each genomes we found
for each my $genome (@genomes){

   #get the metadata we are interested in
   my $size = $genome->genome_size();
   my $gc=$genome->genome_gc();

   #print out a table of genomes
   print join("\t",$genome->org_name,$size,$gc),"\n";
}
```

1. $mysql [options] [database] – options will be listed if you use: mysql –-help

2. $mysql -u root – to connect to the serve.
   Enter password
   A prompt will appear - >mysql

3. From within the prompt,
   $>mysqladmin -u root password 'secret_password'
   >SET PASSWORD FOR 'root'@'localhost' = PASSWORD('secret_password'); -- check accurate

4. $>mysql \q – to exit

5. Selecting a database:
   mysql -u root -p another database: select two databases eg mysql and microbeDB:
   >use microbeDB: select a database after you login to mysql and get a prompt

6.mysqladmin (administers mysql) – create, delete, shut down databases; update privilege tables, view mysql processes
   >mysqladmin [options] command(s); check options list with >mysqladmin --help

   A new database is created as follows:
   >mysqladmin -u root -p create widgets (widget is the new database) and
      Enter Password:

7. Securing database: is accomplished through modifications made to the tables (previlages) found in the mysql database. Sthe pecial commands used are GRANT and REVOKE (INSERT, UPDATE and DELETE are deprecated commands)

GRANT- used to create new users & assign privileges to users

Summary documentation for PostgreSQL

Full documentation at http://www.postgresql.org/docs/9.1/static/index.html

1. Installation: Via Ubuntu synaptic manager
   - /usr/bin/psql
   - check version by: psql –version (for client version)
     select version(); (for server version or if inside database)

2. Architecture Fundamentals:

3. Creating or deleting a database:
   - createdb filenamedb (http://www.postgresql.org/docs/9.1/static/tutorial-createdb.html)
   - su – postgres createdb
   - To delete database: dropdb filenamedb

the postgres user exist in your OS?
with root account try

```
#su - postgres
```

then use the command creatdb

   -
   -

4. Accessing database:
   - psql filenamedb
     - You will be greeted with mydb=> **OR**
     - You will be greeted with mydb=# (for superuser)

4. In the database:
   - \h for help
   - \q to exit databases
   - select version(); (for server version)

FASTX-Toolkit

The FASTX-Toolkit is a collection of command line tools for Short-Reads FASTA/FASTQ files preprocessing.

Next-Generation sequencing machines usually produce FASTA or FASTQ files, containing multiple short-reads sequences (possibly with quality information).

The main processing of such FASTA/FASTQ files is mapping (aka aligning) the sequences to reference genomes or other databases using specialized programs. Example of such mapping programs are: Blat, SHRiMP, LastZ, MAQ and many many others.

However, it is sometimes more productive to preprocess the FASTA/FASTQ files before mapping the sequences to the genome - manipulating the sequences to produce better mapping results.

The FASTX-Toolkit tools perform some of these preprocessing tasks.

Available Tools

- FASTQ-to-FASTA converter: Convert FASTQ files to FASTA files.
- FASTQ Information: Chart Quality Statistics and Nucleotide Distribution
    - FASTQ/A Quality Statistics
    - FASTQ Quality chart
    - FASTQ/A Nucleotide Distribution chart
- FASTQ/A Collapser: Collapse identical sequences in a FASTQ/A file into a single sequence (while maintaining reads counts)
- FASTQ/A Trimmer: Shorten reads in a FASTQ or FASTQ files (remove barcodes or noise).
- FASTQ/A Renamer: Renames the sequence identifiers in FASTQ/A file.
- FASTQ/A Clipper: Removing sequencing adapters / linkers
- FASTQ/A Reverse-Complement: Produce reverse-complement of each sequence in a FASTQ/FASTA file.
- FASTQ/A Barcode splitter: Splits a FASTQ/FASTA files containning multiple samples
- FASTA Formatter: Changes the width of sequences line in a FASTA file
- FASTA Nucleotide Changer: Converts FASTA sequences from/to RNA/DNA
- FASTQ Quality Filter: Filters sequences based on quality
- FASTQ Quality Trimmer: Trims (cuts) sequences based on quality
- FASTQ Masker: Masks nucleotides with 'N' (or other character) based on quality
    - Example: FASTQ Information
    - Example: FASTQ/A manipulation

These tools can be used in two forms:

1. Web-based (with Galaxy, http://main.g2.bx.psu.edu/) : Galaxy's Test website (http://test.g2.bx.psu.edu/) already contains some of the FASTX-toolkit tools.
2. Command-line: Running the tools from command line (or as part of a script).

Command Line Arguments

- Most tools show usage information with **-h**.
- Tools can read from STDIN and write to STDOUT, or
  from a specific input file (**-i**) and specific output file (**-o**).
- Tools can operate silently (producing no output if everything was OK), or
  print a short summary (**-v**).
  If output goes to STDOUT, the summary will be printed to STDERR.
  If output goes to a file, the summary will be printed to STDOUT.
- Some tools can compress the output with GZIP (**-z**).

FASTQ-to-FASTA

$ **fastq_to_fasta -h**
usage: fastq_to_fasta [-h] [-r] [-n] [-v] [-z] [-i INFILE] [-o OUTFILE]

version 0.0.6
  [-h] = This helpful help screen.
  [-r] = Rename sequence identifiers to numbers.
  [-n] = keep sequences with unknown (N) nucleotides.
          Default is to discard such sequences.
  [-v] = Verbose - report number of sequences.
          If [-o] is specified, report will be printed to STDOUT.
          If [-o] is not specified (and output goes to STDOUT),
          report will be printed to STDERR.
  [-z] = Compress output with GZIP.
  [-i INFILE] = FASTA/Q input file. default is STDIN.
  [-o OUTFILE] = FASTA output file. default is STDOUT.

FASTX Statistics

$ **fastx_quality_stats -h**
usage: fastx_quality_stats [-h] [-i INFILE] [-o OUTFILE]

version 0.0.6 (C) 2008 by Assaf Gordon (gordon@cshl.edu)
  [-h] = This helpful help screen.
  [-i INFILE] = FASTA/Q input file. default is STDIN.
          If FASTA file is given, only nucleotides
          distribution is calculated (there's no quality info).
  [-o OUTFILE] = TEXT output file. default is STDOUT.

The output TEXT file will have the following fields (one row per column):
    column = column number (1 to 36 for a 36-cycles read solexa file)
    count = number of bases found in this column.
    min = Lowest quality score value found in this column.
    max = Highest quality score value found in this column.
    sum = Sum of quality score values for this column.
    mean = Mean quality score value for this column.
    Q1 = 1st quartile quality score.
    med = Median quality score.
    Q3 = 3rd quartile quality score.
    IQR = Inter-Quartile range (Q3-Q1).
    lW = 'Left-Whisker' value (for boxplotting).
    rW = 'Right-Whisker' value (for boxplotting).
    A_Count = Count of 'A' nucleotides found in this column.
    C_Count = Count of 'C' nucleotides found in this column.
    G_Count = Count of 'G' nucleotides found in this column.
    T_Count = Count of 'T' nucleotides found in this column.
    N_Count = Count of 'N' nucleotides found in this column.
    max-count = max. number of bases (in all cycles)

FASTQ Quality Chart

$ **fastq_quality_boxplot_graph.sh -h**
Solexa-Quality BoxPlot plotter
Generates a solexa quality score box-plot graph

Usage: /usr/local/bin/fastq_quality_boxplot_graph.sh [-i INPUT.TXT] [-t TITLE] [-p] [-o OUTPUT]

[-p] - Generate PostScript (.PS) file. Default is PNG image.
[-i INPUT.TXT] - Input file. Should be the output of "solexa_quality_statistics" program.
[-o OUTPUT] - Output file name. default is STDOUT.
[-t TITLE] - Title (usually the solexa file name) - will be plotted on the graph.

FASTA/Q Nucleotide Distribution

$ **fastx_nucleotide_distribution_graph.sh -h**
FASTA/Q Nucleotide Distribution Plotter

Usage: /usr/local/bin/fastx_nucleotide_distribution_graph.sh [-i INPUT.TXT] [-t TITLE] [-p] [-o OUTPUT]

[-p] - Generate PostScript (.PS) file. Default is PNG image.
[-i INPUT.TXT] - Input file. Should be the output of "fastx_quality_statistics" program.
[-o OUTPUT] - Output file name. default is STDOUT.
[-t TITLE] - Title - will be plotted on the graph.

FASTA/Q Clipper

$ **fastx_clipper -h**
usage: fastx_clipper [-h] [-a ADAPTER] [-D] [-l N] [-n] [-d N] [-c] [-C] [-o] [-v] [-z] [-i INFILE] [-o OUTFILE]

version 0.0.6
[-h] = This helpful help screen.
[-a ADAPTER] = ADAPTER string. default is CCTTAAGG (dummy adapter).
[-l N] = discard sequences shorter than N nucleotides. default is 5.
[-d N] = Keep the adapter and N bases after it.
(using '-d 0' is the same as not using '-d' at all. which is the default).
[-c] = Discard non-clipped sequences (i.e. - keep only sequences which contained the adapter).
[-C] = Discard clipped sequences (i.e. - keep only sequences which did not contained the adapter).
[-k] = Report Adapter-Only sequences.
[-n] = keep sequences with unknown (N) nucleotides. default is to discard such sequences.
[-v] = Verbose - report number of sequences.
If [-o] is specified, report will be printed to STDOUT.
If [-o] is not specified (and output goes to STDOUT),
report will be printed to STDERR.
[-z] = Compress output with GZIP.
[-D] = DEBUG output.
[-i INFILE] = FASTA/Q input file. default is STDIN.
[-o OUTFILE] = FASTA/Q output file. default is STDOUT.

FASTA/Q Renamer

$ **fastx_renamer -h**
usage: fastx_renamer [-n TYPE] [-h] [-z] [-v] [-i INFILE] [-o OUTFILE]
Part of FASTX Toolkit 0.0.10 by A. Gordon (gordon@cshl.edu)

[-n TYPE] = rename type:
SEQ - use the nucleotides sequence as the name.
COUNT - use simply counter as the name.
[-h] = This helpful help screen.
[-z] = Compress output with GZIP.
[-i INFILE] = FASTA/Q input file. default is STDIN.
[-o OUTFILE] = FASTA/Q output file. default is STDOUT.

FASTA/Q Trimmer

$ **fastx_trimmer -h**
usage: fastx_trimmer [-h] [-f N] [-l N] [-z] [-v] [-i INFILE] [-o OUTFILE]

version 0.0.6
  [-h] = This helpful help screen.
  [-f N] = First base to keep. Default is 1 (=first base).
  [-l N] = Last base to keep. Default is entire read.
  [-z] = Compress output with GZIP.
  [-i INFILE] = FASTA/Q input file. default is STDIN.
  [-o OUTFILE] = FASTA/Q output file. default is STDOUT.

FASTA/Q Collapser

$ **fastx_collapser -h**
usage: fastx_collapser [-h] [-v] [-i INFILE] [-o OUTFILE]

version 0.0.6
  [-h] = This helpful help screen.
  [-v] = verbose: print short summary of input/output counts
  [-i INFILE] = FASTA/Q input file. default is STDIN.
  [-o OUTFILE] = FASTA/Q output file. default is STDOUT.

FASTQ/A Artifacts Filter

$ **fastx_artifacts_filter -h**
usage: fastq_artifacts_filter [-h] [-v] [-z] [-i INFILE] [-o OUTFILE]

version 0.0.6
  [-h] = This helpful help screen.
  [-i INFILE] = FASTA/Q input file. default is STDIN.
  [-o OUTFILE] = FASTA/Q output file. default is STDOUT.
  [-z] = Compress output with GZIP.
  [-v] = Verbose - report number of processed reads.
    If [-o] is specified, report will be printed to STDOUT.
    If [-o] is not specified (and output goes to STDOUT),
    report will be printed to STDERR.

FASTQ Quality Filter

$ **fastq_quality_filter -h**
usage: fastq_quality_filter [-h] [-v] [-q N] [-p N] [-z] [-i INFILE] [-o OUTFILE]

version 0.0.6
  [-h] = This helpful help screen.
  [-q N] = Minimum quality score to keep.
  [-p N] = Minimum percent of bases that must have [-q] quality.
  [-z] = Compress output with GZIP.
  [-i INFILE] = FASTA/Q input file. default is STDIN.
  [-o OUTFILE] = FASTA/Q output file. default is STDOUT.
  [-v] = Verbose - report number of sequences.
    If [-o] is specified, report will be printed to STDOUT.
    If [-o] is not specified (and output goes to STDOUT),
    report will be printed to STDERR.

FASTQ/A Reverse Complement

$ **fastx_reverse_complement -h**
usage: fastx_reverse_complement [-h] [-r] [-z] [-v] [-i INFILE] [-o OUTFILE]

version 0.0.6
  [-h] = This helpful help screen.

[-z] = Compress output with GZIP.
[-i INFILE] = FASTA/Q input file. default is STDIN.
[-o OUTFILE] = FASTA/Q output file. default is STDOUT.

FASTA Formatter

$ **fasta_formatter -h**
usage: fasta_formatter [-h] [-i INFILE] [-o OUTFILE] [-w N] [-t] [-e]
Part of FASTX Toolkit 0.0.7 by gordon@cshl.edu

[-h] = This helpful help screen.
[-i INFILE] = FASTA/Q input file. default is STDIN.
[-o OUTFILE] = FASTA/Q output file. default is STDOUT.
[-w N] = max. sequence line width for output FASTA file.
When ZERO (the default), sequence lines will NOT be wrapped -
all nucleotides of each sequences will appear on a single
line (good for scripting).
[-t] = Output tabulated format (instead of FASTA format).
Sequence-Identifiers will be on first column,
Nucleotides will appear on second column (as single line).
[-e] = Output empty sequences (default is to discard them).
Empty sequences are ones who have only a sequence identifier,
but not actual nucleotides.

Input Example:
>MY-ID
AAAAAGGGGG
CCCCCTTTTT
AGCTN

Output example with unlimited line width [-w 0]:
>MY-ID
AAAAAGGGGGCCCCCTTTTTAGCTN

Output example with max. line width=7 [-w 7]:
>MY-ID
AAAAAGG
GGGTTTT
TCCCCCA
GCTN

Output example with tabular output [-t]:
MY-ID AAAAAGGGGGCCCCCTTTTAGCTN

example of empty sequence:
(will be discarded unless [-e] is used)
>REGULAR-SEQUENCE-1
AAAGGGTTTCCC
>EMPTY-SEQUENCE
>REGULAR-SEQUENCE-2
AAGTAGTAGTAGTAGT
GTATTTTATAT

FASTA Nucleotides Changer

$ **fasta_nucleotide_changer -h**
usage: fasta_nucleotide_changer [-h] [-z] [-v] [-i INFILE] [-o OUTFILE] [-r] [-d]

version 0.0.7
[-h] = This helpful help screen.
[-z] = Compress output with GZIP.
[-v] = Verbose mode. Prints a short summary.
with [-o], summary is printed to STDOUT.

Otherwise, summary is printed to STDERR.
[-i INFILE] = FASTA/Q input file. default is STDIN.
[-o OUTFILE] = FASTA/Q output file. default is STDOUT.
[-r] = DNA-to-RNA mode - change T's into U's.
[-d] = RNA-to-DNA mode - change U's into T's.

FASTA Clipping Histogram

$ **fasta_clipping_histogram.pl**
Create a Linker Clipping Information Histogram

usage: fasta_clipping_histogram.pl INPUT_FILE.FA OUTPUT_FILE.PNG

INPUT_FILE.FA = input file (in FASTA format, can be GZIPped)
OUTPUT_FILE.PNG = histogram image

FASTX Barcode Splitter

$ **fastx_barcode_splitter.pl**
Barcode Splitter, by Assaf Gordon (gordon@cshl.edu), 11sep2008

This program reads FASTA/FASTQ file and splits it into several smaller files,
Based on barcode matching.
FASTA/FASTQ data is read from STDIN (format is auto-detected.)
Output files will be writen to disk.
Summary will be printed to STDOUT.

usage: /usr/local/bin/fastx_barcode_splitter.pl --bcfile FILE --prefix PREFIX [--suffix SUFFIX] [--bol|--eol]
[--mismatches N] [--exact] [--partial N] [--help] [--quiet] [--debug]

Arguments:

--bcfile FILE - Barcodes file name. (see explanation below.)
--prefix PREFIX - File prefix. will be added to the output files. Can be used
to specify output directories.
--suffix SUFFIX - File suffix (optional). Can be used to specify file
extensions.
--bol - Try to match barcodes at the BEGINNING of sequences.
(What biologists would call the 5' end, and programmers
would call index 0.)
--eol - Try to match barcodes at the END of sequences.
(What biologists would call the 3' end, and programmers
would call the end of the string.)
NOTE: one of --bol, --eol must be specified, but not both.
--mismatches N - Max. number of mismatches allowed. default is 1.
--exact - Same as '--mismatches 0'. If both --exact and --mismatches
are specified, '--exact' takes precedence.
--partial N - Allow partial overlap of barcodes. (see explanation below.)
(Default is not partial matching)
--quiet - Don't print counts and summary at the end of the run.
(Default is to print.)
--debug - Print lots of useless debug information to STDERR.
--help - This helpful help screen.

Example (Assuming 's_2_100.txt' is a FASTQ file, 'mybarcodes.txt' is
the barcodes file):

$ cat s_2_100.txt | /usr/local/bin/fastx_barcode_splitter.pl --bcfile mybarcodes.txt --bol --mismatches 2 \
--prefix /tmp/bla_ --suffix ".txt"

Barcode file format

Barcode files are simple text files. Each line should contain an identifier

(descriptive name for the barcode), and the barcode itself (A/C/G/T),
separated by a TAB character. Example:

```
#This line is a comment (starts with a 'number' sign)
BC1 GATCT
BC2 ATCGT
BC3 GTGAT
BC4 TGTCT
```

For each barcode, a new FASTQ file will be created (with the barcode's
identifier as part of the file name). Sequences matching the barcode
will be stored in the appropriate file.

Running the above example (assuming "mybarcodes.txt" contains the above
barcodes), will create the following files:

```
/tmp/bla_BC1.txt
/tmp/bla_BC2.txt
/tmp/bla_BC3.txt
/tmp/bla_BC4.txt
/tmp/bla_unmatched.txt
```

The 'unmatched' file will contain all sequences that didn't match any barcode.

Barcode matching

** Without partial matching:

Count mismatches between the FASTA/Q sequences and the barcodes.
The barcode which matched with the lowest mismatches count (providing the
count is small or equal to '--mismatches N') 'gets' the sequences.

Example (using the above barcodes):
Input Sequence:

```
GATTTACTATGTAAAGATAGAAGGAATAAGGTGAAG
```

Matching with '--bol --mismatches 1':

```
GATTTACTATGTAAAGATAGAAGGAATAAGGTGAAG
GATCT (1 mismatch, BC1)
ATCGT (4 mismatches, BC2)
GTGAT (3 mismatches, BC3)
TGTCT (3 mismatches, BC4)
```

This sequence will be classified as 'BC1' (it has the lowest mismatch count).
If '--exact' or '--mismatches 0' were specified, this sequence would be
classified as 'unmatched' (because, although BC1 had the lowest mismatch count,
it is above the maximum allowed mismatches).

Matching with '--eol' (end of line) does the same, but from the other side
of the sequence.

** With partial matching (very similar to indels):

Same as above, with the following addition: barcodes are also checked for
partial overlap (number of allowed non-overlapping bases is '--partial N').

Example:
Input sequence is ATTTACTATGTAAAGATAGAAGGAATAAGGTGAAG
(Same as above, but note the missing 'G' at the beginning.)

Matching (without partial overlapping) against BC1 yields 4 mismatches:

```
ATTTACTATGTAAAGATAGAAGGAATAAGGTGAAG
GATCT (4 mismatches)
```

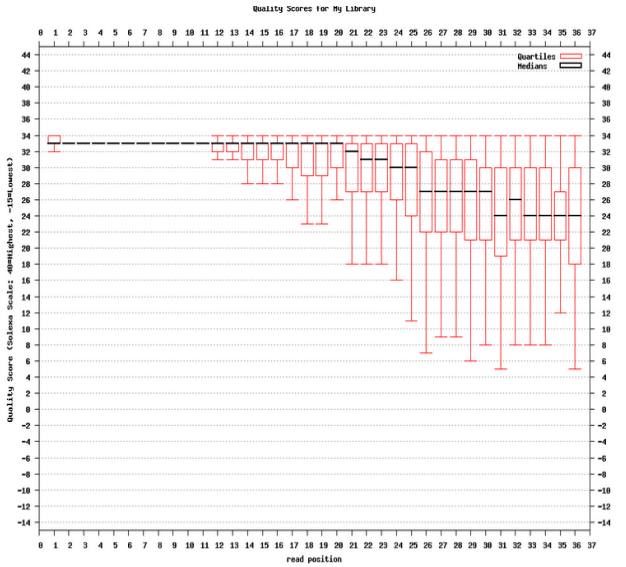
Partial overlapping would also try the following match:
-ATTTACTATGTAAAGATAGAAGGAATAAGGTGAAG
GATCT (1 mismatch)

Note: scoring counts a missing base as a mismatch, so the final
mismatch count is 2 (1 'real' mismatch, 1 'missing base' mismatch).
If running with '--mismatches 2' (meaning allowing upto 2 mismatches) - this
seqeunce will be classified as BC1.

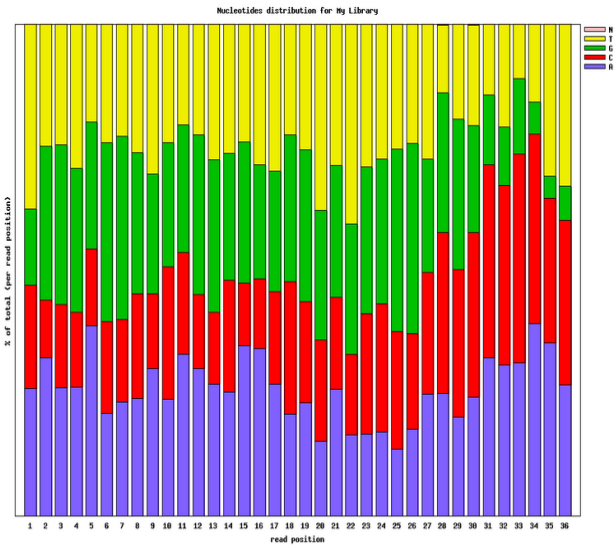
Example: FASTQ Information

Genrating Quality Information on **BC54.fq**:

$ fastx_quality_stats -i BC54.fq -o bc54_stats.txt
$ fastq_quality_boxplot_graph.sh -i bc54_stats.txt -o **bc54_quality.png** -t "My Library"
$ fastx_nucleotide_distribution_graph.sh -i bc54_stats.txt -o **bc54_nuc.png** -t "My Library"



bc54_quality.png



bc54_nuc.png

Manipulation

Common pre-processing work-flow:

1. Covnerting FASTQ to FASTA
2. Clipping the Adapter/Linker
3. Trimming to 27nt (if you're analyzing miRNAs, for example)
4. Collapsing the sequences
5. Plotting the clipping results

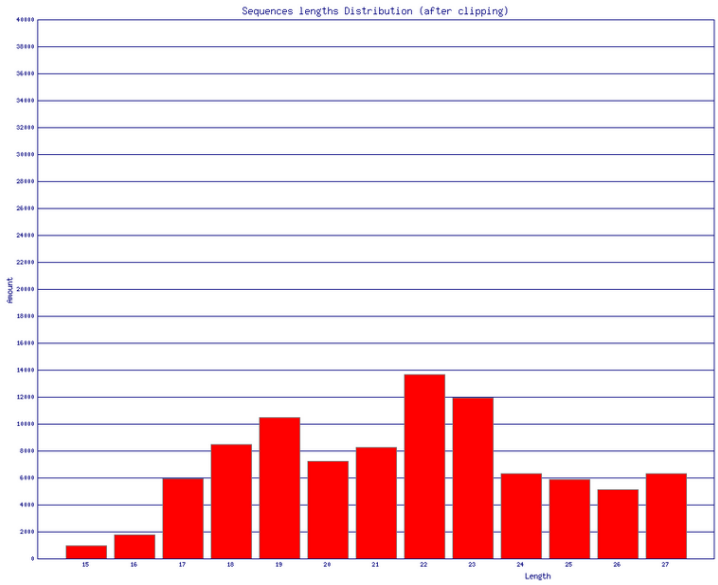
Using the FASTX-toolkit from the command line:

$ **fastq_to_fasta -v -n -i BC54.fq -o BC54.fa**
Input: 100000 reads.
Output: 100000 reads.
$ **fastx_clipper -v -i BC54.fa -a CTGTAGGCACCATCAATTCGTA -o BC54.clipped.fa**
Clipping Adapter: CTGTAGGCACCATCAATTCGTA
Min. Length: 15
Input: 100000 reads.
Output: 92533 reads.
discarded 468 too-short reads.
discarded 6939 adapter-only reads.
discarded 60 N reads.
$ **fastx_trimmer -v -f 1 -l 27 -i BC54.clipped.fa -o BC54.trimmed.fa**
Trimming: base 1 to 27
Input: 92533 reads.

Output: 92533 reads.
$ **fastx_collapser -v -i BC54.trimmed.fa -o BC54.collapsed.fa**
Collapsd 92533 reads into 36431 unique sequences.
$ **fasta_clipping_histogram.pl BC54.collapsed.fa bc54_clipping.png**

(alternatively, run it all together with shell pipes)

$ **cat BC54.fq | fastq_to_fasta -n | fastx_clipper -l 15 -a CTGTAGGCACCATCAATTCGTA | **
fastx_trimmer -f 1 -l 27 | fastx_collapser > bc54.final.fa



bc54_clipping.png

Mapping (or any other kind of analysis) of the Clipped + Collapsed FASTA file will be:

- **quicker** - each unique sequence appears only once in the FASTA file.
- **more accurate** - the Adapter/Linker sequence was removed from the 3' end, and will affect the mapping results.

Preliminary quality control of NGS data

Table of contents

Expected learning outcomes

Next-generation sequencing technologies are becoming widely used and, although a massive number of sequences can be generated in a single experiment, it is still very important to have a direct look at raw data. Performing sanity checks at the read level allows avoiding undesirable outcomes in the assembly or mapping processes. Discarding low-quality reads, controlling for contamination or trimming adaptor sequences are examples of preliminary quality control filters that should be applied to raw reads before further analysis.

The learning objective of this activity is to explore some relevant properties of an ensemble of next generation sequencing reads such as length, quality scores and base distribution in order to assess the quality of the data and to discard low quality reads.

For that, we will use basic UNIX commands and the FASTX-Toolkit applied to two small 454 and Illumina datasets.

Getting started

1. Make sure you have the following programs installed (if you used the customized USB flash drive for software installation, you already have them):
    a. The FASTX-Toolkit
    b. Perl and Bioperl
    c. R
    d. Gnuplot
    e. If you are running the WCG Ubuntu Linux distribution, please run the following command before beginning this activity:

    ```
    sudo chmod +x /usr/local/bin/*.pl
    ```

2. The data you are going to use are located on the USB flash drive, in the Activities/QC_of_NGS_data/ folder. Go to that folder and copy the content to your computer. You should find:
    a. **BBb.fastq**: a small subset of a Illumina run (v.1.5) in a fastq file.
    b. **GCJ_10k.fasta** and **GCJ_10k.qual**: sequences and qualities for small subset of a 454 Titanium run.
    c. **plotLengthDistribution.R**: a simple R script to plot the distribution of read lengths, taking as input a two column tab file where the second column is the length of the sequences
    d. **fastx_nucleotide_distribution.R**: an R script to plot the distribution of nucleotides along the read position (similar to the FASTX-Toolkit tool fastx_nucleotide_distribution.sh, uses

the same input file)
    e. **fastx_quality_boxplot.R**: an R scipt to plot the quality of nucleotides along the read position (similar to the FASTX-Toolkit tool fastx_quality_boxplot.sh, uses the same input file)
3. The Perl scripts that are going to be used in this activity have been installed in /usr/local/bin, if you are interested to look at them:
    a. **fastaNamesSizes.pl**: takes a sequence format as input as outputs the name and the length each sequence. Requires BioPerl
    b. **fastaQual2fastq.pl**: takes a fasta and a qual file and outputs a fastq file. Requires BioPerl
4. Have a look at the documentation for FASTX-Toolkit. Make yourself an idea of the different tools and what they do. In general, you can get help on each program by typing program_name -h

Exercise 1: checking Illumina data

The goal of this exercise is to inspect the content of the data resulting from an Illumina run. Although fastq is the main file received from a sequence provider, some users want to perform the base calling step themselves, using a different package than the proprietary Illumina software. This is not covererd in this exercise, and we start directly from the fastq file.

1. Inspect the data contained in BBb.fastq. Use your favorite text editor or viewer. For example, with less:

   ```
   less BBb.fastq
   ```

2. Using the Perl script fastaNamesSizes.pl, count the length of each sequence in the fastq file:

   ```
   fastaNamesSizes.pl -f fastq-illumina BBb.fastq > BBb.ns
   ```

   Have a look at the numbers output on the terminal. How many sequences do we have? Inspect the output (less BBb.ns).

3. Open the Perl script in a text editor, and inspect it.
4. Convert the fastq file into fasta format using fastq_to_fasta:

   ```
   fastq_to_fasta -n < BBb.fastq > BBb.fasta
   ```

5. Compute quality statistics for the BBb dataset, using fastx_quality_stats

   ```
   fastx_quality_stats -i BBb.fastq > BBb.qualstats
   ```

6. View the statistics file with a text editor. What do the columns represent? (hint: the help file of the program might... help)
7. (optional): produce a more extensive output using the -N flag. View it. What are the extra columns?
8. Draw a box-plot of the reads quality with fastq_quality_boxplot_graph.sh:

   ```
   fastq_quality_boxplot_graph.sh -i BBb.qualstats -o
   BBb.qualstats.png
   ```

   (alternatively with R) `Rscript fastx_quality_boxplot.R BBb.qualstats BBb.qualstats2.png`
9. Open the resulting png file (with open or Preview on a Mac, with Firefox or eog on Ubuntu). What do x and y axis represent? What do the boxes represent? What can you say about the quality in general? What's the average probability of error? How is the quality varying along the read?
10. Draw the distribution of nucleotides with fastx_nucleotide_distribution_graph.sh:

    ```
    fastx_nucleotide_distribution_graph.sh -i BBb.qualstats -o
    BBb.nucdistr.png
    ```

    (alternatively with R) `Rscript fastx_nucleotide_distribution.R`

```
BBb.qualstats BBb.nucdistr2.png
```

11. Open the resulting png file with your favorite graphical program. What can you say about the distribution of nucleotides? Any estimation of the G+C content?

Exercise 2: checking 454 data

The goal of this exercise is to do the same kind of quality checks as in exercise 1, but on 454 data this time. The primary data from 454 is stored in a sff file, but in general, the sequence provider also provides fasta and qual files. It is possible to parse the sff files with different parameters using the proprietary software provided by 454 (sfffile/sffinfo) or a free, open-source tool, sff_extract.

This step is not covered in this exercise, and we start from the GCJ_10k.fasta and GCJ_10k.qual files.

1. Use your favorite text editor to visualize both fasta and qual files.
   [show answer]

   Answer: `less GCJ_10k.fasta`
   `less GCJ_10k.qual`
2. Some functions of FASTX-Toolkit don't like fasta sequences on multiple lines, so let's transform the fasta format with fasta_formatter, outputting each sequence on a single line.
   [show answer]

   Answer: `fasta_formatter < GCJ_10k.fasta > GCJ_10k_1.fasta`
3. Some FASTX-Toolkit programs can only take fastq as an input, let's convert fasta and qual files to a fastq file, using a short Bioperl script:

   ```
   fastaQual2fastq.pl GCJ_10k.fasta GCJ_10k.qual > GCJ_10k.fastq
   ```
4. Inspect the resulting fastq file with your favorite text editor.
5. As previously for the Illumina dataset, calculate the length of each sequence using the fastaNamesSizes.pl script.
   [show answer]

   Answer: `fastaNamesSizes.pl GCJ_10k.fasta > GCJ_10k.ns`
6. Explore the output with a text editor. What would you say about the statistics on length and number of sequences?
7. (optional) Use a combination of grep, cut and wc commands to perform the same output.
8. Plot the distribution of read lengths using a short R script:

   ```
   Rscript plotLengthDistribution.R GCJ_10k.ns GCJ_10k_readDistr.png
   ```
9. (optional) Have a look at the script (using a text editor) to see what it's doing.
10. Alternatively, use the FASTX-Toolkit fasta_clipping_histogram.pl to plot the read length distribution (but not as nicely) (this script might not work on all distributions. If it returns an error, don't insist)
    [show answer]

    Answer: `fasta_clipping_histogram.pl GCJ_10k_1.fasta`
    `GCJ_10k_readDistr2.png`
11. Now compare both plots, GCJ_10k_readDistr.png and GCJ_10k_readDistr2.png. How do they look like? Can you say something about the distribution? What is the mean length? the median?
12. (optional) Modify the R script to display vertical bars of different colors at the mean and median read lenght. Hint: use abline(v=...) command in the R script.
13. Now compute the quality statistics for this subset and look at them in a text editor. Use fastx_quality_stats and view the results in a text editor. Use also the -N switch to produce extensive output.
    [show answer]

Answer: `fastx_quality_stats -i GCJ_10k.fastq > GCJ_10k.qualstats`
`fastx_quality_stats -i GCJ_10k.fastq -N > GCJ_10k.extqualstats`

14. Now plot the quality distribution along the sequence with fastq_quality_boxplot_graph.sh and view the resulting png plot (alternatively, use the R script). What do you see?
[show answer]

    Answer: `fastq_quality_boxplot_graph.sh -i GCJ_10k.qualstats -o GCJ_10k.qualstats.png`

15. Draw the distribution of nucleotides with fastx_nucleotide_distribution_graph.sh (or with the Rscript) and view the result. What do you see?
[show answer]

    Answer: `fastx_nucleotide_distribution_graph.sh -i GCJ_10k.qualstats -o GCJ_10k.nucdistr.png`

16. We need to trim that down. Use head to take only the first lines of the stats file and output it to another file, and redraw the figure. What do you see now? What's your conclusion about this dataset? What should be done about it?
[show answer]

    Answer: `head -n 101 GCJ_10k.qualstats > GCJ_10k.100f.qualstats`
    `fastx_nucleotide_distribution_graph.sh -i GCJ_10k.100f.qualstats -o GCJ_10k.100f.nucdistr.png`

Exercise 3: fixing extra adaptor sequence

Obviously, most of the sequences still have some adaptor sequences left at their 5' end. We need to remove them to avoid further problems with the assembly or mapping. We'll use two different tools, that have a different behavior, fastx_clipper and fastx_trimmer

1. **fastx_clipper** takes a fastq file and searches for a given adaptor sequence at the 3' end of the sequence (common in Illumina datasets). Since we want to remove the adaptor sequence at the 5' end in our 454 set, we will first reverse-complement the sequences with fastx_reverse_complement, search for the reverse-complemented sequence of the adaptor with fastx_clipper and reverse-complement again
[show answer]

    Answer: It is possible to use multiple commands and files, but an elegant way to do it is to use pipes ("|") to link the output of one program to the input of the next one: `cat GCJ_10k.fastq | fastx_reverse_complement | fastx_clipper -a CTCGCGATAT -n -v -l 50 | fastx_reverse_complement > GCJ_10k.clip.fastq`

2. Then recalculate the nucleotide distribution stats and draw the figure. How does it look now?
[show answer]

    Answer: `fastx_quality_stats -i GCJ_10k.clip.fastq | head -n 100 > GCJ_10k.clip.qualstats`
    `fastx_nucleotide_distribution_graph.sh -i GCJ_10k.clip.qualstats -o GCJ_10k.clip.nucdistr.png`

3. **fastx_trimmer**, alternatively, trims a given number of nucleotides, either from the beginning or from the end of the sequence. We can also use it, even though we might lose some non-adaptor sequences in some of the reads.
[show answer]

    Answer: `fastx_trimmer -f 11 -m 50 -i GCJ_10k.fastq -o GCJ_10k.trim.fastq`

4. Then again, recalculate the stats and redraw the figure. Compare the results of both runs, can you tell the difference?
   [show answer]

   Answer: `fastx_quality_stats -i GCJ_10k.trim.fastq | head -n 100 > GCJ_10k.trim.qualstats`
   `fastx_nucleotide_distribution_graph.sh -i GCJ_10k.trim.qualstats -o GCJ_10k.trim.nucdistr.png`
5. (optional) Recalculate the distribution of length sizes after clipping or trimming, and/or compare the lengths before and afer clipping/trimming.

See also

Other tools to verfiy quality of second-generation sequencing results are available:

- Galaxy, a web-based genomics pipeline, in which FASTX-Toolkit is integrated.
- Perl and Bioperl, to write small scripts. There already exist a very large number of packages devoted to genomics in Bioperl.
- R and Bioconductor, another solution to import and verify data. Many packages already exist.

Navigation

- GARLI Web Service - create job
- GARLI Web Service - view job status

Maintained by Adam Bazinet | Direct questions and comments to Michael P. Cummings |

TS software from Ion Torrent
The sofware is installed in /opt

cd /opt
sudo git clone https://github.com/iontorrent/TS.git

convert_project

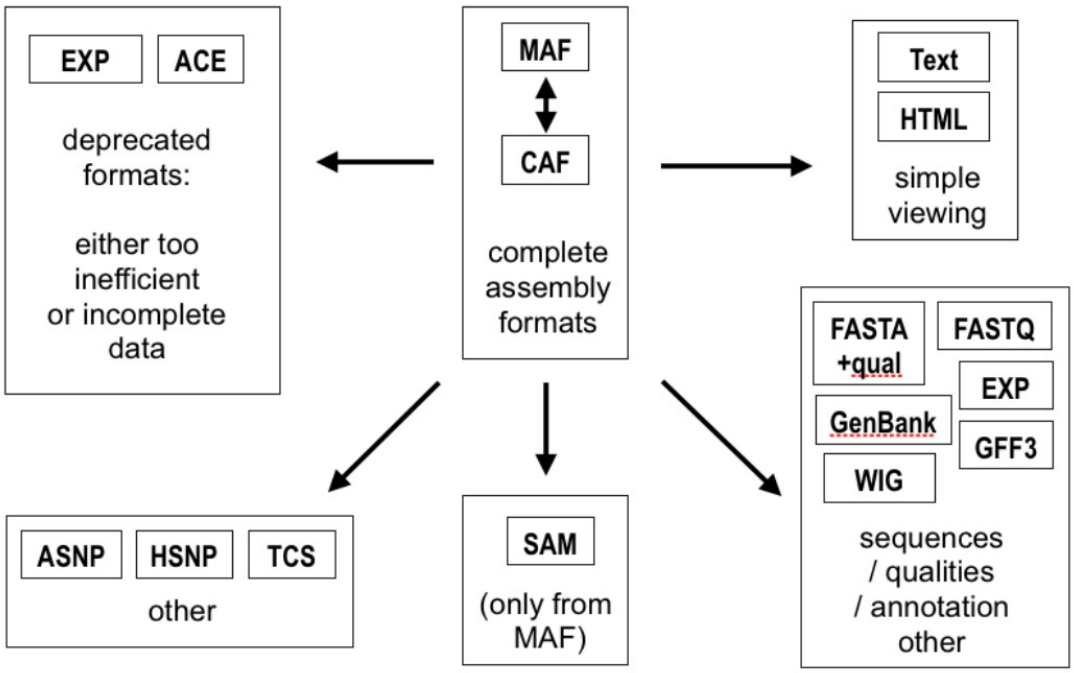


Figure 9.1: **convert_project** supports a wide range of format conversions to simplify export / import of results to and from other programs

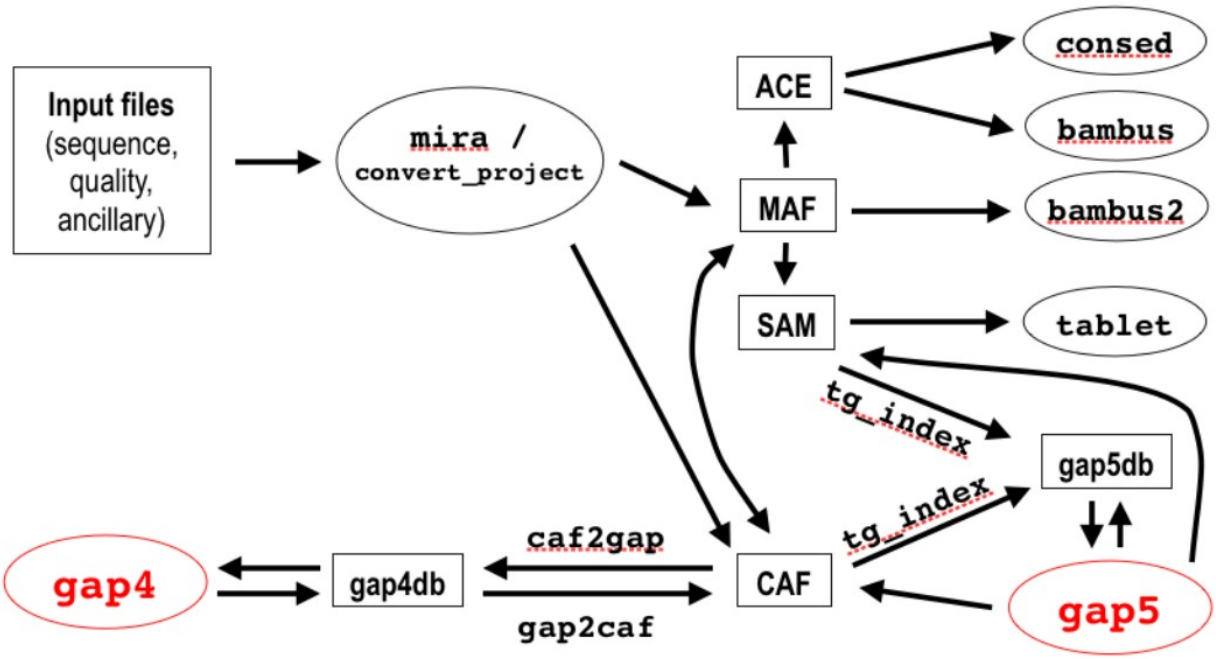


Figure 9.2: Conversion steps, formats and programs needed to reach some tools like assembly viewers, editors or scaffolders.

Usage:

convert_project -h (for help, copied below)

convert_project [-f <fromtype>] [-t <totype> [-t <totype> ...]]
[-aChimMsuZ]
[-AcflnNoqrtvxXyz {...}]
{infile} {outfile} [<totype> <totype> ...]

Options:

-f <fromtype> load this type of project files, where fromtype is:
caf a complete assembly or single sequences from CAF
maf a complete assembly or single sequences from CAF
fasta sequences from a FASTA file
fastq sequences from a FASTQ file
gbf sequences from a GBF file
phd sequences from a PHD file
fofnexp sequences in EXP files from file of filenames

-t <totype> write the sequences/assembly to this type (multiple mentions of -t are allowed):
ace sequences or complete assembly to ACE
caf sequences or complete assembly to CAF
maf sequences or complete assembly to MAF
sam complete assembly to SAM
samnbb like above, but leaving out reference (backbones) in mapping assemblies
gbf sequences or consensus to GBF
gff3 consensus to GFF3
wig assembly coverage info to wiggle file
gcwig assembly gc content info to wiggle file
fasta sequences or consensus to FASTA file (qualities to .qual)
fastq sequences or consensus to FASTQ file
exp sequences or complete assembly to EXP files in directories. Complete assemblies are suited for gap4 import as directed assembly. Note: using caf2gap to import into gap4 is recommended though
text complete assembly to text alignment (only when -f is caf, maf or gbf)
html complete assembly to HTML (only when -f is caf, maf or gbf)
tcs complete assembly to tcs
hsnp surrounding of SNP tags (SROc, SAOc, SIOc) to HTML (only when -f is caf, maf or gbf)
asnp analysis of SNP tags (only when -f is caf, maf or gbf)
cstats contig statistics file like from MIRA (only when source contains contigs)
crlist contig read list file like from MIRA (only when source contains contigs)
maskedfasta reads where sequencing vector is masked out (with X) to FASTA file (qualities to .qual)
scaf sequences or complete assembly to single sequences CAF

-a Append to target files instead of rewriting

-A <string> String with MIRA parameters to be parsed
Useful when setting parameters affecting consensus

```
                calling like -CO:mrpg etc.
                E.g.: -a "454_SETTINGS -CO:mrpg=3"
-b              Blind data
                Replaces all bases in reads/contigs with a 'c'
-C              Perform hard clip to reads
                When reading formats which define clipping points, will
                 save only the unclipped part into the result file.
                Applies only to files/formats which do not contain
                 contigs.
-d              Delete gap only columns
                When output is contigs: delete columns that are
                 entirely gaps (like after having deleted reads during
                 editing in gap4 or similar)
                When output is reads: delete gaps in reads
-F              Filter to read groups
                Special use case, do not use yet.
-m              Make contigs (only for -t = caf or maf)
                Encase single reads as contig singlets into the CAF/MAF
                 file.
-n <filename> when given, selects only reads or contigs given by
                 name in that file.
-i              when -n is used, inverts the selection
-o              fastq quality Offset (only for -f = 'fastq')
                Offset of quality values in FASTQ file. Default of 0
                 tries to automatically recognise.
-Q <quality>            Set default quality for bases in file types without quality values
                Furthermore, do not stop if expected quality files are missing (e.g. '.fasta')
-R <name>       Rename contigs/singlets/reads with given name string
                 to which a counter is appended.
                Known bug: will create duplicate names if input
                 contains contigs/singlets as well as free reads, i.e.
                 reads not in contigs nor singlets.
-S <name>       (name)Scheme for renaming reads, important for paired-ends
                Only 'solexa' is currently supported.

-------------------------------------------------------
The following switches work only when input (CAF or MAF)
contains contigs. Beware: CAF and MAf can also contain
just reads.
-------------------------------------------------------

-M              Do not extract contigs (or their consensus), but the
                 sequence of the reads they are composed of.
-N <filename>           like -n, but sorts output according to order given
                in file.
-r [cCqf]       Recalculate consensus and / or consensus quality values
                 and / or SNP feature tags.
                'c' recalc cons & cons qualities (with IUPAC)
                'C' recalc cons & cons qualities (forcing non-IUPAC)
                'q' recalc consensus qualities only
                'f' recalc SNP features
                Note: only the last of cCq is relevant, f works as a
                 switch and can be combined with cQq (e.g. "-r C -r f")
```

Note: if the CAF/MAF contains multiple strains,
recalculation of cons & cons qualities is forced, you
can just influence whether IUPACs are used or not.

-s split output into multiple files instead of creating a
single file

-u 'fillUp strain genomes'
Fill holes in the genome of one strain (N or @)
with sequence from a consensus of other strains
Takes effect only with -r and -t gbf or fasta/q
in FASTA/Q: bases filled up are in lower case
in GBF: bases filled up are in upper case

-q <integer> Defines minimum quality a consensus base of a strain
must have, consensus bases below this will be 'N'
Default: 0
Only used with -r, and -f is caf/maf and -t is (fasta
or gbf)

-v Print version number and exit

-x <integer> Minimum contig or read length
When loading, discard all contigs / reads with a
length less than this value. Default: 0 (=switched off)
Note: not applied to reads in contigs!

-X <integer> Similar to -x but applies only to reads and
then to the clipped length.

-y <integer> Minimum average contig coverage
When loading, discard all contigs with an
average coverage less than this value.
Default: 1

-z <integer> Minimum number of reads in contig
When loading, discard all contigs with a
number of reads less than this value.
Default: 0 (=switched off)

-l <integer> when output as text or HTML: number of bases shown in
one alignment line. Default: 60.

-c <character> when output as text or HTML: character used to pad
endgaps. Default: ' ' (blank)

Aliases:
caf2html, exp2fasta, ... etc. Any combination of "<validfromtype>2<validtotype>"
can be used as program name (also using links) so as that convert_project
automatically sets -f and -t accordingly.

Examples:

```
convert_project source.maf dest.sam
convert_project source.caf dest.fasta wig ace
convert_project -x 2000 -y 10 source.caf dest.caf
caf2html -l 100 -c . source.caf dest
```

Examples for convert_project:
1. convert_project -t fasta -x 5000 -y 9 Aeb_out.maf AEb_X5000Y8.fasta

where

-t fasta (sequence file output format)

-x 5000 (sequence length)
-y 9 (coverage)
Aeb_out.maf (input file name is either maf or caf)
Aeb_X5000Y8.fasta (output file name, the default tag "_AllStrains" is automatically added to the outfile

2. convert_project -f maf -t caf csstats -x 500 -y 8 infilename_is_out.maf outfile_filterdx500y8
where
-f = <fromtype> load this type of project files, where fromtype is a complete assembly or single sequences from caf, maf, fasta, fastq, gbf, phd,
cstats = contig statistics file like from MIRA (only when source contains contigs)
-x = 500 (sequence length)
-y = 9 (coverage)
infilename_is_out.maf = mira infile maf
outfile_filterdx500y8 = filtered statistic file as well as a stats file (...inf_contigstats.txt) will be produced

3. caf2gap
Example 1.

The settings I used were straight from the MIRA manual (with a different project name):
caf2gap -project c227-11 -ace c227-11-clrc_filteredx500y8.caf

The output was a PROJECTNAME.0 file and a PROJECTNAME.0.aux file

Example 2 (from mira manual)

```
miraconvert -x 500 -y 15 sourcefile.maf tmp.caf
$ caf2gap -project somename -ace tmp.caf
```

```
Example 4 mira (version 4)
The following are the 2 steps that are required for mira output files that work with
gap4:
STEP 1. The length (-x) and coverage (-y) of the contigs in the mira outfile .maf
are selected and the file converted to caf with miraconvert as follows.
        miraconvert -x 500 -y 8 Aeb_out.maf Aeb.caf

STEP 2. The Aeb.caf file is converted to the required gap4 format using caf2gap.
Note that the switch used is -ace though the input file is a .caf format
        caf2gap -project Aebgap4 -ace Aeb.caf

NOTE:
Immediately after assembly with mira, I create a new working directory for further
assembly clean up. I would call this something_gap4. I copy the .maf file into the
directory and mira convert followed by caf2gap
```

Example (e.g. useful with Sanger/454 hybrid assemblies): extracting only contigs ≥ 1000 bases and with ≥ 10 reads from MAF into CAF format, then converting the reduced CAF into a Staden GAP4 project:

usage: caf2gap
-project text [ ]
-version text [0]
-expected integer [4000]
-db_version integer [3]
-caf switch [true]
-force switch [false]

-silent switch [false]
-preserve switch [false]
-ace Readable File [ ]
-help switch [ ]

Dealing with repeats

> On Jul 30, 2013, at 6:54 , km <srikrishnamohan@xxxxxxxxx> wrote:
> > I see that mira detects and classifies repeats during assembly.
> > Do I use mirabait to exclude those reads mapping corresponding to
> repeats ? and then repeat the assembly to get a finer assembly ? Is this
> approach correct. I am dealing with plant genome assemblies with lots of
> repeat content.
>
> If you can already load the complete data set into memory and have enough
> RAM, then using -s2.mnr and -SK:nrr would probably be a better approach:
> MIRA would still assemble those reads which are "almost" repetitive but,
> e.g., differ in a single base which makes them non-repetitive.
>
> Else your approach seems a good way to go …. though I'd increase the -n of
> mirabait to something around maybe 10 (if you have Illuminas).

Subject: [mira_talk] Re: Mira for Mixed platform Metagenomic assembly?

On Mar 14, 2013, at 22:14 , raw937@xxxxxxxxx wrote:
> In that I have sequencing data from 454Ti, GaIIx PE76, HiSeq PE100 and MiSeq
> PE250 for some metagenomic samples.
> could I trim the data for quality then pool all the data then use Mira to
> assemble it all even with different lengths of sequence and insert size?

Throw everything into MIRA without trimming (but make sure the 454 data follows the lowercase/uppercase rules for clipping).

MIRA should takeover just fine from there and will clip just as needed. Do NOT try to clip or pre-trim the Illumina data … MIRA has far better clipping algorithms than any other pipeline which either throw away too much or not enough.

Notes:
- assemble in EST mode, not genome mode
- in case there are extremely overrepresented sequences in your sample, MIRA will choke on those. Once identified (read repeats file!), use mirabait to bait them out and assemble the rest.

11.2 mirabait

11.2.1 Synopsis

`mirabait` [options] *`bait_file input_file output_basename`*

While input and output file can have any of the supported formats (see -f and -t options), the bait file needs to be in FASTA format.

11.2.2 Description

mirabait selects reads from a read collection which are partly similar or equal to sequences defined as target baits. Similarity is defined by finding a user-adjustable number of common k-mers (sequences of k consecutive bases) which are the same in the bait sequences and the screened sequences to be selected, either in forward or reverse complement direction.

The search performed is exact, that is, sequences selected are guaranteed to have the required number of k-mers equal to the bait sequences while sequences not selected are guaranteed not have these.

11.2.3 Options

`-f` ***`caf | maf | fasta | fastq | gbf | phd`*** 'From-type', the format of the input file. Default: fastq.

`-t` ***`caf | maf | fasta | fastq`*** 'To-type', the format of the output file. Default: format of the input.
Multiple mentions of -t are allowed, in which case the selected sequences are written to all file formats chosen.

`-k` ***`k-mer-length`*** k-mer, length of bait in bases ($\leq$32, default=31)

`-n` ***`minoccurence`*** Minimum number of k-mers needed for a sequence to be selected. Default: 1.

`-i` Inverse hit: selects only sequence that do not meet the -k and -n criteria.

`-r` Does not check for hits in reverse complement direction.

mirabait:

The following approach is based on grabbing reads with "mirabait" which have a 31 nucleotide stretch identical to something from your MT.

1. MIRA will have cleaned the Solexa reads as well as it could, it is advisable to take these instead of the raw Illumina data. For this, in the directory with your mapping assembly, grab the file *_assembly/*_chkpt/readpool.maf and convert it back to FASTQ, performing a hard clip: convert_project -f maf -t FASTQ -C readpool.maf mynewl8data

2. Grab the result from your mapping and convert the consensus from your strain to FASTA. That is extract your strain specific sequence from the resulting assembly (convert_project -f maf -t fasta -A "SOLEXA_SETTINGS -CO:fnicpst=yes" salarismtlane8_out.maf iteration1)

3. pick the iteration1_salarismtlane8.fasta, that is the current reference

4. each stretch of "X" or "@" in the FASTA above should be replace by a stretch of, say, 300 "N". Do that with a script. Name it whatever you like, I'll use "ref_iter1_backbone_in.fasta"

5. now bait out all reads from the totality of your Illumina reads which seem to match that consensus:
mirabait -k 31 -n 1 ref_iter1_backbone_in.fasta mynewl8data.fastq mymtreads_iteration1.fastq

5a. bonus step: if you have paired end: write a script which looks through "mymtreads_iteration1.fastq" and grabs all pairs from "mynewl8data.fastq" to form a

new
"mymtreads_iteration1.fastq"

6. map the reads "mymtreads_iteration1.fastq" to "ref_iter1_backbone_in.fasta" like you
did initially for all reads. To save some time, you can switch off all MIRA clipping (these reads were
already clipped!) via "--noclipping -CL:pec=no"

7. rinse, repeat, go back to step 2 (Two! not one) until you are satisfied. Satisfied means: the number of reads chosen by mirabait in a given iteration is not significantly higher than in the previous iteration.

The above works because MIRA, in each mapping, extend a bit the ends of the reference into stretches containing Ns with reads mapping to the refernce. The result should be a FASTQ file containing only reads from the mitochondrium.

Those you can then assemble de-novo (or map with very weak alignment parameters) using MIRA or any other assembler.

IMPORTANT NOTE: this approach will fail in cases where parts of the MT DNA have been integrated into the genome. More specifically: any 31mer identical in the MT DNA and genome DNA will break the approach and one will have to resort to more complex filtering.

Mirabait (rRNA):

Step 1 Mirabait: Extract all reads which could possibly match with the 1400bp sequence. Use "mirabait" with a comparatively short kmer and adapted number of kmers needed. This depends on the sequencing technology (you have Ion) and on how close you think your sequencing data matches your 1400bp … for Ion, if it's very close you could use something like k=17 and n=4. If you think they are quite dissimilar, you can go down to something like k=12 and n=12.

Note: the above step may also extract some reads only faintly related to your 1400bp … that's life.

Step 2 Mapping: map the reads you got against your 1400 bp sequence

Step 3 Prepare a fastaq input file: get the names of the reads which mapped, prepare a FASTQ input file of those

Step 4 Assemble de-novo:

Question

How to create consensus phylogenetic tree for sequence clusters?

I have several orthologous sequence clusters. I want to create a consensus phylogenetic tree by exploiting these clusters. The number of sequences per cluster varies from 2-13 and the sequence cluster contains orthologous sequences from 2-8 species/cluster.

Toni Gabaldón · 45.05 · 579.08 · CRG Centre for Genomic Regulation
Since the maximum number of sequences per cluster is higher than the maximum number of species per cluster, i deduce your orthologous clusters contain (in)paralogs. I also assume the clusters are from different families and what you want to infer is the species tree. Let me know otherwise.

One possibility to combine the phylogenetic information from all clusters is to build a "supertree", there are many ways to do that. In all, you first build a tree per orthologous group. Then you combine the trees into a consensus one, you can do this using the program duptree, which uses a gene tree parsimony approach finding the species tree that is imputes the lowest number of duplication in the family tree. It is quite powerful and enables the presence of paralogs in the trees (it actually exploits it).

The presence of paralogs in your tree hampers strategies such as gene concatenation or supertree approaches that do not enable duplications (e.g. the consensus approaches in PHYLIP).... a way out of this is to decompose the trees with duplication in trees containing only orthologs using the treeKO algorithm (http://treeko.cgenomics.org)

Hope it helps.

Brian Foley · 52.92 · 334.74 · Los Alamos National Laboratory
There are two possible meanings of "consensus tree". One is when you compare hundreds of trees and plot the consensus tree as in bootstrapping phylogenetic analysis, and the other is to build a consensus sequence for groups of genes and
then build a phylogenetic tree from the resulting consensus sequences.

There are dozens of tools available for each of those routes. Almost all phylogenetics program packages such as PHYLIP, MEGA, DAMBE etc have tree bootstrapping and consensus built in. BioEdit and many other multiple sequence alignment editing
tools have consensus sequence building built in.
www.mbio.ncsu.edu/bioedit/bioedit.html

The treemaker and PhyML server at the HIV Databases, as well as most multiple sequence alignment servers, allow you to download the multiple sequence alignment in tree order so that the most similar sequences are next to each other in the alignment, making it easy to build a consensus for each group, whether you are interested in one consensus for each homolog (alpha globin, beta globin etc in many mammals) or one consensus for each organism (one alpha-beta-gamma globin consensus for each mammal species).
http://www.hiv.lanl.gov/components/sequence/HIV/treemaker/treemaker.html

David Peris · 13.90 · 11.72 · University of Wisconsin, Madison
Well, following the suggestion of Toni Gabaldón, if you have paralogs maybe some of the species doesn't have the complete set of genes, so my recommendation is to use supernetworks, with a similar meaning that supertrees. The weakness of supernetworks are the no existance, at this point, statistical support but

you can compare with your supertree or individual trees. The strongest of this method is that you can visualize all species in one network and conflicting data.
To do this you can use the software SPLITSTREE 4, DENDROSCOPE, QUARTETNET or PHYLONET.
Good luck!
Peris

Ebru Tekin · 4.99 · 3.28 · Ege University
Hi, there is the upper version of MEGA4 that you can easily use. If you are a new user I recommend you MEGA5 which is much easy than MEGA4.

Titanium Library

The Lib-L stands for Library created by Ligation and Lib-A for Library created by Amplification (with respective primers).

FLX had three different kits for ePCR but not Lib-L and Lib-A which are only for Titanium.

The kit Lib-A contains two different types of beads (A and B). Library fragments are attached to the beads with either A or B adapter and the other one is used for sequencing (direction towards the bead). So the beads have to fit to the direction one want to use for sequencing. Hence for bidirectional sequencing one uses both types of beads.
For unidirectional amplicon sequencing one can prepare ePCR with either A or B beads from Lib-A kit (using two kits as the kit has half A and half B beads) or with Lib-L kit if the primers were designed for it (only A direction is possible).

Then, 454 FLX Titanium employed a completely different set of adaptor/PCR primer sequences for "standard" libraries, as follows:

Titanium Primer A: 5'-CCA TCT CAT CCC TGC GTG TC-3'
Titanium Primer B: 5'-CCT ATC CCC TGT GTG CCT TG-3'

==========

Ligation Problem (generating mRNA --> cDNA)

I used the kit Rapid cDNA Library (GS FLX Titanium). To recover the small RNA, I modified slightly the protocol at the level of purification (calibration of beads Ampur) between the enzymatic steps (RT, Fragment repair, ..) Unfortunately, it seems that the step of the ligation of double-stranded adapters doesn't work very well because the controls checked using Bioanalyser of Agilent (high sensitivity) show the presence of my product, but no matches were exploitable using the TBS fluorometer and at the stage of titration

Trouble shooting:
1. Did you do a positive control to make sure the primers you are using will work?

Test with primers A and B on a previous library prepared from a kit GS FLX Titanium (not quick Libary preparation) and saw a good smear on the agarose gel after PCR.

If so, then it is likely that your adaptors failed to ligate to the double stranded cDNA. Possible explanations for this would include:
(1) Contaminants from clean-up steps/washes (eg, chaotropes or alcohols) prevented one or more of the blunting, a-tailing or ligation reactions. The most common issue I see is:

(i) MinElute - ethanol contamination. This is easy to fix using a speed vap. If you nose is sensitive to ethanol, you can smell it.

(ii) Acetate salts - I think. You can actually see these, if they are really high, in the UV spectrum as a peak in the 230 nm region.

(2) First or second strand cDNA synthesis failed.

I do not think that it was a trouble with the RT or the double strand synthesis because the Bioanalyser (DNA Chip) posts a explanable profile, it is therefore clearly an dsDNA. Moreover, With the same RNA, i tested another kit from Ambion which included a RT and double strand synthesis, then after purification with Ampure, i came back into the Roche kit at the Fragment repair step until the end.

(3) exonucleases destroyed your adaptors during the ligation step.
Rapid Libraryy Presp:

500 ng adequate (need good, reproducible quantitation - fluroscence)

sometimes there are a lot of adapter dimers & need to gel isolate the libraries to avoid those

I do think that I need to modify my shearing protocol a bit to add more time because the average fragment lengths were on the higher side (upper 800s). Although the kit says that 600-900bp average is good, I'd rather it be a bit smaller so that there is limited kick out of larger fragments in emPCR to prevent preferential amplification of only smaller frags.

rRNA libraries:
Typical PCR reactions yield vastly more product for short templates than for longer ones.

But it could also be that your read lengths are short for some other reason. If your templates are actually short you should have a high number in your "Short Primer" metric. If not, then something else is likely the source of your short reads.

I will mention one: polyA, of course, is the bane of 454 runs... Even with random primed cDNA libraries, it is possible to run into polyA problems. It should be less, but polyA+ RNA isolation will enrich for polyA. If you start with heavily degraded RNA and pull out polyA from that, you may get a high percentage of your cDNA being polyA or polyT--even if you used random primed reverse transcription.

How to write text tab separated maping files (qiime & clovr) using gedit (linux ubuntu):

1. Create a text file as normal

2. To see the tabs, find text and replace with tabs

3. For this enable the Draw Spaces plugin which also draws tabs:
   Choose Edit > Preferences from the Gedit menu.
   Choose the Plugins tab, then select Draw Spaces.
   Choose the Configure Plugin button and verify that Draw tabs is selected.

4. You can than toggle the display spaces and tabs from the menu:
   Choose View > Show White Space.

5. Tabs are represented by an arrow.

6. Gedit uses an escaped-t (\t) in find and replace to represent the tab character. For example, you can change a comma delimited file into a tab delimited file:
   Choose Search > Replace from the Gedit menu.
   Enter ',' in the Search for field.
   Enter '\t' in the Replace with field
   Choose either Replace or Replace All to change the text to tabs.

I suggest you enable the Draw Spaces plugin which also draws tabs:
Choose Edit > Preferences from the Gedit menu.
Choose the Plugins tab, then select Draw Spaces.
Choose the Configure Plugin button and verify that Draw tabs is selected.

You can toggle the display spaces and tabs from the menu:
Choose View > Show White Space.

Tabs are represented by an arrow.

Gedit uses an escaped-t (\t) in find and replace to represent the tab character. For example, you can change a comma delimited file into a tab delimited file:
Choose Search > Replace from the Gedit menu.
Enter ',' in the Search for field.

Enter '\t' in the Replace with field
Choose either Replace or Replace All to change the text to tabs.

If you have access to the Roche software (Newbler) that would be your easiest and best place to begin for 454 assembly. It does a great job and you should be able to contact whoever did the sequencing to run a basic assembly.

You can request a copy of Newbler from Roche. I don't believe it is available for download, but there should be instructions on their website on how to obtain the software. If that doesn't work you may want to consider the following open-source options.

1) FASTQC (http://www.bioinformatics.bbsrc.ac.uk/projects/fastqc/). Sequence data is never of equal quality for all reads. You will want to trim/filter some of your reads to enrich for high-quality data. FASTQC is a multi-platform application which will aid in your visualization of the quality of your data.

2) GALAXY and the FASTX Toolkit (http://main.g2.bx.psu.edu/ and http://hannonlab.cshl.edu/fastx_toolkit/). FASTQC should tell you things like how is the sequence quality at the 5' end of my reads. Frequently it will be low and you may want to exclude this sequence from subsequent analysis. Using tools available in GALAXY and the FASTX Toolkit you should be able to filter and trim your data to your hearts content. Both packages are well documented. GALAXY has more functionality than a swiss army knife wielding ninja and I recommend you take a look at the entire package as well as some of the web-tutorials. It offers an ideal platform for an entry level bioinformatician looking to do some work in genomics. A short tutorial on how to do QC on sequence data can be found here:
http://www.molecularevolution.org/resources/activities/QC_of_NGS_data_activity along with a variety of other tools/tutorials that might be of interest to you.

3) Bowtie/VELVET/NEWBLER/AbYSS/MIRA: These programs should help you to assemble your filtered/trimmed data into something a bit more reasonable to handle. There are a large number of assemblers and mappers that can help you do this task. Assembling next-gen data is an under appreciated and challenging aspect of genomics to many biologists. Each data set has it's unique qualities making it no so easy to cookie cut. As I recommended above, I would use NEWBLER if you have access to it. If not, any of the ones listed here should get you started. Bowtie is easy to use and very fast. If you have a high-quality reference genome this may be the way to go. VELVET takes a lot of memory, but may be an option if you do not have a good reference genome. AbYSS and MIRA can work if you do or do not have a reference genome. Bastien Chevreux has done an excellent job at writing and documenting MIRA. It is worth going through some of his exercises just for the learning experience alone.

4) Another package you may want to consider in order to QC, filter and trim your data is PRINSEQ: http://edwards.sdsu.edu/prinseq_beta/. I recommended it in another thread to someone else new to sequence analysis and it seemed to be a hit.

SAH
my excel are in this distribution: Contig name; Number EST; BlastClust 85%; Blast Info see image here: http://img848.imageshack.us/f/semttulogq.png/

NOTE:
I would suggest that you try to get the raw data either as SFF file or as FASTQ file. From the SFF file, you can extract the sequence and quality data and convert it into FASTQ format using e.g. PRINSEQ (or upload the FASTA and QUAL files directly to its web interface).

I am not aware of a program that will process your data in an Excel spreadsheet. If you can't get the raw data, try to convert your spreadsheet into a FASTA file.
Looking at your screenshot, it looks like someone already run BLAST on the data. It also looks like the sequences are contigs (header in first column), which suggests that they are already assembled.
If you want to redo the analysis, start with the raw data and process it with PRINSEQ or an alternative. If

you are not sure what parameters to use for the processing, take a look at the manual site of PRINSEQ (http://prinseq.sourceforge.net/manual.html).

Again, each assembly has it's own nuances, so you may end up using multiple packages/techniques to get the job done. But I think any of these programs should get you started.

Depending on what you are able to assemble you will then need to decide what interests you about the data. Comparison with other species? Polymorphism analysis? Do you need to gather more data? Gene finding? synteny? Getting the data to a reasonable quality, assembling it and taking a look should help you to answer these questions.

Apache2 Ubuntu

Debian, Ubuntu (Apache httpd 2.x):

```
ServerRoot              ::      /etc/apache2
DocumentRoot            ::      /var/www
Apache Config Files     ::      /etc/apache2/apache2.conf
                        ::      /etc/apache2/ports.conf
Default VHost Config    ::      /etc/apache2/sites-available/default,
                                /etc/apache2/sites-enabled/000-default
Module Locations        ::      /etc/apache2/mods-available,
                                /etc/apache2/mods-enabled
ErrorLog                ::      /var/log/apache2/error.log
AccessLog               ::      /var/log/apache2/access.log
cgi-bin                 ::      /usr/lib/cgi-bin
binaries (apachectl)    ::      /usr/sbin
start/stop              ::      /etc/init.d/apache2
       (start|stop|restart|reload|force-reload|start-htcacheclean|stop-htcacheclean)
```

Notes:

1. *The Debian/Ubuntu layout is fully documented in /usr/share/doc/apache2/README.Debian*
2. *Debian/Ubuntu use symlinks to configure vhosts and load modules. Configuration files are created in their respective sites-available and mods-available directories. To activate vhosts and modules, symlinks are created in the respective sites-enabled and mods-enabled directories to the config files in either sites-available and mods-available. Debian provides scripts to handle this process called 'a2ensite' and 'a2enmod' which activates vhosts and modules.*
3. *The default vhost is defined in /etc/apache2/sites-available/default, and overrides the DocumentRoot set in the server context.*

   `(start|stop|restart|reload|force-reload|start-htcacheclean|stop-htcacheclean)`

Testing the version of apache2:

>/usr/sbin/apache2ctl status | grep Version

'www-browser -dump http://localhost:80/server-status' failed.
Maybe you need to install a package providing www-browser or you need to adjust the
APACHE_LYNX variable in /etc/apache2/envvars

> apache2 -v

Server version: Apache/2.4.6 (Ubuntu)
Server built: Jul 30 2013 15:27:49

Setting permissions:

For web presentation:

1. `chmod -R 777 /www/store` will set read, write and execute to all files and directories R = recursive) for the whole world which is not what is required. (Alternatively navigate to the folder and `chmod -R 777`)

2. `chmod 755`

```
   7       5       5
user    group   world
r+w+x   r+x     r+x
4+2+1   4+0+1   4+0+1   = 755
```

3. chmod 644

- **R**ead 4 - Allowed to read files
- **W**rite 2 - Allowed to write/modify files
- e**X**ecute1 - Read/write/delete/modify/directory

Setting up file permissions properly:
You may not have gotten all of your files/folders set properly. Doing it manually is not failsafe. Have you tried running any shell scripts to repair your permissions? Have you also checked the ownership of your files?

Navigate to your web directory, then
find . -type f -exec chmod 644 {} \; && find . -type d -exec chmod 755 {} \;

BE CAREFUL only run this in the web directory where all your publicly-visible files are located.

You may also need to chown your files to your correct user/group - that setting will depend on your web host, though, so I can't give you the exact command.

De novo assembly for short-read mRNA-Seq

Overview

This presentation will include a live demo of both the Trinity and Oases *de novo* RNA-Seq assemblers, which are arguably, currently the two most popular *de novo* RNA-Seq assemblers at the moment. The presentation will also include a chalk talk comparing the theory underlying the approaches taken by each assembler. Discussion will then turn towards practical considerations, as well as any other issues brought up during the presentation. If time permits, there may also be very brief demonstrations of potential "next steps" to take with newly assembly mRNA contigs.

Getting Started with a new system

Most modern operating systems, even Linux, do not come with prepackaged with comprehensive development environments. This saves both bandwidth and space for the vast majority of users who will never use any part of their computer that can't be navigated by mouse clicks alone. For the rest of us, this means that we have to do a bit of tinkering before we can really get started using a new system.

Mac OS X

1. Open the App Store App
2. Install Xcode
3. Open Xcode
4. Navigate to Preferences (click the Apple in the upper left corner)
5. Select the "Downloads" tab
6. Click the button to install "Command Line Tools"

Windows

I would strongly recommend using Cygwin, Wubi, or gaining access to a *nix-based system if you intend to develop or use open source software on a regular basis.

Ubuntu (Linux)

I will use Ubuntu 12.04 LTS for the rest of this demo.

1. Open a terminal
2. Type the following commands to install various things from the Apt repository:

```
sudo apt-get update
sudo apt-get install build-essential
sudo apt-get install linux-headers-$(uname -r)
```

Downloading software from the command line

Clicking web links to download files is very convenient as long as you're using the same local computer for everything. Downloading files to a local computer, only to immediate have to copy them to a remote server can quickly become tedious however, so I recommend learning to use `wget`. This allows files to be downloaded directly to any machine, without having to stage it on some intermediate work station.

I find it helpful to create a single directory to hold all of the software I download. I tend to use `/usr/local/bin`, but this requires root access and can sometimes create other complications, so I will use the placeholder `/local/source/code/repository/` for the rest of this demo.

```
ssh user@server
cd /local/source/code/repository/
wget <link>
```

It is worth noting that links copied from browsers will frequently have the form `http://some.website.com/filename.tgz/download`. If this is passed to wget, then wget will save the file as "download". This is annoying because tar will then complain about the suffix. An appropriate suffix can be added after the fact, but I find it easier to just remove the `/download` from the link. I can't guarantee that this will always work, but I've never had problems doing it.

Unpacking compressed directories

Many software packages are available as compressed directories. This greatly reduces the amount of information that must be transmitted over the web. The simplest way to unpack these "tarballs" is using the `tar` command, which can be found on most computers.

```
tar -xzvf <filename>.tgz
tar -xzvf <filename>.tar.gz
tar -xjvf <filename>.bz2
tar -xjvf <filename>.bzip2
```

All those flags tell `tar` to e***x***tract either a g***z***ipped or, uh, bzipped (denoted ***j*** for some reason) ***f***ile and provide ***v***erbose output as it works. The original compressed files can then be deleted with the `rm` command.

Compiling and installing software

Once the directories have been unpacked, they will likely contain some combination of source code and/or prebuilt binaries. If they contain binaries, and you were careful to download the correct version for your system, then installation may be as simple as copying them to a directory in your path. The most common place to install custom software is `/usr/local/bin`, which requires root access.

```
cd /local/source/code/repository/<software>/
sudo cp <binary_name> /usr/local/bin/
```

It is common for binaries to located in a subdirectory called `bin`, although this is far from universally practiced.

If the directory contains source code that must first be built, the most common method is using the `make` command. First, change into the source code directory, then look for files called `configure`, `config`, `make`, or `Makefile`. If you see some combination of these, then try the following set of commands, knowing that the first and third may fail.

```
./configure
make
sudo make install
```

If you get a message saying that the `./configure` command wasn't found, then don't worry about it. Same with `sudo make install`. As long as `make` completes successfully, then you should have created binaries somewhere in that directory. The "install" often just copies them into `/usr/local/bin/`, which you now know how to do yourself :)

Installation Summary (for Ubuntu 12.04)

Install Java and some additional libraries

```
sudo apt-get install default-jre
sudo apt-get install zlib1g-dev
sudo apt-get install libncurses5-dev
sudo apt-get install texlive-full
```

Install samtools

```
cd /local/source/code/repository/
wget
http://sourceforge.net/projects/samtools/files/samtools/0.1.19/samtools-0.1.19.tar.b
z2
tar -xjvf samtools-0.1.19.tar.bz2
cd samtools-0.1.19
make
sudo cp samtools /usr/local/bin/
```

Install Bowtie

```
cd /local/source/code/repository/
wget
http://sourceforge.net/projects/bowtie-bio/files/bowtie/1.0.0/bowtie-1.0.0-linux-x86
_64.zip
unzip bowtie-1.0.0-linux-x86_64.zip
cd bowtie-1.0.0
sudo cp bowtie /usr/local/bin/
sudo cp bowtie-build /usr/local/bin/
sudo cp bowtie-inspect /usr/local/bin/
sudo chmod +x /usr/local/bin/*
```

Install Trinity

```
cd /local/source/code/repository/
wget
http://sourceforge.net/projects/trinityrnaseq/files/trinityrnaseq_r2013_08_14.tgz
tar -xzvf trinityrnaseq_r2013_08_14.tgz
cd trinityrnaseq_r2013_08_14
make
sudo ln -s Trinity.pl /usr/local/bin/
```

Software installed by Trinity:

- JellyFish
- Inchworm
- Chrysalis
- QuantifyGraph
- GraphFromFasta
- ReadsToTranscripts
- fastool
- parafly
- slclust
- collectl

Install Velvet

```
cd /local/source/code/repository/
wget http://www.ebi.ac.uk/~zerbino/velvet/velvet_1.2.10.tgz
tar -xzvf velvet_1.2.10.tgz
cd velvet_1.2.10
make 'CATEGORIES=1' 'MAXKMERLENGTH=64' 'OPENMP=1'
sudo cp velvet[gh] /usr/local/bin/
```

Install Oases

```
cd /local/source/code/repository/
wget http://www.ebi.ac.uk/~zerbino/oases/oases_0.2.08.tgz
tar -xzvf oases_0.2.08.tgz
cd oases_0.2.08
make 'VELVET_DIR=/usr/local/src/velvet/1.2.10' 'CATEGORIES=1' 'MAXKMERLENGTH=64'
'OPENMP=1'
sudo cp oases /usr/local/bin/
sudo cp scripts/oases_pipeline.py /usr/local/bin/
```

Sample Data

Many software packages come with small test data sets that can be used by end users verify that they have the software installed and running properly. Trinity and Oases both come with test data sets, but the Oases reads are stored in an interleaved FASTA file, which pretty much only Velvet uses. I'm therefore going to focus on the Trinity practice reads for this demo.

Interestingly, the most recent version of Trinity lacks support for compressed read files, even though the practice reads still come gzipped, so we first have to uncompress them.

```
gunzip -c
/local/source/code/repository/trinityrnaseq_r2013_08_14/sample_data/test_Trinity_Ass
embly/reads.left.fq.gz > ~/Desktop/BYOB_2013-09-10/reads.left.fq
gunzip -c
/local/source/code/repository/trinityrnaseq_r2013_08_14/sample_data/test_Trinity_Ass
embly/reads.right.fq.gz > ~/Desktop/BYOB_2013-09-10/reads.right.fq
```

Running Trinity

Trinity is relatively simple to run, for an assembler. If the `Trinity.pl` script has been linked to the user's `$PATH`, and the user is in the directory containing the reads file `reads.left.fq` and `reads.right.fq`, then Trinity can be run as follows:

```
Trinity.pl --seqType fq --JM 1G --left reads.left.fq --right reads.right.fq
```

Arguably, a better way to run it is to be more specific about the paths and the parameters:

```
/usr/local/src/trinityrnaseq/r2013_02_25/Trinity.pl --seqType fq --JM 1G --left
~/Desktop/BYOB_2013-09-10/reads.left.fq --right
~/Desktop/BYOS_2013-09-10/reads.right.fq --output byob_trinity_r2013_02_25_demo
--CPU 2
```

This is pretty difficult to read though, so a nicer way to run Trinity is to create `run.sh` script that lays out the commands in a way that makes it easier to read, and therefor to catch potential errors.

```
#!/bin/bash
```

```
left=~/Desktop/BYOB_2013-09-10/reads.left.fq
right=~/Desktop/BYOS_2013-09-10/reads.right.fq

time nice /usr/local/src/trinityrnaseq/r2013_02_25/Trinity.pl \
--seqType fq \
--JM 1G \
--left $left \
--right $right \
--output byob_trinity_r2013_02_25_demo \
--CPU 2 \
| tee byob_trinity_r2013_02_25_demo.log
```

To run a different version of Trinity, just link to the corresponding `Trinity.pl` script.

Running Oases

Oases also contains a wrapper script called `oases_pipeline.py`, which I also prefer to call from a `run.sh` shell script within each working directory:

```
#!/bin/bash

reads1=~/Desktop/BYOB_2013-09-10/reads.left.fq
reads2=~/Desktop/BYOB_2013-09-10/reads.right.fq

kmin=15
kmax=63
step=2
merge=25
insLen=300

time nice oases_pipeline.py \
-m $kmin \
-M $kmax \
-s $step \
-g $merge \
-o byob_oases_demo \
-d " -shortPaired -separate -fastq $reads1 $reads2 " \
-p " -ins_length $insLen "
```

Unlike trinity, Oases does not provide verbose output as it works, so I use the following command to monitor it's progress:

```
while true; do ls -lhd */; sleep 5; done
```

After this completes, the merged assembly can be tweaked by using the `-r` command with the `oases_pipeline.py` script:

```
time nice oases_pipeline.py -m 21 -M 41 -g 21 -r -o byob_oases_demo
```

Install Bowtie2

```
cd /local/source/code/repository/
wget
http://sourceforge.net/projects/bowtie-bio/files/bowtie2/2.1.0/bowtie2-2.1.0-linux-x
86_64.zip
unzip bowtie2-2.1.0-linux-x86_64.zip
cd bowtie2-2.1.0
sudo cp bowtie2* /usr/local/bin/
```

Comparing the Assemblies

Installing NCBI-BLAST+

```
cd /local/source/code/repository/
wget
ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/ncbi-blast-2.2.28+-x64-li
nux.tar.gz
tar xzvf ncbi-blast-2.2.28+-x64-linux.tar.gz
cd ncbi-blast-2.2.28+
sudo cp bin/* /usr/local/bin/
```

Downloading NCBI BLAST databases

There is a "proper" way to do this using wget options, but I don't know those options, and I *do* know how to use BASH for-loops, so I did this instead:

```
cd /local/database/repository/
for i in seq $(seq -f "%02g" 0 15); do wget ftp://ftp.ncbi.nlm.nih.gov/blast/db/nt.$
{i}.tar.gz; done
wget ftp://ftp.ncbi.nlm.nih.gov/blast/db/est_mouse.tar.gz.md5
```

I'm pretty sure this next part doesn't require a loop, but I used one anyway. Note that I dropped the dash (-) on the `tar` options. This was not a typo. The `tar` command will accept this, though it is good to be in a habit of using normal command line option syntax.

```
for file in *.tar.gz; do tar xzvf $file; done
```

Installing HMMer3

```
cd /local/source/code/repository/
wget
ftp://selab.janelia.org/pub/software/hmmer3/3.1b1/hmmer-3.1b1-linux-intel-x86_64.tar
.gz
tar xzvf hmmer-3.1b1-linux-intel-x86_64.tar.gz
cd hmmer-3.1b1-linux-intel-x86_64
sudo cp binaries/* /usr/local/bin/
```

Downloading Pfam-A

```
cd /local/database/repository/
wget ftp://ftp.sanger.ac.uk/pub/databases/Pfam/current_release/Pfam-A.hmm.gz
gunzip Pfam-A.hmm.gz
```

Then prepare it for use with either `hmmsearch` or `hmmscan` by formatting it with `hmmpress`:

```
cd /local/database/repository/
hmmpress Pfam-A.hmm
```

Installing Google sparehash

```
cd /local/source/code/repository/
wget https://sparsehash.googlecode.com/files/sparsehash-2.0.2.tar.gz
tar -xzvf sparsehash-2.0.2.tar.gz
cd sparsehash-2.0.2
./configure
make
sudo make install
```

Installing the C++ Boost libraries

```
cd /local/source/code/repository/
wget
http://downloads.sourceforge.net/project/boost/boost/1.50.0/boost_1_50_0.tar.bz2
tar xjvf boost_1_50_0.tar.bz2
cd boost_1_50_0
sudo ./boostrap.sh
sudo ./b2
```

Installing ABySS

```
cd /local/source/code/repository/
wget
http://www.bcgsc.ca/platform/bioinfo/software/abyss/releases/1.3.6/abyss-1.3.6.tar.g
z
tar -xzvf abyss-1.3.6.tar.gz
cd abyss-1.3.6
ln -s /local/source/code/repository/boost_1_50_0/boost boost
./configure
make
sudo make install
```

Installing R & Rstudio (and some useful packages)

I wanted R 3.0, which isn't currently available through the apt-repositories, so I had to do a bit of tinkering as per the instructions on this website:
http://cran.r-project.org/bin/linux/ubuntu/README.html

First, open `/etc/apt/sources.list` as root using your-favorite-text-editor. I like vim, though I don't recommend it.

```
sudo vim /etc/apt/sources.list
```

Add this line to the end (I normally use the NCI mirror, but I wasn't able to reach it while I was making this tutorial, so I used the CMU mirror instead):

```
deb http://lib.stat.cmu.edu/R/CRAN/bin/linux/ubuntu precise/
```

Then install R and some useful libraries (everything except `plyr`, `ggplot2`, and `knitr` are on this list because R complained about them being out of date).

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys E084DAB9
sudo apt-get update
sudo apt-get install r-base
sudo apt-get install r-base-dev
sudo R
install.packages("codetools", dependencies=True)
install.packages("plyr", dependencies=True)
install.packages("MASS", dependencies=True)
install.packages("lattice", dependencies=True)
install.packages("survival", dependencies=True)
install.packages("rpart", dependencies=True)
install.packages("foreign", dependencies=True)
install.packages("cluster", dependencies=True)
install.packages("ggplot2", dependencies=True)
install.packages("knitr", dependencies=True)
```

Press ctrl+d to close the R session, then install Rstudio from the web and enjoy the simple nostalgic pleasure of clicking buttons for a few minutes.

Installing Prokka

1. Download prokka-1.7.tar.gz

2. sudo cp prokka-1.7.tar.gz /opt

3. sudo tar -xvf prokka-1.7.tar.gz

4. vi .zshrc and add path
   - export PATH=$PATH:/opt/prokka-1.7/bin

5. installing HMMer3 (replace ver 3.0 with ver 3.1)
   wget ftp://selab.janelia.org/pub/software/hmmer3/3.1b1/hmmer-3.1b1-linux-intel-x86_64.tar.gz OR
   Download from ….....
   sudo tar xzvf hmmer-3.1b1-linux-intel-x86_64.tar.gz
   cd hmmer-3.1b1-linux-intel-x86_64
   sudo cp binaries/* /usr/local/bin

6. Installing new version of tbl2asn
   ftp://ftp.ncbi.nih.gov/toolbox/ncbi_tools/converters/by_program/tbl2asn/
   sudo gunzip linux64.tbl2asn.gz to uncompress & rename the file to remove the platform designation
   sudo cp tbl2asn to /usr/local/bin

P

Steps in Phylogenetic Analysis using phylosift

1. RAST:
   Download annotated GenBank files (.gbk) from RAST server

2. Convert Genbank files to fasta format (keeping the annotations intact) using fasta.phy
   > eg: genbank_to_fasta.py -i file.gbk -o file.faa -q locus_tag, gene,product,location
   > eg: genbank_to_fasta.py -i rast.gbk -m genbank -o outfile.fasta -s aa -f CDS rRNA tRNA -q product

3. Command: phylosift all --output=results <filename.faa>.
   > eg: phylosift all --besthit Y50.fasta --output=results
   > eg: phylosift all --besthit --output=results file.fasta

4. Results will be in the "results" directory , which is newly created within your `phylosift_v1.0.0_02` folder. The directory should contain a number of different files and folders, as follows:

5. Forester -
   - java -ver
   - java -jar forester.jar

6. krona (krona tools is in usr/local)

6. Archaeopteryx (http://aptxevo.wordpress.com/)

7.

5.

fasta.py

Installation Instructions:
1. Make sure you have a working version of Biopython installed. This software was developed on version 1.54, though later versions should work. It can be found at: http://biopython.org/.

2. Adjust the first line of the genbank_to_fasta.py file to point to your system's python binary. If you don't know where it is, type 'which python' into the command line.

3. Move the genbank_to_fasta.py file to an executable directory. On many systems, it is /usr/local/bin/ or /usr/bin/.

4. From the command line type 'genbank_to_fasta.py -h' for usage information.

Usage:
   genbank_to_fasta.py -h for help

   genbank_to_fasta.py -i FILE [options]

Options:
 --version

Show program's version number and exit

-h, --help
Show this help message and exit

-i FILE, --in_file=FILE
Specify the input FILE that you wish to convert

-m FORMAT, --file_format=FORMAT
Specify the input file format. Specify 'genbank' or 'embl'. Default is genbank.

-o FILE, --out_file=FILE
Specify the path and name of the output fasta file you wish to create. Default will be the same as the in_file, but with a 'fasta' suffix.

-s SEQUENCE_TYPE, --sequence_type=SEQUENCE_TYPE
Specify the kind of sequence you would like to extract. Options are 'aa' (feature amino acids), 'nt' (feature nucleotides), 'whole' (the entire sequence, not just sequence corresponding to features) and 'taa' (amino acids translated on the fly, which generates amino acid sequence by translating the nucleotide sequence rather than extracting from the feature table).Default is 'aa'.

-f FEATURE_TYPE, --feature_type=FEATURE_TYPE
Specify the type of feature that you would like to extract. This option accepts arbitrary text, and will fail if you input a non-existent feature name. Common options are 'CDS', 'rRNA', 'tRNA', or 'gene'. Default is 'CDS'.
-d DELIMITER, --delimiter=DELIMITER
Specify the character you wish to use to separate header elements. Options are 'tab', 'space', 'spacepipe', 'pipe', 'dash', or 'underscore'. Default is 'spacepipe'.

-q QUALIFIERS, --qualifiers=QUALIFIERS
Specify which qualifiers should make up the fasta header line. Takes comma separated list. Will accept any qualifier that appears in your genbank file, (e.g. 'note', 'protein_id', etc). Qualifiers appear in the header line in the order you list them. Use 'location_long' for the exact location information as it appears in the input file. Default is 'locus_tag,gene,product,location'.

-a ANNOTATIONS, --annotations=ANNOTATIONS
Specify which record annotation should make up the header line. Takes comma separated list. Will accept any annotation that appears in your genbank file, (e.g. 'comment', 'taxonomy', accessions, etc). Only used with --sequence_type = whole. Default is 'organism'.

-u USER_HEADER, --user_header=USER_HEADER
If you prefer to specify your own completely custom header line, you may specify it here. Should be speccified in single quotes. Only used with --sequence_type = whole.

Phylosift

PhyloSift currently accepts input data in the following file formats:

* FASTA
* paired FASTA (specify paired data by using the --paired flag)
* interleaved paired FASTA (specify paired data by using the --paired flag)
* FASTQ (this file type is the standard output from Illumina platforms)
* interleaved FASTQ (same as FASTA but with quality scores)

* .gz (any of the above file types compressed using gzip)
* .bz2 (any of the above file types compressed using bzip2)

PhyloSift can be executed by running the following command:

$ phylosift <Mode> <options> <sequence_file>

$ phylosift <Mode> <options> --paired <sequence_file_1> <sequence_file_2> sequence_file_1 and _2 must contain paired sequences

Creating a PhyloSift marker :

Example 1: `phylosift build_marker -f --alignment=test.aln --reps_pd=0.01`

Example 2: `phylosift build_marker -f --alignment=test.aln --taxonmap=test.taxonmap`

NOTE:

- The new marker will be added directly to the phylosift marker database.
- This step does not automatically add the new marker to the search databases. You will need to run
- `phylosift index` after building a marker.
- If a marker with the same name already exists, the marker build process will be halted unless the -f or --force option is used.
- The marker name will be the same as the alignment given minus the trailing suffix. If the user wants to build a marker from the file <test.aln>, the marker name will be <test>

== Index the search databases ==

This step is run automatically when markers are downloaded but if you add new markers, you will need to run this step manually.

`phylosift index`

== Modes ==

- commands: list the application's commands
- help: display a command's help screen
- align: align homologous sequences identified by 'search'
- all: run all steps for phylogenetic analysis of genomic or metagenomic sequence data
- benchmark: measure taxonomic prediction accuracy on a simulated dataset
- build_marker: add a new marker the reference database based on a sequence alignment
- dbupdate: update the phylosift database with new genomic data
- index: index a phylosift database after changes have been made
- name: Replaces phylosift's own sequence IDs with the original IDs found in the input file header
- place: place aligned reads onto a reference phylogeny
- search: search input sequence for homology to reference gene database
- simulate: simulate sequencing from a metagenomic sample
- summarize: translate a collection of phylogenetic placements into a taxonomic summary
- test_lineage: conduct a statistical test (a Bayes factor) for the presence of a particular lineage in a sample

== Requirements ==

PhyloSift requires a 64-bit operating system. PhyloSift will NOT work on a 32bit operating system. PhyloSift depends on a great many other open source software packages. The precompiled version linked above bundles most of the dependencies into a single downloadable package.

== Results ==

Results are saved to the path specified with the --output option. If no path is given, the default location

for results is `PS_temp/<filename>/`.

* blastDir: All files related to the search step.
  * *.candidate.aa.* Fasta format of the candidate sequences in Protein space for each marker
  * *.candidate.ffn.* Fasta format of the candidate sequences in DNA space for each marker (option activated)

* alignDir: All files related to the alignment and masking steps.
  * *.newCandidate.* Fasta file of the candidate sequences copied from the blastDir
  * *.fasta hmmer3 generated alignment for the candidate sequences in fasta format
  * *.codon.*.fasta reverse-translated alignment of DNA
  * *.unmasked. The unmasked sequence of homologs that aligned successfully and passed all filter thresholds

* treeDir: Files containing placements of sequences onto reference phylogenetic trees
  * *.jplace Phylogenetic placements of the aligned sequences
  * *.codon.*.jplace Phylogenetic placements for codon alignments

* taxasummary.txt Probability mass over taxa present in the sample, tab-delimited text
  Column 1 -- NCBI Taxon ID
  Column 2 -- Taxonomic rank (genus, species, phylum, etc)
  Column 3 -- Name
  Column 4 -- Read/sequence probability sum placed at this taxon
  The values in this column can be normalized to sum to 1, the result will be a rank-abundance distribution

== SUPPORT AND DOCUMENTATION ==
After installing, you can find documentation for this module with the perldoc command.
  perldoc Phylosift::Phylosift

Bugs and other apparent problems with the software can be reported as issues in our github issue tracker :
https://github.com/gjospin/PhyloSift/issues

== 3rd PARTY SOFTWARE ==
PhyloSift is distributed with several open source components that were developed by other groups. These components are (c) their respective developers and are redistributed with PhyloSift to provide ease-of-use.

Please see the following web sites for licensing details and source code for these other components:
* pplacer -- http://matsen.fhcrc.org/pplacer/
* HMMER 3 -- http://hmmer.janelia.org/
* LAST -- http://last.cbrc.jp
* pda -- http://www.cibiv.at/software/pda/
* FastTree -- http://www.microbesonline.org/fasttree/

* infernal -- http://infernal.janelia.org/

The above list is not exhaustive.

== CONTACT INFORMATION ==
Please direct correspondence to aarondarling@uc.davis.edu Or on twitter to @PhyloSift

MY NOTES ON PHYLOSIFT

1. When phylosift is run, than the marker genes are downloaded from (http://edhar.genomecenter.ucdavis.edu/~koadman/ncbi.tgz) and becomes available in /home/bharat/share/phylosift. Perhaps can also be obtained by using wget with the URL.

2.

Converting rast GenBank file (contains multi contigs and their annotations)

- The output rast file (.gbk) will only open the first contig in Artemis.

- This rast file can be concatenated into single contig) using the following UNION command (EMBL):

  union -sequence infile.gbk -outseq outfile.gbk -osformat genbank -feature -auto

- Run artemis:

  art outfile.gbk

WSG Submission

1. Go to ….... and register the Project Details. Wait for GenBank to send you a confirmation that the project has been registered.

2. Go to http://www.ncbi.nlm.nih.gov/WebSub/template.cgi, create GenBank submission template and save it in your directory (it will be saved as template.sbt, which is a default file) for use with tbl2asn

3. Go to rast http://rast.nmpdr.org/rast.cgi and download the whole genome directory.

4. Create a new directory (eg WGS_Genome_Data_Preparation)
   - copy template.sbt and the genome directory.
   - Expand the genome directory with command: tar -xvf filename.tgz
   - Go into the genome directory and add headers to the contigs file using the following command:
     perl -p -e 's/^(>.*)$/$1 **[organism=xxxxxx]**/g' **infile** > **outfile.fsa**
     eg perl -p -e 's/^(>.*)$/$1 **[organism=Halomonas] [strain=BC04]**/g' contigs_test > contigs.fasta
   - Create a new directory (eg Submission_Data) and mv template.sbt and outfile.fsa into it
   - Use the command tb2asn to add the template to contig.fsa
     tbl2asn -p path_to_fsa_files -t template.sbt -M n -Z discrep; where where path_to_files is the path to the directory where the .fsa and .tbl files are located.
     NOTES: Only include the path and NOT the files in the command otherwise it will fail
     The following files will be generated:
     - contigs.val (check this file for errors)
     - discrep
     - errorsummary.val (check this file for errors)
     - contigs.sqn (for submission)

6.

eg tbl2asn -p /home/bharat/Downloads/Desktop/Genomes_Final_Oct13/Bharat_genomes/AeB_Fervidicella_metallireducens/GenBank_Submission_and_Papers/submission -t template.sbt -M n -Z discrep

NOTE:

6.

Phylosift website – update info about output files

November 9, 2012 Leave a comment

Went through the updated list of output files with Guillaume. Here are the deets for all the files now being created in the PS_temp directory for each run:

alignDir

protein coding markers

- *1.unmasked – aligned protein with no masking – not used in downstream analyses
- *codon.updated.1.fasta – nucleotide, aligned and masked
- *.newCandidate.aa.1 – same file (unaligned version of hits)
- *.updated.1.fasta – protein, aligned and masked

.1 refers to chunk number – so if you have duplicate files with .2, etc

16/18S

- .1.unmasked – aligned nucleotide ith no masking
- .short.1.fasta – alignment using cmalign, with masking
- .long.1.fasta – will we have this too? and sep file for unmasked long sees?

Do we get two .unmasked files if we have a mix of short and long sequences?

no unaligned file in alignDir for 16S/18S data

blastDir

- marker_summary.txt – how many hits per marker for each gene

Search mode –keep_search – flag that retains all the search info in the BLAST directory; automatically retains the temp blast files

–keep_search – just undocumented, need to document this under output for **all** mode

treeDir

- enolase.codon.updated.sub1.1.jplace — nucleotide jplace
- enolase.updated.1.jplace — aa jplace

How is the information from codon and aa trees used in phylosift summaries?

Main output directory – Krona reports

- filename_allmarkers.html – all markers in treeDir with jplaces
- filename.html – core markers DNGNGWU only
- filename.jnlp – javascript of FAT tree visualization
- filename.xml – fat tree viz itself?

Main output directory – summary files

- marker_summary.txt – based off of the taxon summary files
- run_info.txt – going to be updated in the next few days; lists commands and md5 sums and step completion status (start/end time and duration for each chunk at each step – search, align, place, summarize)
- sequence_taxa_summary.1.txt – summary of chunk
- sequnce_taxa_summary.txt – combined info from all chunks
- sequence_taxa.1.txt – summary of chunk
- sequence_taxa.txt – combined information from all the chunks
- taxa_90pct_HPD.txt -
- taxasummary.txt

1. Making symbolic links

ln -s source_file target_file

Recog client (java)

- Download: recog-client-1.0.17.tgz from http://mbgd.nibb.ac.jp/RECOG/
- sudo chmod a+rx recog-client-1.0.17.tgz
- cp recog-client-1.0.17.tgz to /opt
- Extract recog-client-1.0.17.tgz: sudo tar zxvf recog-client-1.0.17.tgz
- cd into the created recog directory
- sudo make
- vi recog.sh and include the path to java (/usr/bin/java) in line 14, as follows:
    - ${JAVA_HOME}**/usr/bin/java** ${JAVA_OPTS} -classpath ${CLASSPATH} ${MAIN}
- Invoke ./reoc.sh – this will create RECOG folder in the home directory
    - /home/bharat/RECOG

NOTE: You need to install the RECOG server on your local machine only when you want to use genome data that are not avialble on the MBGD server.

Assemblers most recent website:

- http://ccb.jhu.edu/gage_b/genomeAssemblers/index.html

Griffith University's 23 Node (276 core) Gowonda HPC cluster

A quick start document can be found at: http://confluence.rcs.griffith.edu.au:8080/display/GHPC/Quickstart

Getting Help:

Griffith Library and IT help 3735 5555 or X55555
email support: eresearch-services@griffith.edu.au
You can log cases on service desk (category: eResearch services.HPC)
http://www.griffith.edu.au/servicedesk

eResearch Services, Griffith University
Phone: +61 - 7 - 373 56649 (GMT +10 Hours)
Email: eresearch-services@griffith.edu.au
Web: eresearch.griffith.edu.au

Getting Started:

ssh:

ssh s230993@gowonda.rcs.griffith.edu.au

Login:

UserID: s230993
Password: Central GU password

Transferring files between your desktop and gowonda cluster:

http://confluence.rcs.griffith.edu.au:8080/display/GHPC/Transferring+files+between+your+desktop+and+gowonda+cluster

Once you are on the system, have a look around. Your home directory is stored in:
/exports/home/<SNumber>, where you have 100GB of allocated space.

Work space is available at:
/scratch/<snumber> for short lived data; week old data is deleted from the folder

You should not read or write directly into your home directory with submitted jobs - they should always use the "/scratch/<snumber>" filesystem

linux and mac command line
use "scp" to transfer back and forth.

Running Jobs:

We provide modules that set up the environments you need to use for specific packages. You can see what modules are available with the command:

module available

and you can load them with the command:

module load MODULENAME

See sample pbs scripts here.
If you don't specify a maximum runtime, you will end up with a maximum runtime as short as 30 minutes, depending on which node your job runs on.

Job State:

when you run qstat from the command line, you will see your job's state within the queue. The following form lists the possible states and their descriptions

state code	State description
q	queued
r	running

A user guide has been developed and posted at:

http://confluence.rcs.griffith.edu.au:8080/display/GHPC/Gowonda+user+guide

If you are not familiar with secure shell usage, please look at this page:

http://confluence.rcs.griffith.edu.au:8080/display/GHPC/Quickstart#Quickstart-Whatisssh

You do everything on the login node, including running the batch jobs. The batching software is PBSPro11.The instructions for running batch jobs (for example matlab) are in the FAQ section of cluster documentation.

http://confluence.rcs.griffith.edu.au:8080/display/GHPC/FAQ

Should you have a need to visually examine your results, then gowonda has X11 enabled. You would need to connect using an X11 aware ssh client (eg. openssh -Y gowonda.rcs.griffith.edu.au from linux) On windows, you could use Xming. See instruction at:
http://confluence.rcs.griffith.edu.au:8080/display/GHPC/Quickstart#Quickstart-windowsplatform

You should compile and test your code on gowonda node only.
Please don't compile on other nodes.

Please adjust your codes and PBS scripts to make use of "/scratch/<snumber>" directory instead of your home directory.
If you have large data sets, you will need to copy them into scratch on the compute node(s) as part of your job submission script. See the user support FAQ for more information.

THERE IS A USER SUPPORT WEB SITE LOCATED AT :
http://confluence.rcs.griffith.edu.au:8080/display/GHPC/gowonda
http://tinyurl.com/7t6jsg7

In making use of this service, you agree to be bound by the Griffith Use of University Information Technology Resources Code of Practice which is available online at
http://www.griffith.edu.au/ins/org/techmenu/security/content_coc.html

If you have any questions about, or problems with, the cluster please don't hesitate to email eresearch-services@griffith.edu.au

You can log cases on service desk (category: eResearch services.HPC)
http://www.griffith.edu.au/servicedesk

Indy Siva
HPC Cluster Administrator
Information Services (Scholarly Information and Research)
Gold Coast Campus, Griffith University
QLD 4222, Australia

Information Services, G10, Room 3.29
Phone: +61 (07) 555 27259
Fax: +61 (07) 555 27255
Mobile: 0434 600 814
Email: i.siva@griffith.edu.au
http://eresearch.griffith.edu.au/

http://confluence.rcs.griffith.edu.au:8080/display/GHPC/Gowonda+user+guide (OR:

http://tinyurl.com/8a6fqpm)

Bio-hal

TO RUN Bio-Hal:

module load bio-hal

hal path_to_directory eg hal /export/home/s230993/bio-hal/bio-hal/trunk/test_data/small_complete_17/

BIOHAL TEST:

I think I have installed this now.

http://confluence.rcs.griffith.edu.au:8080/display/GHPC/bio-hal

To test it, I did this:

```
module load bioinformatics/prottest
module load bioinformatics/phylip/3.69
module load bioinformatics/raxml/7.3.0-SSE3-gcc
module load perl/5.15.8
module load bioinformatics/bio-hal/biohal
```

```
bio-hal is located (root)
    cd /sw/bioinformatics/bio-hal/trunk
    /sw/bioinformatics/bio-hal/trunk/requirements.pl
```

Bio-Hal Manual

***** Refer to the INSTALL document for instalation instructions and requirements for getting started *****

Hal is a perl program that executes our own perl scripts and search, clustering, alignment and phylogenetic programs from others to automate the identification of homologous clusters, building of amino acid super alignments and the phylogenetic analysis of it.

The INSTALL document lists the required pre-installed programs and has a test command to make sure Hal is working.

** The Basics **

Hal takes a directory of files (or list of files), with each file named Genus_specie.fasta and containing amino acid sequence in fasta format from one organism, as input. As output it produces a directory called 'results', which contains the super alignments of orthologous sequences, analyzed with the phylogenetic program of your choice (if you wish to see the intermediate step and additional reports, run hal with -E).

Hal will run through these basic steps:

1. All-vs-all Blastp
2. MCL clustering
3. Cluster selection and filtering
4. Alignment of clusters
5. Removal of poorly aligned positions and divergent regions
6. Concatenation of orthologous sequences into a super alignment
7. Optional parsimony analysis (Paup)
8. Optional neighbor-joining analysis (Phylip)
9. Amino acid substitution model test (ProtTest) if RAxML or Phyml are going to be used
10. Optional maximum likelihood analysis (RAxML)
11. Optional maximum likelihood analysis (Phyml)

Hal processes in the order above and if interupted will automatically restart from the correct point without redoing work it already completed.

** Interpreting the output **

When you run hal it will create three folders in the current directory: errors, workspace, results. The errors directory will only contain data if something failed while hal was trying to run. The workspace directory contains a lot of book keeping and intermediate files that the user should not touch (or do so knowing that they are voiding the warranty.). The results directory is the one of greatest interest. It will contain your super alignment and results from your phylogenetic analyses. It will also contain several intermediate steps and additional reports if ran with -E.

** The Scripts **

Much of the processing that hal does is delegated to other scripts. These scripts can also be used on their own. Below is a brief description of each.

Run any of the scripts with no arguments to see the usage and help.

hal:

Glues all the scripts and other programs together for the complete analysis.

parablast.pl:

This is a wrapper script to blastall. If no database is provided to blast against then the input is used (all-vs-all blast).

parseBlast.pl:
Parses a raw blastall result file into a tab delimited format.

paraClustering.pl:
This is a wrapper script for mcl (The Markov Cluster Algorithm) which takes a raw blastall result file as main input and runs mcl on it several times over a series of inflation parameters to create clusters. The main output is the selection of clusters, across several inflation parameters, with one sequence per organism or if specified one sequence per the number of organisms above a given threshold. Run with "-h" for more info.

toTabDelim.pl:
This script will take a parsed blast file and find the best hit for each query. If a list file, containing cluster assignments, is given, then the script will limited output to clusters in the list that contained sequences which had best hits to sequences in the same cluster.

toFasta.pl:
In the hal pipeline this program uses the output from toTabDelim.pl as input to search for the best hitting query for each subject and only display fasta from clusters that contained sequences that received best hits from sequences in the same cluster. Run with -h for more info on other uses as a standalone program.

Thus the scripts toTabDelim.pl and toFasta.pl filters clusters to allow only sequences in the alignment that had best bi-directional hits from sequences in the same cluster.

paraAlignCluster.pl:
Takes a fasta file containing headers formatted like ">[name],[cluster]" as input and then produce alignments from each cluster in interleave phyliip format.

gblocksWrapper.pl:
This program parses alignments by removal of poorly aligned positions and divergent regions at three different levels of strictness. This involved removal of all gap containing columns as one option and then processing by Gblocks at two different settings. Run with "-h" for more info regarding the Gblocks settings.

normalizeAlignments.py:
Goes through a list of alignments and makes sure that each alignment has the same organisms present as all of the other alignments. If this is not the case then empty sequences (missing characters represented by ?) are inserted so that this is true.

nogapFasta.pl:
Takes a fasta file containing headers formatted like ">[name],[cluster]" and a file with a list of clusters as input. The output is fasta only from the clusters in the list.

catPhylip.pl:
This program takes a list or a directory of alignments in interleaved phylip format to be concatenated into one super alignment.

paupWrapper.pl:
This wrapper script executes a batch of paup commands that will perform a parsimony analysis doing a heuristic search, evaluating 100 random-addition replicates (maximum trees = 100) and excluding uninformative characters. The number of bootstraps and outgroup are user defined variables.

phylipWrapper.pl:
This wrapper script executes a batch of phylip programs that will compute a distance matrix using the JTT amino acid substitution model with no variation among sites (protdist) using the Neighbor-Joining method of clustering (neighbor) and construct a strict majority rule consensus tree (consense).

paraProtTest.pl:
This script execute the ProtTest program for a user specified number of alignments in phylip format (interleave) using fast optimization. The raw output produce a table ranking models under all selection strategies with more details about models under the AIC framework for each alignment.

parseProtTest.pl:
As input it takes a directory of ProtTest result files with the file names formatted some_output_<clusternumber>.phy.out (format is the result of hal scripts) and the model selection strategy (framework) by which to summarize. The output is three files. See the section 'Interpreting the output' for a description of the prottest_*.txt files.

raxmlWrapper.pl:
This wrapper script execute the RAxML program which will perform a maximum likelihood analysis. User defined variables are the amino acid substitution model/s (partitioned file or model string), rate heterogeneity and bootstraps.

phymlWrapper.pl:
This wrapper script execute the PhyML program which will perform a maximum likelihood analysis. User defined variables are the amino acid substitution model for the super alignment and number of bootstraps. If the alignment length is very long eg: 100 000 characters, PhyML requires a machine with enough RAM.

unmapOldHal.pl:
Will take a map file and several files that have had their sequences names shorted to alias names and then produce output with the original names.