MIRA

Mira source files can be found at /home/scott/Downloads/mira-3.2.1; these include demos (see the "minidemo" directory).

Visit Mira's sourceforge documentation.

Running mira is a bit odd - you use either "condensed" or "extensive" command line switches, but you never specify input files. Input files must follow a prescribed naming convention, set by the job name (which you pick) and the type of data ("454", "solexa", "solid", etc.)

If you're assembling 454 data (ESPECIALLY mate-pair data), you must use the mira utility program sff_extract utility to convert the sff data into the format mira likes. Note that ALL your 454 reads must be in one file set (.fasta/.fasta.qual). If you have multiple library sizes, you must create an xml traceinfo file to specify each read's insert size.

For example, let's say you have the files 454Reads.MID5.sff and 454Reads.MID8.sff - to formulate the mira input files for the project "ct", you'd do:

```
sff_extract -s ct_in.454.fasta -q ct_in.454.fasta.qual -x ct_in.454.xml 454Reads.MID5.sff 454Reads.MID8.sff
```

and then to run mira you'd use something like:

```
mira --project=ct --job=denovo,genome,accurate,454 -SK:not=8 &> log_assembly.txt &
```

Getting .sff and Illumina .fastq files ready for Mira, and actually running Mira, can be automated using the a small script called "MiraWrapper.py". You provide MiraWrapper.py with your 454 and/or Illumina paired-end files, together with a project name and the Mira commands. If your 454 files are "454Reads.MID5.sff" and "454Reads.MID8.sff", and your paired-end Illumina files are "IlluminaReads1.fastq" and "IlluminaReads2.fastq", the syntax for running MiraWrapper is:

```
MiraWrapper.py --project ct --sff 454Reads.MID5.sff 454Reads.MID8.sff --pe IlluminaReads1.fastq IlluminaReads2.
fastq --mira --job=denovo,genome,accurate,454,solexa -SK:not=8 -notraceinfo &> log_assembly.txt &
```

CAUTION: Note that the project name is specified by `--project ct`, and not in the usual mira parameters (like `-project=ct`). This is necessary so that MiraWrapper.py can keep track of all the intermediate file names. Also, MiraWrapper does not yet support information about mate-pair data, so you must run Mira with the `-notraceinfo` option.

If you have mate-pair data, you have to give mira information about the library sizes during sff_extract, for example:

```
sff_extract -Q -a -x ct_traceinfo_in.454.xml -s ct_in.454.fastq -l ../linker.fasta -i "insert_size:2894,
insert_stdev:724" sff_dir/GXGRU2301_253.sff
```

To use consed to do editing & finishing after assembly, jump to Phred, Phrap, Consed, cross_match, daev.

Installation notes for the future:
On installation, mira required BOOST installed as: sudo ./bootstrap.sh --with-libraries=all --prefix=/usr/local, followed by sudo ./bjam threading=multi install, then mira needed: ./configure --with-boost-libdir=/usr/local/lib, followed by make/make install.