Appendix A

Aligning Multiple Whole Genomes with Mercator and MAVID

The availability of an increasing number of whole genome sequences presents us with the need for tools to quickly put them into a nucleotide-level multiple alignment. Mercator and MAVID are two programs that can be combined to accomplish this task. Given multiple whole genomes as input, Mercator is first used to construct an orthology map, which is then used to guide nucleotide-level multiple alignments produced by MAVID. These programs are both fast and freely available, allowing researchers to perform genome alignments on a single laptop. This tutorial will guide the researcher through the steps required for whole-genome alignment with Mercator and MAVID.

This tutorial will guide you through the process of aligning multiple whole genome sequences with Mercator [35] and MAVID [12]. Both programs are freely available and allow researchers to align moderately-sized genomes on a single laptop. The combination of Mercator and MAVID is an example of a *hierarchical* strategy for aligning genomes [36]. First, Mercator is used to construct an *orthology* map between the input genomes, which is a high-level one-to-one mapping between genomic segments. The second step is to run MAVID, a global multiple alignment program, on the sets of orthologous (and colinear) segments specified by the orthology map. The result is a set of multiple alignments with the property that every nucleotide is part of at most one multiple alignment.

For the tutorial, we will align the genomes of three fruit fly species: *Drosophila melanogaster*, *Drosophila yakuba*, and *Drosophila ananassae*. The genome sequences of the first two species are organized into chromosomes, while that of the third is currently comprised of over 10,000 unmapped scaffolds. The tutorial will begin with the downloading of the raw genome sequences. I will then describe how to prepare the genome sequences and create an orthology map between them using Mercator. Procedures for comparatively scaffolding genomes and discovering rearrangement breakpoints with Mercator will also be described. The tutorial will conclude with the generation of nucleotide-level alignments using MAVID, and the extraction of a specific interval from the resulting whole-genome alignment.

A.1 Materials

For the purposes of this tutorial, it is assumed that you are using a UNIX-like computing environment (e.g., Linux, Mac OS X, or Cygwin on Microsoft Windows). All software distributions listed in Table A.1 should be downloaded and compiled. Compiled binaries should all be made available through the `PATH` environment variable.

A.2 Methods

This tutorial will specify every command, in order, for the processing and alignment of three fruit fly genomes. Commands to be run will be specified by lines beginning with `$`. The output for some commands will be shown and truncated output will be indicated by ellipses (...). Approximate running times for selected commands will be specified as comments. Running times are for an Apple PowerBook with a 1.25 GHz PowerPC G4 processor and 1 GB of RAM.

We will begin by starting in an empty directory and creating subdirectories for the input and output files of the alignment process.

```
$ mkdir input
$ mkdir output
```

A.2.1 Obtaining Genome Sequences

Genome sequences can be obtained from many sources on the Internet. Most sources are either genome sequencing centers or databases that collect from many primary sources. We will download the *D. melanogaster* release 4 and *D. yakuba* release 2 assemblies from a database site, the UCSC Genome Browser [59]. The *D. ananassae* CAF1 assembly will be downloaded from the AAA *Drosophila* Web site. See Notes A.3.1 for additional information on obtaining sequence from the UCSC Genome Browser.

```
$ cd input
$ # Define a variable for the UCSC URL
$ GOLDENPATH=http://hgdownload.cse.ucsc.edu/goldenPath/
$ # Download the DroMel genome from UCSC (39 MB)
$ wget $GOLDENPATH/dm2/bigZips/chromFa.zip
...
$ mv chromFa.zip DroMel.zip
$ # Download the DroYak genome from UCSC (49 MB)
$ wget $GOLDENPATH/droYak2/bigZips/chromFa.tar.gz
...
$ mv chromFa.tar.gz DroYak.tar.gz
$ # Download the DroAna genome from AAA (317 MB)
$ wget http://rana.lbl.gov/drosophila/caf1/dana_caf1.tar.gz
...
$ mv dana_caf1.tar.gz DroAna.tar.gz
```

In the case that any of these assemblies are no longer found at the URLs cited above, I have placed copies of them at http://bio.math.berkeley.edu/mercator/tutorial/.

To check that the downloaded assemblies are valid and to get some basic statistics about them, we will use the `faCount`, `faLen`, and `stats` utilities (Mercator distribution). The `faCount` utility calculates nucleotide frequencies within each input FASTA record (chromosomes or contigs in our case) and the `faLen` utility simply outputs the length of each sequence. Combining `faLen` with `stats`, which calculates some basic descriptive statistics of a set of numbers, allows us to calculate useful statistics for the draft assembly of *Drosophila ananassae*.

```
$ unzip -p DroMel.zip | faCount
#seq    len     A       C       G       T       N       cpg
chr4    1281640 415025  225495  224520  416500  100     40533
chrM    19517   8152    2003    1479    7883    0       132
chrU    8724946 1494654 978040  986285  1522538 3743429 186802
...
$ tar zxOf DroAna.tar.gz dana/scaffolds.bases | faLen | stats
           N = 13749
         SUM = 230993012
         MIN = 55
1ST-QUARTILE = 1191
```

```
      MEDIAN = 1517
3RD-QUARTILE = 3575
         MAX = 23697760
        MEAN = 16800.7136519
         N50 = 4599533
```

From the output of the last command, we see that half of the bases in the *D. ananassae* assembly are in scaffolds of length 4,599,533 or greater (this is the N50 statistic for a genome assembly).

A.2.2 Preparing the Genome Sequences

Unfortunately, it often the case that two whole genome sequences downloaded from the Internet are in different formats, so some work must be done to prepare the sequences for alignment.

Masking Repeats

For the best genome annotations and alignments, the genome sequences must be "masked" for repeats. See Notes A.3.2 for details on the different ways in which a sequence can be masked. Fortunately for us, the sequences obtained from the UCSC Genome Browser Web site are already softmasked with RepeatMasker [97] and Tandem Repeats Finder [3]. For the *D. ananassae* sequence, we will need to do the masking ourselves. We will use the RepeatMasker program, as well as `nmerge` (WU-BLAST distribution, often required by RepeatMasker) and `faSoftMask` (Mercator distribution) utilities.

```
$ # Extract sequence for DroAna (1 min)
$ tar zxOf DroAna.tar.gz dana/scaffolds.bases > DroAna.fa.unmsk
$ # Mask interspersed repeats (19 hours)
$ ln -s DroAna.fa.unmsk DroAna.fa.int
$ RepeatMasker -no_is -nolow -species drosophila DroAna.fa.int
RepeatMasker version open-3.1.5
Search engine: WUBlast

analyzing file DroAna.fa.int
identifying matches to drosophila genus sequences in batch 1 of 6036
...
$ # Mask low complexity repeats (13 hours)
$ ln -s DroAna.fa.unmsk DroAna.fa.low
$ RepeatMasker -no_is -noint -species drosophila DroAna.fa.low
```

```
RepeatMasker version open-3.1.5
Search engine: WUBlast

analyzing file DroAna.fa.low
identifying simple repeats in batch 1 of 6036
identifying more simple repeats in batch 1 of 6036
identifying low complexity regions in batch 1 of 6036
...
$ # Merge masking into one hardmasked file (1 min)
$ nmerge DroAna.fa.int.msk DroAna.fa.low.msk > DroAna.fa.msk
$ # Create softmasked file (2 min)
$ faSoftMask DroAna.fa.unmsk DroAna.fa.msk > DroAna.fa
```

Creating Sequence Database Files

For efficiency purposes, we need to put our FASTA-formatted sequences into another format. I have developed a file format, the Sequence Database format (SDB), that allows for fast random access to multiple sequences stored in a single file. See Notes A.3.2 for descriptions of the command-line utilities available (as part of the Mercator distribution) for creating and accessing SDB files. We will use the `fa2sdb` utility to put our softmasked genomes into SDB format.

```
$ unzip -p DroMel.zip | fa2sdb -c DroMel.sdb
$ tar zxOf DroYak.tar.gz | fa2sdb -c DroYak.sdb
$ cat DroAna.fa | fa2sdb -c DroAna.sdb
```

To get a listing of the *D. melanogaster* chromosomes and their lengths, we can use the `sdbList` utility.

```
$ sdbList -l DroMel.sdb
chr2L   22407834
chr2R   20766785
chr2h   1694122
chr3L   23771897
...
```

To get the sequence from a specific genomic interval, we can use the `sdbExport` utility.

```
$ # Get sequence of 2nd coding exon of gene "dachshund"
$ sdbExport -r DroMel.sdb chr2L 16477453 16477480 -
>chr2L:16477453-16477480-
ATGCCTATCGATCAAGCCACCAGAAAG
```

A.2.3 Obtaining Gene Annotations

The simplest way to use Mercator for orthology map creation is to use coding exons as map *anchors*. Therefore, we need to obtain gene annotations for each of our genomes. For the *D. melanogaster* and *D. yakuba* genomes, we will simply download annotations. For the *D. ananassae* genome, we will have to produce our own annotations through the use of gene prediction software. See Notes A.3.3 for tips on obtaining annotations and details on the annotation format required by Mercator.

Let us first download annotations for *D. melanogaster* and *D. yakuba* from the UCSC Genome Browser and convert them to GFF using the utility program `ucsc2gtf` (Mercator distribution).

```
$ # Obtain annotations for DroMel
$ wget $GOLDENPATH/dm2/database/flyBaseGene.txt.gz
...
$ zcat flyBaseGene.txt.gz | ucsc2gtf flybase > DroMel.gff
$ # Obtain annotations for DroYak
$ wget $GOLDENPATH/droYak2/database/genscan.txt.gz
...
$ wget $GOLDENPATH/droYak2/database/xenoRefGene.txt.gz
...
$ zcat genscan.txt.gz | ucsc2gtf genscan > DroYak.gff
$ zcat xenoRefGene.txt.gz | ucsc2gtf xenoRefSeq >> DroYak.gff
```

Notice that we have combined two independent annotations of *D. yakuba* into one GFF file. You can use as many annotation sets as you like and, in fact, the more the better (sensitivity is all that matters).

Now we will generate an annotation of the *D. ananassae* genome using the SNAP [64] gene prediction program (wrapped by the `runSnap` script, Mercator distribution). The program `zff2gtf` (Mercator distribution) is used to convert from SNAP's ZFF format (Table A.3) to GFF.

```
$ # Run SNAP with D. melanogaster parameters (2 hours)
$ runSnap /usr/local/snap/HMM/fly < DroAna.fa.int.msk > DroAna.zff
$ cat DroAna.zff | zff2gtf --source=SNAP > DroAna.gff
```

A.2.4 Generating Input for Mercator

With SDB and GFF files for each genome in hand, we are now ready to generate the input files for Mercator. The easiest way to do this is with the `makeMercatorInput` script (Mercator distribution). We simply supply the names of the assemblies as arguments to this script. The `makeMercatorInput` script will look in the current directory for each genome's SDB and GFF file. See Notes A.3.4 for information regarding custom jobs with or without `makeMercatorInput`.

```
$ # Create input files for Mercator (15 min)
$ makeMercatorInput DroMel DroYak DroAna
Making chromosome file for DroMel...done
Making anchors for DroMel...done
Extracting protein sequences for anchors...done
Making chromosome file for DroYak...done
Making anchors for DroYak...done
Extracting protein sequences for anchors...done
Making chromosome file for DroAna...done
Making anchors for DroAna...done
Extracting protein sequences for anchors...done
BLATing anchors pairwise...
DroMel-DroYak
Loaded 10029188 letters in 98948 sequences
Searched 7247210 bases in 53254 sequences
...
```

This script performs the following tasks:

1. Creates a file for each genome specifying the names and lengths of the sequences that make up that genome.

2. Creates a set of non-overlapping anchor intervals for each genome from the CDS records of the GFF files.

3. Creates a file for each genome of the protein sequences coded for by each of the anchor intervals.

4. Compares the protein sequences of each genome pairwise using the BLAT [61] program to create "hit" files.

Also required by some components of Mercator and by MAVID is a phylogenetic tree relating the input species. The branch lengths of the tree should be the expected number of substitutions per site along each branch. The tree must be in Newick format (Table A.3). We will put our tree in the file `treefile`.

```
$ echo "((DroMel:0.1,DroYak:0.1):0.4,DroAna:0.6);" > treefile
```

A.2.5 Constructing an Orthology Map with Mercator

Running Mercator is simple and fast once all of the input files have been generated. Because the *D. ananassae* assembly is still in scaffolds, we will tell Mercator that it should be treated as a draft genome by using the `-d` flag.

```
$ cd ..
$ mercator -i input -o output DroMel DroYak -d DroAna
...
Loading input files...
Loading chromosome files...
DroMel 13 chromosomes
DroYak 21 chromosomes
DroAna 13749 contigs
Loading anchor files...
DroMel 53254 anchors
DroYak 98948 anchors
DroAna 89541 anchors
Loading hit files...
DroMel-DroYak 75082 hits (2380 filtered)
DroAna-DroMel 75397 hits (4355 filtered)
DroAna-DroYak 110120 hits (3324 filtered)
Sorting edges...
Time spent loading files: 16 seconds
Making map...
...
Assembling draft genomes...
Number of runs: 1177 (using 46614 cliques)
Checking cliques...
Map-making completed
Number of runs: 1177
Number of cliques: 46614
Mean run length: 39.6041
Median run length: 19
Max run length: 513
Min run length: 1
Coverage of DroMel anchors: 98.4133% (52409/53254)
Coverage of DroYak anchors: 81.5964% (80738/98948)
Coverage of DroAna anchors: 81.738% (73189/89541)
Writing coverage files...
Coverage of DroMel: 82.3921%
Coverage of DroYak: 69.2232%
Coverage of DroAna: 58.1449%
...
```

```
Run time: 38 seconds
$ # Mercator has finished, let us look at the output files
$ cd output
$ ls
DroAna.agp          DroMel.coverage    genomes
DroAna.anchors      DroMel.mgr         map
DroAna.coverage     DroYak.agp         pairwisehits
DroAna.mgr          DroYak.anchors     runs
DroMel.agp          DroYak.coverage    pre.map
DroMel.anchors      DroYak.mgr
```

After running the main Mercator program, we now have an orthology map where the orthologous intervals are defined by the boundaries of the landmarks in the file "pre.map" and a map with the breakpoint regions cut in half in the file "map." See Notes A.3.5 for more details on Mercator.

A.2.6 Comparatively Scaffolding Draft Genomes

When a genome is specified as "draft" to Mercator (using the `-d` option), the program will attempt to comparatively scaffold that genome's component sequences. That is, it uses information from the other genomes to orient and join the draft genome's contigs or scaffolds. Mercator specifies the comparative scaffolding of a draft genome in the form of an AGP file (Table A.3). Later steps in the alignment process will not be aware of comparative scaffolding, so we must provide updated SDB files for each genome. In our alignment, *D. ananassae* has been comparatively scaffolded by Mercator, so we must "assemble" its component sequences into a new SDB file using the `sdbAssemble` program (Mercator distribution). For the other genomes, we will simply make a link to original SDB files. See Notes A.3.6 for additional details on the comparative scaffolding aspect of Mercator.

```
$ sdbAssemble ../input/DroAna.sdb DroAna.sdb < DroAna.agp
$ ln -s ../input/DroMel.sdb
$ ln -s ../input/DroYak.sdb
```

A.2.7 Refining the Map via Breakpoint Finding

Because Mercator has only used exons as landmarks for determining orthologous segments, the exact boundaries of the orthologous segments are not yet determined. If we wish to refine the boundaries of the identified orthologous segments, we can use the breakpoint finding program included in the Mercator distribution. This program attempts to find the best position within each "breakpoint region" (intervals in between segments identified in the "pre.map") at which to break and add the left and right intervals to the flanking segments. This procedure may be skipped if the exact boundaries of the segments are not required. Locating breakpoints involves a number of steps. Note that SDB files for each genome must be present in the current directory (`output`), as set up in the last section. See Notes A.3.7 for additional information on the breakpoint finding process.

```
$ # The breakpoint finding algorithm requires the tree
$ ln -s ../input/treefile
$ # Convert the orthology map into a more general homology map
$ omap2hmap genomes < pre.map > pre.h.map
...
$ # Create the graph relating the breakpoint regions
$ makeBreakpointGraph pre.h.map treefile
$ # Make pairwise alignments for breakpoint regions (2 hours)
$ mkdir bp_alignments
$ makeBreakpointAlignmentInput --out-dir=bp_alignments
$ mavidAlignDirs --init-dir=bp_alignments
$ # Find a good configuration of breakpoints (8 min)
$ findBreakpoints pre.h.map treefile edges bp_alignments > breakpoints
$ # Refine the map by splitting the breakpoint regions
$ breakMap breakpoints < pre.h.map > better.h.map
$ # Convert back to the orthology map format
$ hmap2omap genomes < better.h.map > better.map
```

A.2.8 Generating Input for MAVID

Now that we have an orthology map, we are ready to run a global multiple alignment program on each orthologous segment set identified by the map. To help in the alignment process, we will give the alignment program a set of "constraints": short intervals within the orthologous segments that we know should be aligned. These constraints are derived from the sequence similarities identified between the anchors given to Mercator. To make the `constraints` file, we run the following command:

```
$ # Convert pairwise hits to alignment constraints (2 min)
$ phits2constraints -i ../input < pairwisehits > constraints
```

The input files for MAVID are then generated by `makeAlignmentInput`.

```
$ # Create directories and files for alignment (3 min)
$ mkdir alignments
$ makeAlignmentInput --map=better.map . alignments
```

See Notes A.3.8 for information on the input files that are required for MAVID and that are generated by `makeAlignmentInput`.

A.2.9 Aligning Orthologous Segments with MAVID

With the input for MAVID generated, all that is left is to run MAVID on the sequences for each orthologous segment set. Each segment set is stored in a separate subdirectory. This is a good step at which to parallelize, but if that is not an option, the `mavidAlignDirs` script (Mercator distribution) can be used. See Notes A.3.9 for details on the nucleotide-level alignment step.

```
$ # Align all sequence files in directory structure (13 hours)
$ mavidAlignDirs --init-dir=alignments
```

We now have a multiple whole-genome alignment of *Drosophila melanogaster*, *Drosophila yakuba*, and *Drosophila ananassae*.

A.2.10 Extracting Subalignments

We may now extract parts of the whole-genome alignment that are of particular interest using the `sliceAlignment` program (Mercator distribution). For example, we may wish to get the alignment of the second coding exon of the gene *dachshund*. The `sliceAlignment` program outputs alignments in multi-FASTA format, so we will use the `fa2clustal` utility (Mercator distribution) to put the exon alignment into a more readable form. See Notes A.3.10 for more details on `sliceAlignment`.

```
$ sliceAlignment alignments DroMel chr2L 16477453 16477480 - > exon.mfa
$ fa2clustal < exon.mfa
CLUSTAL

DroMel          ATGCCTATCGATCAAGCCACCAGAAAG
DroYak          ATGCCTATCGATCAAGCCACCAGAAAG
DroAna          ATGCCTATCGATCAAGCCACCAGAGAG
                ************************ **
```

A.3 Notes

A.3.1 Obtaining Genome Sequences

To download genomes from the UCSC Genome Browser, it is easiest to go through the "Downloads" section of the Web site. For the assembly of interest, click on the "Full data set" link to access complete genome sequences as compressed FASTA files (A.3). A selection of Web sites that provide whole genome sequences is given in Table A.2.

A.3.2 Preparing the Genome Sequences

Masking Repeats

An "unmasked" FASTA formatted file has all characters in uppercase. A masked sequence can either be "hardmasked" or "softmasked." In hardmasked files, characters that are part of repetitive sequence are changed to `N`'s, while in softmasked files they are changed to lowercase. Unmasked and softmasked sequence may also have `N`'s, which are commonly used to indicate assembly gaps. Ideally, we would like our genome sequences to be softmasked, so that we have repeat annotations as well as full sequence information.

Masking repeats is a bit of an art, and I will not go into all of the details here. Very briefly, one needs to mask both *interspersed* and *simple* (or *low complexity*) repeats. Masking of these two types of repeats should be done separately because gene finding is best done on sequence hardmasked for interspersed repeats (simple repeats can occur within genes).

Creating Sequence Database Files

There are four command-line utilities made available in the Mercator distribution for handling SDB files. The Mercator library code may also be used for writing C++ programs that access SDB files directly. The command-line utility `fa2sdb` is used to create or append to a SDB file from sequence records in FASTA format. DNA sequences may be compressed (2 nucleotides per byte) inside of a SDB file if the `-c` option is specified. The `sdbExport` utility is used for the extraction of specific genomic intervals from a SDB file. It can extract one or more intervals at a time and outputs sequences in FASTA format. The `sdbList` utility is used to list the names and lengths (with the `-l` option) of the records inside of a SDB file. Lastly, the `fa2sdb` utility is used to convert a SDB file into FASTA format.

A.3.3 Obtaining Gene Annotations

Gene annotations for many genomes can be obtained at the same database sites that provide whole genome sequences. For the UCSC Genome Browser site, annotations can be obtained either through the "Table Browser," or directly from the "Downloads" section. If annotations are not available online, you can produce them using gene prediction software. The easiest prediction programs to use in this case are single-genome *ab initio* gene finders (e.g., geneid [47], GENSCAN [17], and SNAP [64]). Regardless of how the annotations are obtained, they need to be converted to the GFF format (Table A.3). Three scripts (`genscan2gtf`, `ucsc2gtf`, and `zff2gtf`) in the Mercator distribution are available for converting to GFF from some common formats. Mercator requires that GFF annotations have CDS records (lines with "CDS" in the feature field) for the coding intervals of each exon. It is critical that the "frame" field be specified for each CDS record in the GFF files. This field allows Mercator to translate each coding exon correctly.

A.3.4 Generating Input for Mercator

For custom jobs (e.g., to parallelize some tasks), you may wish to generate the input for Mercator without using the `makeMercatorInput` script. In such cases, consult the `README` file in the Mercator distribution for exact specifications of the various input files that are required. Some routines of `makeMercatorInput` are customizable via command-line options. Use the use `--help` option to get full usage information.

A.3.5 Constructing an Orthology Map with Mercator

Mercator has a number of user-settable parameters that may be specified as command-line options. The options that affect Mercator's performance are `--min-run-length`, `--prune-pct`, `--join-distance`, `--max-eval`, `--repeat-num`, and `--repeat-pct`. Consult the Mercator README file for descriptions of these options.

A.3.6 Comparatively Scaffolding Draft Genomes

When Mercator comparatively scaffolds the components of a "draft" genome, it joins components that it believes should be adjacent to each other into new sequences with names beginning with `assembled`. For example, in our fruit fly alignment, the `scaffold_13770`, `scaffold_13165`, and `scaffold_13337` sequences from the *D. ananassae* assembly are joined into a new sequence called `assembled6`, with a string of `N`s separating the component sequences within `assembled6`. The number of separating `N`s may be specified by Mercator's `--padding` command-line option. These `N`s are meant to indicate gaps of unknown length between the component sequences.

A.3.7 Refining the Map via Breakpoint Finding

The breakpoint finding process can be very computationally intensive, depending on the input genomes. If a cluster is available to the user, it is a good idea to parallelize the `mavidAlignDirs` step. When running the `findBreakpoints` program, accuracy may be traded for speed via the `--resolution` option. Breakpoints will be found more accurately with larger "resolution" values.

A.3.8 Generating Input for MAVID

MAVID requires, at a minimum, three input files. These files are a phylogenetic tree in Newick format, unmasked sequences in a multi-FASTA file, and a hardmasked version of the multi-FASTA file. When Mercator is used, alignment constraints may be given to MAVID via the `-c` command-line option. In this tutorial, the `makeAlignmentInput` and `mavidAlignDirs` take care of generating and passing the correct files to MAVID.

A.3.9 Aligning Orthologous Segments with MAVID

Although the focus of this tutorial is on the application of Mercator and MAVID, the hierarchical strategy for whole-genome alignment allows for the components to be substituted with similar programs independently of each other. For example, in cases where the orthologous segments are very small, CLUSTAL W [104] could be used to do the multiple nucleotide alignment instead of MAVID. However, there is a significant advantage to using MAVID as the nucleotide-level aligner with Mercator: alignment constraints. By using the alignment constraints output by Mercator, MAVID can more accurately align coding regions and is able to process longer sequences.

A.3.10 Extracting Subalignments

The `sliceAlignment` program is designed to efficiently extract subalignments from a multiple whole-genome alignment. It extracts alignments based on the coordinates given as input for a specified reference genome. A single interval may be given as command line arguments or multiple intervals can be given on the standard input. With multiple intervals as input, the program will be very efficient if the intervals are sorted by their start coordinates.

A.4 Concluding Remarks

This tutorial has taken you through the basic steps of creating a multiple whole-genome alignment using Mercator and MAVID. There are many additional details and options that have been left out of this tutorial at each step. More details are available in the full documentation of each of the programs.

Mercator	`http://bio.math.berkeley.edu/mercator/`
MAVID	`http://bio.math.berkeley.edu/mavid/`
RepeatMasker	`http://www.repeatmasker.org/`
WU-BLAST	`http://blast.wustl.edu/`
SNAP	`http://homepage.mac.com/iankorf/`
BLAT	`http://www.cse.ucsc.edu/~kent/`

Table A.1: Web sites of programs for the alignment of multiple whole genomes.

AAA (*Drosophila*)	`http://rana.lbl.gov/drosophila`
UCSC Genome Browser	`http://genome.ucsc.edu`
NCBI	`http://www.ncbi.nlm.nih.gov`
Ensembl	`http://www.ensembl.org`

Table A.2: Web sites providing whole genome sequences.

AGP	`http://www.ncbi.nlm.nih.gov/Genbank/WGS.agpformat.html`
FASTA	`http://bioperl.org/wiki/FASTA_sequence_format/`
GFF	`http://www.sanger.ac.uk/Software/formats/GFF/`
Newick	`http://evolution.genetics.washington.edu/phylip/newicktree.html`
ZFF	`http://bioperl.org/wiki/ZFF`

Table A.3: File formats used by Mercator and MAVID.