Module 5

Part I. Gene Prediction

Goal: Learn to utilize the command line programs **SNAP** and **AUGUSTUS** to search genomes for predicted genes.

5.1.1 SNAP

SNAP is an *ab initio* gene-finding program that uses a hidden Markov model to search for regions of a genome that are likely to be genes. **SNAP** and other hidden Markov model gene-finders use statistical properties of the DNA sequence to infer features such as start codons, splice junctions, and untranslated regions.

Korf, I. Gene finding in novel genomes. *BMC Bioinformatics* 5:59, 2004.

https://github.com/dzerbino/velvet/tree/master

To predict genes well, these methods need a description of the statistical parameters of the model: that is, what do genes look like in the organism under study?

SNAP ships with a number of parameter files (models) for existing organisms and can be trained on other organisms.

☐ If we run in screen we need to provide a special argument that tells screen to begin **bash** as a login shell. This is required so that environment variables needed by the programs we will be using are set correctly (**note the bash option is a lowercase L**).

```
• screen -S genes bash -l
```

☐ First, we'll list the available models. The environment variable `ZOE` says where **SNAP** is installed:

- `echo $ZOE`

☐ Look inside that directory:

- `ls /usr/share/snap`

☐ The HMM directory contains the parameter files:

- `ls /usr/share/snap/HMM`

☐ Look at the contents of one of the parameter files:

- `less /usr/share/snap/HMM/C.elegans.hmm`

SNAP parameter files typically have the *.hmm* extension, although, as you can see from files such as *fly*, *rice*, and *thale*, the extension is not necessary. It doesn't look like any of the parameter files are particularly closely related to our organism (a fungus), so we will have to train our own.

5.1.2 Preparing training data

Goal:	Use annotations from a related genome to train **SNAP** to recognize *M. oryzae* genes.
Input):	`FH_V2_maker.gff (a list of gene feature coordinates in a previously annotated genome of M. oryzae)`
Output):	`Moryzae.hmm`

For our training data, we will use gene annotations from the closely related FH_V2 genome. These annotations were generated using the **MAKER** gene annotation pipeline, which we will discuss in detail later. The annotation file is in a standard format, known as the General Feature Format version 3 (GFF3).

The columns in the GFF3 format are as follows:

1. Seqid — Sequence ID (e.g., chromosome name)
2. Source — The source of the annotation (e.g., Ensembl, RefSeq, maker)
3. Type — Feature type (e.g., gene, mRNA, exon, CDS)
4. Start — Start coordinate of the feature (1-based)
5. End — End coordinate
6. Score — A floating-point score indicating feature quality (based on evidences)
7. Strand — +, -, or . (unknown)
8. Phase — Reading frame of coding sequences (0-indexed)
9. Attributes — A semicolon-separated list of key=value pairs (e.g., ID=gene0001;Name=BRCA1)

☐ Let's take a quick look at the FH_V2_maker.gff file so we understand the gff3 format. We will encounter it later:

- `grep maker FH_V2_maker.gff | head -n 15 | awk -F ";" '{print $1}'`

```
contig00823     maker   gene    2339    8274    .       -       .       ID=maker-contig00823-fgenesh-gene-0.0
contig00823     maker   mRNA    2339    8274    .       -       .       ID=maker-contig00823-fgenesh-gene-0.0-mRNA-1
contig00823     maker   exon    2339    7627    76.62   -       .       ID=maker-contig00823-fgenesh-gene-0.0-mRNA-1:exon:4187
contig00823     maker   exon    7705    8274    1030.88 -       .       ID=maker-contig00823-fgenesh-gene-0.0-mRNA-1:exon:4188
contig00823     maker   CDS     2339    7627    .       -       0       ID=maker-contig00823-fgenesh-gene-0.0-mRNA-1:cds:4182
contig00823     maker   CDS     7705    8274    .       -       0       ID=maker-contig00823-fgenesh-gene-0.0-mRNA-1:cds:4181
contig00638     maker   gene    4598    5121    .       +       .       ID=maker-contig00638-snap-gene-0.12
contig00638     maker   mRNA    4598    5121    .       +       .       ID=maker-contig00638-snap-gene-0.12-mRNA-1
contig00638     maker   exon    4598    4768    31.533  +       .       ID=maker-contig00638-snap-gene-0.12-mRNA-1:exon:3768
contig00638     maker   exon    4870    5121    34.173  +       .       ID=maker-contig00638-snap-gene-0.12-mRNA-1:exon:3769
contig00638     maker   CDS     4598    4768    .       +       0       ID=maker-contig00638-snap-gene-0.12-mRNA-1:cds:3758
contig00638     maker   CDS     4870    5121    .       +       0       ID=maker-contig00638-snap-gene-0.12-mRNA-1:cds:3759
contig00638     maker   gene    9497    10375   .       +       .       ID=maker-contig00638-fgenesh-gene-0.1
contig00638     maker   mRNA    9497    10375   .       +       .       ID=maker-contig00638-fgenesh-gene-0.1-mRNA-1
contig00638     maker   exon    9497    10375   49.82   +       .       ID=maker-contig00638-fgenesh-gene-0.1-mRNA-1:exon:3770
```

Here we can see information about features associated with three genes on contigs 00823 and 00638. We see each gene and corresponding transcript's span, as well as the positions of their respective exons. If desired, we can use this information to create of diagram of each gene's position on the contig and its precise structure.

To start our gene prediction exercises, we will work in the *snap* subdirectory because **SNAP** training produces many files, and we want to avoid clutter.

☐ Change to the *snap* directory under *genes* where we'll keep our intermediate results and output.

We have a FASTA file with the genome sequences and a GFF file from **MAKER** with both gene annotations and genome sequences. However, **SNAP** requires the training genes be in a custom format (used only by **SNAP**) called "ZFF". Fortunately, **MAKER** comes with a script to convert its annotations into ZFF format.

☐ Convert the **MAKER** annotations to ZFF for **SNAP:**

- `maker2zff FH_V2_maker.gff`

The command above creates files *"genome.ann"* (ZFF format) and *"genome.dna"* (FASTA format) in the current directory. The *.ann* file represents the positions of exons and genes within each contig, while the *.dna* file contains simply the contig sequences.

☐ Now we can use **fathom**, a sequence analysis and extraction tool that comes with **SNAP**, to extract the gene sequences from these two files.

Usage: `fathom <annotations> <contigs> -<subcommand> [options]`

The **fathom** tool does not require specific file extensions; however, the annotations file must be in ZFF format, and the contigs file must be in FASTA format.

A number of subcommands are available; we will use only a few. For more information, you can run

`fathom -help | less`.

☐ First, get some information on the annotations with `-gene-stats`. This subcommand outputs to the screen the number of sequences (contigs), the number of genes annotated, GC content, average intron and exon lengths, and more.

```
• fathom genome.ann genome.dna -gene-stats
```

Note: It is expected to see error messages about one or two gene models; this can occasionally happen when the annotations contain overlapping genes on different strands.

☐ Now we will extract the genome regions containing unique genes using the `-categorize` subcommand. This subcommand creates a number of pairs of ZFF and FASTA files: one pair for regions with errors, one for regions with overlapping genes, and so on. We will ask for up to 1000 base pairs of intergenic sequence on both sides of each gene to help train the HMM about what sequences are likely to occur near genes.

```
• fathom genome.ann genome.dna -categorize 1000
```

Note: Once again, it is expected to see error messages about one or two gene models.

☐ Look at the contents of the directory; there are now a number of pairs of *.ann* (ZFF format) and *.dna* (FASTA format) files. Look at `fathom -help` again to see what these categories mean.

☐ Training **SNAP** usually works best with unique, non-overlapping genes without alternative splicing; these annotations may be found in *uni.ann*, and the accompanying sequences in *uni.dna*. Let's look at their statistics again:

```
• fathom uni.ann uni.dna -gene-stats
```

☐ Next, we will use the `-export` subcommand to extract the genome, transcript, and protein sequences from these genes. We must tell `-export` as well to keep 1000 base pairs of context and also (with `-plus`) to flip genes that are on the reverse strand (that is, with their 3' end nearer the beginning of the contig than their 5').

```
• fathom uni.ann uni.dna -export 1000 -plus
```

The output is in four files:

export.ann	ZFF annotations for the exons of each gene.
export.dna	DNA sequence for each gene, including introns and flanking regions.
export.tx	DNA sequence for each transcript.
export.aa	Protein sequence for each gene.

☐ Use `-gene-stats` again to review the statistics of the *export.ann* and *export.dna* files again; the statistics should be very close, if not identical, to those for *uni.ann*.

☐ Finally, we have the data in suitable format and are ready to train the HMM. The **forge** tool does this part of the process.

- `forge export.ann export.dna`

☐ After a few seconds, you will have a large number of *.model* and *.count* files representing the model parameters: probabilities and frequencies of different types of nucleotide sequences and gene features. List the directory to check that the files were created.

☐ The number of files is too unwieldy for most uses, so we will use one final **SNAP** tool, **hmm-assembler.pl** to condense everything into a single file for use with runs of **SNAP**.

- `hmm-assembler.pl Moryzae . > Moryzae.hmm`

The first argument, `Moryzae`, is the name to use in the header of the output, mostly for identification purposes; it should be a sequence of letters. The second argument is the directory in which to find all of the *.model* and *.count* files—here, the current directory. The program writes the model to standard output, so we use ">" to redirect it to a file.

5.1.3 Gene calling with SNAP

Goal:	Predict genes using **SNAP**.
Input(s):	`magnaporthe_oryzae_70-15_8_supercontigs.fasta` `Moryzae.hmm`
Output(s):	`magnaporthe_oryzae_70-15_8_single_contig-snap.gff2` `magnaporthe_oryzae_70-15_8_single_contig-snap.zff`

Now that we have our parameter file *Moryzae.hmm*, we can get to the business of predicting genes. We'll test **SNAP** out on just a single contig.

☐ Extract a single contig from your *Magnaporthe oryzae* assembly located in the *~/blast* directory into a file. The instructions following the creation of the single contig file (see below) assume that the file is in *magnaporthe_oryzae_70-15_8_single_contig.fasta*, but you are welcome to use a different name or location.

- `samtools faidx ~/blast/magnaporthe_oryzae_70-15_8_supercontigs.fasta Chromosome_8.7 > magnaporthe_oryzae_70-15_8_single_contig.fasta`

After specifying the filename and location, we specify the id of the sequence we want to extract. The id of our single contig is >`Chromosome_8.7`. **Note** that the ">" at the beginning of the sequence ID is not included in the command.

Finally, we redirect the output to *magnaporthe_oryzae_70-15_8_single_contig.fasta* using ">".

☐ To run **SNAP**, give it the name of your parameter file and your FASTA file. Output goes to standard output, so you probably want to redirect it:

```
• snap-hmm Moryzae.hmm
  magnaporthe_oryzae_70-15_8_single_contig.fasta
  > magnaporthe_oryzae_70-15_8_single_contig-snap.zff
```

☐ Look at the resulting *magnaporthe_oryzae_70-15_8_single_contig-snap.zff*. Use **fathom** again to compute its statistics, using the sequence file *magnaporthe_oryzae_70-15_8_single_contig.fasta*.

```
• fathom magnaporthe_oryzae_70-15_8_single_contig-snap.zff
  magnaporthe_oryzae_70-15_8_single_contig.fasta -gene-stats
```

☐ The default output is in ZFF format, which can be used as input for training **SNAP**, but must be converted to work with most other programs. It is also possible to generate a GFF file in the older GFF2 format:

```
• snap-hmm Moryzae.hmm
  magnaporthe_oryzae_70-15_8_single_contig.fasta -gff >
  magnaporthe_oryzae_70-15_8_single_contig-snap.gff2
```

☐ Look at the resulting file: *magnaporthe_oryzae_70-15_8_single_contig-snap.gff2*. Compare the format to that of the GFF3 file: *FH_V2_maker.gff*. Note that it is very similar but column 9 has a lot more information in the GFF3 file.

5.2.1 AUGUSTUS

Goal:	Predict genes using **AUGUSTUS**.
Input(s):	`magnaporthe_oryzae_70-15_8_single_contig.fasta`
Output(s):	`magnaporthe_oryzae_70-15_8_single_contig-augustus.gff3`

AUGUSTUS is another gene finder, similar in principle to **SNAP**. It uses a similar, but distinct, hidden Markov model to predict genes. Although we will not explore this use, **AUGUSTUS** is capable of incorporating information such as protein alignments into its model's parameters. **AUGUSTUS** is free for all uses and comes with a rather large collection of parameter files, trained on various species.

M. Stanke , O. Schöffmann , B. Morgenstern, S. Waack (2006) Gene prediction in eukaryotes with a generalized hidden Markov model that uses hints from external sources. BMC Bioinformatics 7:62.

http://bioinf.uni-greifswald.de/augustus/

5.2.2 Running AUGUSTUS

- ☐ Change to the **AUGUSTUS** directory in ~/*genes*/*augustus*, which is where we'll keep our intermediate results and output.
- ☐ First, let's look at the help:

```
• augustus --help 2>&1 | less
```

Here we are simply piping the output of **augustus** --help into the **less** text viewer. So, why do we need the `2>&1` in the command above? Some programs, including, **augustus** send certain messages to a stream known as stderr instead of stdout, and only stdout can be redirected by a normal pipe. The 2>&1 allows us to redirect the stderr message (2) to stdout, which we can then pipe into **less**.

- ☐ Let's look at the list of parameter files.

```
• augustus --species=help 2>&1 | less
```

We see that among the listed organisms is *Magnaporthe grisea*. This organism is very closely related to ours, so we won't have to retrain **AUGUSTUS**.

- ☐ Make sure you are still in a screen

```
• echo $STY
```

If you are in a screen session, this will list with a number in front of the name of the active screen:

```
630678.genes
```

☐ If you do not see a screen listed start a new one named augustus:

```
• screen -S augustus bash -l
```

☐ Running **AUGUSTUS** is similar to running other gene finders, although, as usual, the details are different. Rather than specifying a parameter file explicitly, you use the name of one of the included species listed by the above commands.

```
• augustus --species=magnaporthe_grisea --gff3=on
  --singlestrand=true --progress=true
  ../snap/magnaporthe_oryzae_70-15_8_single_contig.fasta
  > magnaporthe_oryzae_70-15_8_single_contig-augustus.gff3
```

This step takes around 6-8 minutes. (If you are not already, you may want to use **screen** to protect against disconnection.)

We give **AUGUSTUS** several parameters here:

`--species=magnaporthe_grisea`	Specifies the parameter file to use. Run `augustus --species=help` for a list.
`--gff3=on`	Produce GFF3-format output. The default is GTF.
`--singlestrand=true`	Predict genes on each strand separately; this option allows **AUGUSTUS** to find genes which overlap on opposite strands.
`--progress=true`	Show a progress bar for each step of the gene finding process. There are two steps per contig, or twice that with the option `--singlestrand=true`. **AUGUSTUS** breaks up contigs larger than about 100 kb and considers each portion separately, which may result in more steps.
`../snap/magnaporthe_oryzae_70-15_8_single_contig.fasta`	The last argument is the name of the genome sequence FASTA file.
`> magnaporthe_oryzae_70-15_8_single_contig-augustus.gff3`	Like many programs, **AUGUSTUS** writes its results to standard output, so you most likely want to redirect to a file. The progress bars and error messages will still be visible.

- ☐ Look at the output file to see the results. In addition to describing the locations and relationships of predicted genes and their features, this GFF file also includes the inferred protein sequence for each predicted gene.

- ☐ Take a look at the GFF files created by **SNAP** and **AUGUSTUS**.

Note how **SNAP** produces a rather basic gff2 report which simply lists initial exons (Einit), internal exons (exon), or terminal exons (Eterm). On the other hand, **augustus** produces an output that is much more feature-rich and more easily interpretable.

Note: If you are interested in training AUGUSTUS for your own organism, the easiest way is to use the AUGUSTUS web server:

http://bioinf.uni-greifswald.de/webaugustus/training/create

Select "AUGUSTUS training submission" and follow the directions; a tutorial and sample data are provided on the web site. Training requires a genome FASTA file, and either a list of cDNA or protein sequences from your organism, or a GenBank-format file with gene annotations. When the run completes, you will receive a file *parameters.tar.gz* containing the parameter files for your species. You may then download and build AUGUSTUS, and extract this file into the *config/species/* directory; if it worked correctly, your species will appear in the output of `augustus --species=help`.

Part II. Combining evidences with MAKER

Goal:	Use **MAKER** to prepare and execute a gene prediction and annotation workflow.

Input(s):	`magnaporthe_oryzae_70-15_8_single_contig.fasta`
	`genes/snap/Moryzae.hmm`
	`genes/maker/Moryzae_RNAseq.gtf`
	`genes/maker/genbank/ncbi-cds-Magnaporthe_organism.fasta`
	`genes/maker/genbank/ncbi-protein-Magnaporthe_organism.fasta`

Output(s):	`maker-annotations.gff3` `maker_proteins.fasta`

MAKER is a gene annotation pipeline that integrates the results of various gene finding programs, along with EST and protein sequences, to produce a single consistent set of annotations that takes both predictions and evidences into account.

Cantarel BL, Korf I, Robb SM, Parra G, Ross E, Moore B, Holt C, Sánchez Alvarado A, Yandell M. (2008) MAKER: an easy-to-use annotation pipeline designed for emerging model organism genomes. Genome Res. 18: 188–196.

http://www.yandell-lab.org/software/maker.html

5.3 Running MAKER

MAKER has a very large assortment of options—too many to specify everything on the command line. Instead, we can control **MAKER** through a set of configuration files. The first step in running **MAKER** is to create the configuration files:

- ☐ Change to your *~/genes/maker* directory.
- ☐ List the contents of the directory. You will see a *genbank* directory, a gff file and a merged
- ☐ Create the **MAKER** configuration files:

```
• maker -CTL
```

This last command generates three files:

maker_exe.ctl	lists the locations of the programs **MAKER** uses.
maker_bopts.ctl	sets thresholds for accepting alignments of EST and protein evidence
maker_opts.ctl	Describes the data to be used as input to **MAKER**, the steps to perform, and options for those steps.

The first two files do not change much from one run to another; we will not be changing them, though it doesn't hurt to take a look at their contents. All of our configuration will take place in *maker_opts.ctl*.

☐ Open *maker_opts.ctl* with a text editor such as nano.

Do not be alarmed by the several pages of options; we will be changing only a few of them. Each option has the format:

```
Option=value
```

Number signs ("#") mark the beginning of a comment that extends to the end of the line. Each option has a short comment explaining its meaning. When editing it doesn't matter whether you keep the comments or not but keeping them might be helpful if you later need to remember what the option does.

There are a few options we can set now.

☐ Find the lines for the following options and edit them as shown, changing directory and file names as appropriate to match the file paths in your workspace.

Hint: To navigate quickly in nano, use Ctrl-W followed by the option name to find the proper option to be edited. For example, for the first option type Ctrl-W `genome` to see all the places where the word genome is located and find the correct option.

genome=*/home/yourusername/genes/snap/magnaporthe_oryzae_70-15_8_single_contig.fasta*

The FASTA file of genome contigs to annotate. We will use just a single contig for our run. You must specify the full path to the file, including your home directory.

model_org= must be set to blank

The organism to use for repeat masking. We want to leave this option blank because masking can take a long time, so we will remove the default "all" setting, leaving the value blank, to turn off this step. Note that the equal sign is still required!

repeat_protein= must be set to blank

If this value is not already blank, disable searching for repeat proteins by removing anything that follows "=." Otherwise, this will greatly increases the running time.

snaphmm=*/home/yourusername/genes/snap/Moryzae.hmm*

The parameter file to use for **SNAP**.

augustus_species=magnaporthe_grisea

The model organism (parameter file) to use for **AUGUSTUS**.

est2genome=1

Infer predictions from protein homology, 1 = yes, 0 = no

keep_preds=1

This option keeps *ab initio* gene predictions in the output, even if there is no EST or related-organism evidence for those predictions.

☐ Save your updated *maker_opts.ctl.*

We will also use evidence from our organism and related ones to help **MAKER** improve its annotations. "Evidence" here refers to new Expressed Sequence Tags (ESTs/RNAseq data) and

protein sequences as well as pre-computed EST alignments. **MAKER** accepts sequences in FASTA format and gene annotations in GFF version 3 format.

- ☐ We will provide **MAKER** with a .gtf file that contains the reference gene set (i.e., the best guess at gene predictions for *M. oryzae),* which has been enriched with RNAseq data. These data not only provide support for most of the predicted genes, but they also reveal chromosome regions that are expressed but were not previously predicted (i.e. cryptic genes).
- ☐ Convert the *merged.gtf* file in the current directory into GFF3 format. The script **gtf2gff3.pl** can do this for us:

```
☐ gtf2gff3.pl merged.gtf > cufflinks.gff3
```

- ☐ Inspect the resulting file to check that it has the correct GFF3 format. The resulting GFF file describes the locations of the splice junctions in a form that **MAKER** can understand. In a moment, we will configure **MAKER** to use this file as evidence for improving gene predictions. Before we do so, we will obtain some additional data from related organisms.
- ☐ Let's explore how to download existing information on *M. oryzae* genes from the National Center for Biotechnology Information (NCBI). Visit the NCBI-GenBank site at http://www.ncbi.nlm.nih.gov/genbank/
- ☐ At the top of the page, select "Nucleotide" from the drop-down, enter "Magnaporthe [organism]" in the search box, and click search. This returns a list of gene sequences that have "Magnaporthe" as part of their organism name. Click on the name of a sequence to look at the GenBank entry for that gene, or click the "FASTA" link below the result to view the gene sequence in FASTA format. With over 15,000 results, it is infeasible to download the sequences one-by-one. Fortunately, GenBank allows downloading the full search results.
- ☐ Return to the search result page; click "Send to" near the top. A drop-down will appear; click "File" there. Then, under "Format" select "FASTA". If you now click "Create File", your download will begin. However, we do not want to overload the network in the computer lab, so we have already placed the results in your directory. Cancel your download if you have begun.

We can now configure **MAKER** to use the evidences we have gathered:

- ☐ Make sure you're in your *~/genes/maker* directory.
- ☐ Use **nano** to open up *maker_opts.ctl* for editing again.
- ☐ Change the following settings:

est_gff=*/home/yourusername/genes/maker/cufflinks.gff3*

This option provides EST evidence in the form of GFF3 annotations. We are providing the set of splice junctions detected with **cufflinks** (a program related to **stringtie** that will be covered in Lab 6); **MAKER** will use these data to refine the sometimes-incorrect predictions from the gene finder programs.

protein=*/home/yourusername/genes/maker/genbank/ncbi-protein-Magnaporthe_organism.fasta*

This option provides protein sequences from this or related organisms, in FASTA format. These sequences are mapped to the genome using **exonerate**

protein2genome, which acts like **tblastn** (protein-versus-DNA) but pays more attention to potential splice junctions in the genome sequence.

☐ Save and close the *maker_opts.ctl* file.

Now that we have added these settings, we can begin the run of **MAKER**. **MAKER** produces a lot of output to the screen, so it makes sense to redirect it to a log file. On the other hand, we might want to keep an eye on what is going on. We can pipe output to the UNIX **tee** command to send the output to both places.

☐ Let's try **tee**.

```
• ls -l ~ | tee listing.txt
```

☐ Look at the resulting file *listing.txt*.

We are now ready to run **MAKER**. However, it will take a fairly long time to complete, so it is good practice to run it using **screen**. That way, MAKER will continue processing if our network connection is dropped, or we can detach from the screen if the process has not completed before class is over.

☐ Ensure you are in **screen**.

☐ Since we have specified all of our options in the configuration file, there is no reason to provide any on the command line; we do want to log errors and not just normal output, so we use `2>&1` to send errors to the pipe. Finally, we'll append an ampersand to run the process in the background, because it can take a long time to complete (approximately 20 minutes).

```
☐ maker 2>&1 | tee maker.log
```

As **MAKER** executes, you can see the various programs that it runs: **Exonerate**, **SNAP**, and so on. This execution will take approximately 20 minutes.

Note: if **MAKER** finishes much earlier than 20 minutes, there is probably an error in your input, even if no error message appears.

☐ After completion, list the contents of the directory and take a look at the results. The output has gone to the directory *magnaporthe_oryzae_70-15_8_single_contig.maker.output*. Take a look at its contents as well.

☐ There is a *magnaporthe_oryzae_70-15_8_single_contig_datastore* directory containing copies of all the configuration files (renamed with .log file extensions), and a few additional files. One of these files contains a list of all the contigs examined, and the names of the subdirectories in which the results for that contig may be found.

☐ Change back to your *maker* directory and look at the log file. Remember to use tab completion.

```
☐ cat magnaporthe_oryzae_70-15_8_single_contig.maker.output/
  magnaporthe_oryzae_70-15_8_single_contig_master_datastore_index.log
```

Each contig has its own subdirectory buried deep within the top-level directory *magnaporthe_oryzae_70-15_8_single_contig_master_datastore*, and this directory contains results for all the various programs **MAKER** used.

☐ Merge everything together into one GFF file:

```
☐ gff3_merge -d
  magnaporthe_oryzae_70-15_8_single_contig.maker.output/
  magnaporthe_oryzae_70-15_8_single_contig_master_datastore_index.log
  -o maker-annotations.gff3
```

`-d datastore_index.log` (file path shortened for formatting)	Look for GFF files in the directories mentioned in the log file
`-o maker-annotations.gff3`	Write results to this GFF file

☐ Use **less** or **more** to look at the resulting file *maker-annotations.gff3*.

☐ As a reminder, the fields are as follows:

- seqname - name of the chromosome or scaffold; chromosome names can be given with or without the 'chr' prefix. Important note: the seqname must be one used within Ensembl, i.e. a standard chromosome name or an Ensembl identifier such as a scaffold ID, without any additional content such as species or assembly. See the example GFF output below.
- source - name of the program/project/database that generated the feature.
- feature - feature type name, e.g. Gene, Variation, Similarity.
- start - Start position of the feature, with sequence numbering starting at 1.
- end - End position of the feature, with sequence numbering starting at 1.
- score - A floating point value.
- strand - defined as + (forward) or - (reverse).
- frame - One of '0', '1' or '2'. '0' indicates that the first base of the feature is the first base of a codon, '1' that the second base is the first base of a codon, and so on..
- attribute - A semicolon-separated list of tag-value pairs, providing additional information about each feature

☐ Once you understand this format, you should be able to pick a gene and draw its structure, including start codon, exons, introns, and stop codons. Draw to approximate scale and note coordinates of each feature on the "virtual" genome.

What if we want to extract the sequences of the various features identified (transcripts, proteins)?

- ☐ We can do this using the **fasta_merge** tool (do not include a space in the file path):

```
☐ fasta_merge -d
  magnaporthe_oryzae_70-15_8_single_contig.maker.output/
  magnaporthe_oryzae_70-15_8_single_contig_master_datastore_index.log
  -o magnaporthe_oryzae_70-15_8_single_contig
```

- ☐ Use grep to count the number of genes predicted for the single contig representing Chromosome 8.7: _____

Note: Although we will not be doing so, it is possible to use the results of this **MAKER** run to re-train **SNAP** and run **MAKER** again, producing more and more accurate predictions in an iterative process.

We have only begun to scratch the surface of **MAKER**. For more information, visit the **MAKER** website at gmod.org, where many helpful links and resources are available:

http://gmod.org/wiki/MAKER

http://www.yandell-lab.org/software/maker.html