Chapter 15

Fungal Genome Annotation

Sajeet Haridas, Asaf Salamov, and Igor V. Grigoriev

Abstract

The term "genome annotation" includes identification of protein-coding and noncoding sequences (e.g., repeats, rDNA, and ncRNA) in genome assemblies and attaching functional information (metadata) to these annotated features. Here, we describe the basic outline of fungal nuclear and mitochondrial genome annotation as performed at the US Department of Energy Joint Genome Institute (JGI).

Key words Genome, Annotation, Gene prediction, Pipeline, Functional annotation

1 Introduction

Genome annotation consists of three main steps:

1. Identifying noncoding features of the genome that do not code for proteins.
2. Identifying protein coding genes, generally referred to as gene prediction.
3. Attaching biological information (functional annotation) to these genome features, for example, pfam domains, repeat classes, putative gene functions, and descriptive names.

In practice, the vast majority of annotation efforts are to identify protein-coding genes in the genome and to assign biologically meaningful names and functions to these genes.

The quality of automated annotation of genomes is highly dependent on the quality of the assembly and the availability of associated data such as RNA and protein sequences from the organism in question or a close relative. Annotation of genomes is a complex process with many input and output files and interdependent procedures. Frequently, these operations are combined into a single annotation pipeline which feeds appropriate inputs to the underlying software tools and keeps track of the output files. Many sequencing centers like the Joint Genome Institute (JGI) and the Broad Institute have developed specialized pipelines to run

Ronald P. de Vries et al. (eds.), *Fungal Genomics: Methods and Protocols*, Methods in Molecular Biology, vol. 1775,
https://doi.org/10.1007/978-1-4939-7804-5_15,

on large compute clusters [1–3]. The protocol described here is based on the JGI Genome Annotation pipeline approximated using readily available software tools (*see* **Note 1**).

2 Materials

Several pieces of software will be required for the annotation. You will need to install these programs on a Unix-like operating system using the documentation included with each one. Some of the most popular ones include (*see* **Note 1**) the following:

1. RepeatMasker (http://repeatmasker.org) to identify known repeats in the genome.
2. RepeatScout [4] and RepeatModeler (http://www.repeatmasker.org/RepeatModeler.html) to identify novel repeats in the genome.
3. BLAT [5] (https://genome.ucsc.edu/FAQ/FAQblat.html) and BLAST [6] to align transcripts and protein sequences to the genome.
4. A short read aligner like BWA [7] (http://bio-bwa.sourceforge.net) or BBmap (https://sourceforge.net/projects/bbmap/) to align transcriptome reads to the genome.
5. Ab initio gene modelers like GeneMark [8] (http://exon.gatech.edu/GeneMark/), SNAP [9] (http://korflab.ucdavis.edu/software.html), Augustus [10] (http://bioinf.uni-greifswald.de/augustus/), and Fgenesh [11] (http://www.softberry.com).
6. Evidence-based gene modelers like GeneWise [12] (http://www.ebi.ac.uk/~birney/wise2/), PASA [13] (http://pasapipeline.github.io), and Fgenesh+ (http://www.softberry.com).
7. A filtering pipeline like EVidenceModeler [14] (EVM, https://evidencemodeler.github.io) or Maker [15] (http://www.yandell-lab.org/software/maker.html) to filter and consolidate the results of the multiple genome predictors.
8. Databases like the NCBI nonredundant protein database (nr), RefSeq (https://www.ncbi.nlm.nih.gov/refseq/), UniProt/Swiss-Prot (http://www.uniprot.org) to identify homologs and impute function to the predicted genes.
9. Any number of specialized tools to add functional annotation to the predicted genes such as pfam domains [16] (http://pfam.xfam.org), signal peptides [17] (http://www.cbs.dtu.dk/services/SignalP/), and EC assignment [18] (e.g., http://priam.prabi.fr).

The annotation will require several input files including the genome assembly FASTA file (*see* **Notes 2** and **3**). Depending on the type of annotation, several additional input files will be required like mRNA or protein sequences from this or a related organism to predict protein-coding genes.

3 Methods

3.1 Identify Noncoding Genome Features

The genome encodes several kinds of noncoding features including repeats, tRNA and other ncRNA, and rDNA. Each of these can be identified using specific software tools. Of these, identification of repeats and masking them is critical to successful identification of good quality protein-coding genes (*see* **Note 4**).

The following three-step process masks repeats and transposable elements (TE) in genome sequence using RepeatModeler and RepeatMasker using 4 compute threads (-pa 4). Additional steps below identify other noncoding sequences in the genome assembly and can be skipped without affecting the downstream prediction of protein-coding genes.

1. Build a database of the genome FASTA file for RepeatModeler to run:

   ```
   $ /path/to/RepeatModeler/BuildDatabase -name myGenome -engine ncbi myGenome.fasta
   ```

2. Run RepeatModeler.

   ```
   $ RepeatModeler -engine ncbi -pa 4 -database myGenome
   ```

   Since this step will run for a long time, users may want to consider running this using nohup, screen, or a job submission system like qsub, and capturing the stdout and stderr into log files. Once the RepeatModeler run is complete, the identified repeats will be in the folder RM_*some_name* as

   ```
   consensi.fa.classified.
   $ /path/to/RepeatMasker/util/queryRepeatDatabase.pl -species fungi > fungi_repeats.lib
   ```

   Add these newly identified repeat sequences to the RepeatMasker library and create a custom library for this genome:

   ```
   $ cat fungi_repeats.lib consensi.fa.classified > myGenome.custom.repeat.lib
   ```

3. Now, run RepeatMasker on the genome using the custom library. This should also be run using nohup, screen, or some job submission system like qsub so that it can run to completion.

```
$ RepeatMasker -engine crossmatch -lib myGenome.cus-
tom.repeat.lib -pa 4 -no_is myGenome.fasta
```

The "-no_is" in the above command skips the bacterial insertion element check. You can choose not to use this option.

4. Predict tRNAs using tRNAscan-SE [19] (http://lowelab.ucsc.edu/tRNAscan-SE/):

```
$ tRNAscan-SE -o myGenome.tRNAscan.results myGen-
ome.fasta
```

The predicted tRNAs are listed in the output file set with the –o option.

5. Predict other noncoding elements. snoSeeker [20] can be used to identify snoRNA. If specialized sequencing for other ncRNA (like miRNA) was performed, reads can be aligned to the genome and tools like miRDeep [21] or miRanalyzer [22] can be used to identify these features in the genome. Alternately, users can identify known ncRNA homologs using ERPIN [23] and miRAlign [24]. Tools for identification of novel ncRNAs are still in development and remain highly experimental, often with high error rates. In the absence of accepted standards and high reliance on experimental evidence to validate these annotations, these techniques are beyond the scope of this guide. A general method is to use Infernal (http://eddylab.org/infernal/) to predict all noncoding RNAs which have corresponding covariance models in the RFAM database:

```
$ cmsearch -tblout output.file --cut_ga Rfam.cm
myGenome.fasta
```

The `-tblout` option puts the output in the file `output.file` in a tabular format which is easy to parse. Users may choose to output the full format output (default) in readable form.

3.2 Identify Protein Coding Genes

Due to intron–exon structure of eukaryotic genes, gene prediction in eukaryotes is one of the most challenging parts of the genome annotation. We recommend using several gene prediction approaches to combine different lines of evidence used for annotation: ab initio, homology-based, and transcriptome-based.

1. **Ab initio gene prediction**. All ab initio gene finding tools, e.g., GeneMarkHMM, FGENESH, Augustus, SNAP, and GlimmerHMM [25] have to be individually trained using the masked genome as described in the previous section. Here is an example of executing self-training GenMark v4.32 running on 16 compute threads:

```
$ gmes_petap.pl --ES --fungus --cores 16 --sequence
myGenome.fasta
```

2. **Homology-based gene prediction**. Homology-based gene prediction is done by mapping proteins from other organisms to the genome of interest. For example, GeneWise can use a large database like nr, uniref90, or UniProt/Swiss-Prot to generate homology-based gene models (*see* **Note 5**). The command below uses GeneWise v2.2 to search both DNA strands and produce gene models in the gff3 output format:

```
$ genewise protein.database.fasta myGenome.fasta -
both -gff
```

3. **Transcriptome-based gene prediction**. RNA-Seq data can be used in two different ways. RNA-seq reads can be mapped to the genome to predict transcripts and generate a gff3 file using the cufflinks suite as described by Trapnell et al. [26]. However, you may want to restrict the max-intron-length parameter to about 1–2 kb (from the 300 kb default) because average fungal introns are about 60 bp.

   The second approach involves mapping RNA-Seq assemblies to the genome and then building gene models from these aligned transcripts. You can see a complete walkthrough of the procedure using PASA and an explanation of the parameters at http://pasapipeline.github.io.

```
$ Launch_PASA_pipeline.pl -c alignAssembly.config -C -
R -g myGenome.fasta -t transcripts.fasta.clean -T -u
transcripts.fasta -f FL_accs.txt -USE_GMAP
```

4. **Inspect gene models**. Visually inspect the gene predictions from the different modelers by loading them into a genome viewer (like IGV, http://software.broadinstitute.org/software/igv/) to make sure that most models from the different modelers are similar to each other at the same locus for several random loci (Fig. 1). If one or more modelers produce models that are significantly different from the others, identify and correct the source of the error. For example, if the models generated by GlimmerHMM are significantly different from Augustus and SNAP (which are similar to each other), this could point to poor training of GlimmerHMM. In this case, the GlimmerHMM predictions should be either dropped from downstream processing or retrained and rerun until congruence with other modelers is achieved (also *see* **Note 6**).
5. **Select best models**. Since using multiple gene predictors creates several alternative gene models for every locus, we would like to select or construct from existing models the best model for each locus. At the JGI we use a scoring filtering procedure where every model is evaluated by transcriptome and homology support. There are several publicly available tools like EVidenceModeler to identify the best model at each locus as the first draft automated predicted gene set for this genome.

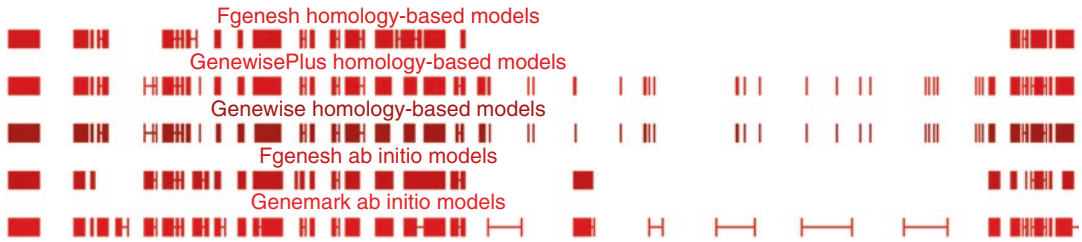

Fig. 1 Screenshot of the JGI genome browser in MycoCosm (http://jgi.doe.gov/fungi) showing the aberrant behavior of GeneMark compared to other gene predictors in this genome due to poor training. The bad short models from GeneWise are due to the badly curated protein database

Each model is given a particular weight based on the preponderance of evidence and a model is chosen for each locus based on the "winner take all" strategy. This set is further filtered after functional annotation (*see* below). In our experience, we have come to rely on a few broad criteria for weighing the fitness of a model. The weight given to particular model sources should be balanced against the probability of it being real or spurious (*see* **Notes 6–8**).

This produces the first automated approximation of the proteome. At this point, perform a sanity check on the data by comparing it to related genomes as a quality control exercise (*see* **Notes 9** and **10**).

3.3 Functional Annotation

Functional annotation of non-protein coding genome features is usually concurrent with their identification. Predicted proteins can be functionally annotated using a wide variety of tools depending on the user's requirements. The three general approaches include (1) characterization of protein sequence parts such as domains, (2) detecting similarity to already characterized protein sequences, and (3) annotation according to existing classification schemes such as EuKaryotic Orthologous Groups (KOG [27]), Gene Ontology (GO http://www.geneontology.org), Kyoto Encyclopedia of Genes and Genomes (KEGG [28] http://www.genome.jp/kegg). Many of these tools have online servers, but it may be more efficient for multiple genomes to use a local installation.

Some of the most popular domain or protein sequence feature predictors include the following:

1. hmmscan (http://hmmer.org) to identify pfam domains in predicted proteins. Genes encoding proteins harboring known TE domains should be removed from the predicted gene set because these are traditionally not included in the organism's gene set.

```
$ hmmscan -domtblout output_filename --cut_ga
Pfam-A.hmm myProteins.fasta
```

2. signalP (http://www.cbs.dtu.dk/services/SignalP/) to identify signal peptides suggesting protein secretion in predicted

genes. The presence of signal peptides can also serve as evidence for a valid gene model, which could be especially important for small single exon predictions where error (false prediction) rates are high.

```
$ signalp myProteins.fasta > output_filename
```

3. TMHMM (http://www.cbs.dtu.dk/services/TMHMM/) predicts transmembrane domains, useful to identify membrane-bound proteins:

```
$ tmhmm myProteins.fasta > output_filename
```

4. Psort (http://psort.hgc.jp/) predicts cellular localization of proteins:

```
$ runWolfPsortSummary fungi < myProteins.fasta >
output_filename
```

5. InterProScan [29] offers a collection of functional and structural protein domains. It is available from EMBL-EBI (http://www.ebi.ac.uk/interpro/download.html) and can be run using:

```
$ interproscan.sh -i myProteins.fasta -b output_
filename -f gff3
```

   You can use the option "-f tsv" if you prefer a tab-separated text file output rather than a gff3. The –b option automatically adds file extension based on the type defined by –f.

6. Protein alignments to NCBI nr, SwissProt (http://www.expasy.org/sprot/), or UniProt using blast can serve as the first approximation of protein function. Based on blast hits to specialized databases, you can perform targeted annotations such as the identification of peptidases using the MEROPS database (http://merops.sanger.ac.uk). Using these data, often a putative function can be assigned to over half of the predicted genes which can be used to provide a biologically meaningful descriptive name to the model.
7. Gene classification systems offer another way to annotate proteins and put these annotations in a comparative context. Some of these include: (1) Gene Ontology (http://www.geneontology.org) which assigns GO terms from one of three categories: Biological Process, Molecular Function, and Cellular compartment. Interpro and SwissProt hits are used to map gene ontology to predicted proteins. (2) KEGG for metabolic pathways. This assigns EC numbers (http://www.expasy.org/enzyme/) to the proteins and maps them to metabolic pathways. (3) KOG for eukaryotic clusters of orthologs, which also provides additional support for individual models.

Small models without functional annotations, especially single exon genes, may be spurious and may need to be removed unless there appear to be lineage-specific expansion of a novel gene family,

which can be identified using tools like OrthoFinder [30] and OrthoMCL [31, 32]. A quality control check by comparison to related genomes is invaluable at this stage (*see* **Notes 11–13**).

3.4 Mitochondrial Genome Annotation

The mitochondrial genome of most organisms is highly conserved. The widely accepted view is that mitochondria evolved from an alpha-proteobacterial symbiont in the ancestor of all eukaryotes. Most mitochondria still retain many bacterial-type features such as its circular topology. Some mitochondria harbor multiple circular chromosomes (e.g., cucumber) and others have linear chromosomes (like some ciliates, cnidarians, and Chlamydomonas).

The mitochondrial genome has undergone massive reduction with many genes moving to the nuclear genome or their function being replaced by nuclear-encoded orthologs. The mitochondrial gene set is usually limited to known genes in the electron transport chain, two ribosomal RNAs, and several transfer RNAs. In spite of the limited set of fungal mitochondrial genes, the gene structures can be highly variable and complex. Several genes, especially *cox1*, usually have introns that harbor TE-like endonucleases. Some exons and introns can be very short (<10 bp) making their identification difficult. Due to these and other factors, accurate automated annotation of mitochondrial genomes has remained elusive.

The small size and limited gene set make the errors glaringly apparent. Many erroneous annotations and nonstandard gene names have been published and deposited into GenBank and RefSeq (Table 1), making their classification and identification a laborious manual process. The major steps in the annotation of mitochondrial genomes are:

1. Mitochondrial genes are transcribed polycistronically and cleaved by endonucleases at tRNAs. Therefore conceptually, the first step in mitochondrial genome annotation is the prediction of tRNAs. While several software packages exist for this, in our experience, tRNAscan-SE with organellar option (-O) works well. Other packages such as ARWEN [33] and RNAweasel [34] have also been reported to perform well on mitochondrial genomes.

   ```
   $ tRNAscan-SE -O -o mito.tRNAscan.results mito.
   fasta
   ```

2. A sizeable fraction of fungal mitochondrial protein-coding genes contain introns and they are usually of self-spliced type I or (rarely) of type II, which, unlike splicesomal introns, do not have conserved sequence motifs, and therefore present a challenge for their correct prediction. We use three methods for predicting mitochondrial protein-coding genes.

Table 1
Gene sizes (aa) of mitochondrial genes in RefSeq. Each of the 14 canonical protein coding genes (except for atp9) has over 4000 models in RefSeq. In addition, there are 1–300 models of many (>50) noncanonical mitochondrial genes. We looked at several (including some annotated as DNA or RNA polymerases) and found that these were erroneous annotations and spurious models

Gene	Min	Max	Median
cox1	65	778	516
cox2	13	1296	229
cox3	185	503	261
cob	252	537	380
nad1	54	386	322
nad2	135	923	347
nad3	66	567	116
nad4	77	580	459
nad4L	40	498	98
nad5	399	924	606
nad6	103	375	173
atp6	112	427	227
atp8	26	256	55
atp9	14	75	127

The first method is similar to prokaryotic gene finder algorithms, using translation code and protein-coding potential specific for mitochondrial genomes. The second and third methods use homology-based approaches to map intron-containing genes. For the second method, we use TBLASTN, which maps proteins to the genome without consideration of splice site consensuses and then, using a custom-made Perl script, refine the boundaries, preserving the reading frame. For the third method, we built HMMs for 14 core mitochondrial genes and use the GeneWise algorithm, with mitochondrial genetic code to predict them in the genome. Multiple predictions at particular loci can be filtered using custom scripts and manual inspection. These steps rely on the availability of a well curated and continuously updated library of good quality gene models. As seen in Table 1, standard databases like GenBank and RefSeq are unreliable for this step. Some curated databases such as the one by F.Lang (http://www.bch.umontreal.ca/

People/lang/FMGP/proteins.html) are good starting points for users to create such a database.

3. The ribosomal RNA genes are perhaps the most difficult mitochondrial genes to annotate due to their high levels of length variability. In our experience, Infernal (http://eddylab.org/infernal) with covariance models "LSU_rRNA_bacteria" and "SSU_rRNA_bacteria" from RFAM database [35] works well for fungal mitochondrial genomes.

```
$ cmsearch -tblout output.file --cut_ga covariance.
cm mito.fasta
```

4. Assembly of the circular mitochondrial genome is problematic, and its linear representation may sometimes show duplicate sequences at both ends. Such sequences may be genuine repeats in the genome sequence or artifacts of the assembly process. This can cause gene duplication or gene split across the ends. The presence of a truncated, duplicated or missing gene from the canonical set is potentially a sign of this. In such cases, resplitting the FASTA file at a new gene sparse location is recommended. The newly reconstituted FASTA sequence should be then reannotated using the steps outlined above.

4 Notes

Once the annotation is complete, it should be checked for accuracy and quality. The large number of poor models in GenBank is a testament to the lack of quality control in published genomes. Some of the most common errors that we notice are the presence of organellar scaffolds in genome assemblies, TE elements in the protein set, and incomplete or chimeric gene models. In order to generate a high-quality annotation, users should consider the following.

1. Many of the software tools used here are under active development. New version of the tools may have additional features and parameters. We have provided basic command line parameters that may change in later versions. Users should read documentation and help pages of the tools being used.
2. Use the highest possible quality assembly with a large L50 (the shortest contig length of the most contiguous 50% of the genome). A poor quality fragmented assembly will produce poor quality annotations.
3. Evaluate assembly quality using a completeness tool like CEGMA [36] or BUSCO [37]. These tools can suggest the upper limit of recoverable protein-coding genes from the assembly and can prod the user to assembly improvement if the numbers are below expectation.

4. Insufficient masking is usually more disastrous than overmasking. Undermasking can generate hundreds of spurious models, while overmasking might lose a few models that lie in or span these masked nonrepeats.
5. The quality of the protein database will have an effect on the homology-based models. Since protein conservation is restricted to active sites and those residues that affect structure and folding, it is often difficult to identify full-length gene models with homology evidence. Be aware of partial (incomplete) gene models. An ad hoc approach to extend the models in the 5′ and 3′ directions in order to identify the potential start and stop codons can affect functional annotation such as the prediction of secretion signals.
6. Compare selected model structure against mapping of RNA-Seq reads and assembly to evaluate gene models and intron–exon boundaries. In general, predicted gene models should closely match the mapping of RNA-Seq reads to the genome. A significant departure from this can point to poor structural annotation or an incorrect genome assembly.
7. Some fungal genomes are highly compact with closely spaced genes, often with overlapping UTRs. In these cases, RNA-Seq mapping sometimes produces chimeric models and noisy RNA-seq data exacerbates this problem. Using strand-specific RNA-Seq and comparison with high-quality annotations of related genomes can help mitigate such problems.
8. Gene calling for small models is highly error-prone. At the JGI, we predict genes >49aa and only keep models under 200aa if they have some additional evidence like the presence of pfam domains, signal peptides, transmembrane domains or similarity to proteins from a related genome with high quality gene predictions. These cutoffs are dependent on sequencing and assembly methods of genome and transcriptome.
9. Compare gene model statistics, such as a number of predicted genes, gene length distribution, and number and size of exons and introns, with other related genomes. This can point out errors in gene prediction.
10. Generate a phylogenetic tree using single copy orthologs (we use ~200) to confirm that the genome assembly is placed where expected as compared to its nearest relatives. This can help identify misidentified DNA source material or point to errors in its previous classification.
11. Compare functional annotation with that of related genomes such as the proportion of gene models with pfam domains. Like the previous point, this can also identify errors in structural and functional annotations.

12. Compare nr BLAST hits within each scaffold. A large proportion (>50%) of hits to other phyla can identify assembly artifacts. While bacterial and human contamination of samples and reads is well known, we have seen scaffolds showing hits to a wide variety of taxonomic lineages including amphibians, reptiles, and plants. The ability to extract DNA from a pure culture is not a sufficient safeguard from this, since contamination can occur at later stages which are often outside the control of the DNA producing laboratory.
13. After initial quality assessment of the annotated genome, it is often subject to a multi-tier process that includes an assessment by peers, community annotation, and GenBank review. While automated annotation is an important source of information, manual curation is still useful in many cases, despite a lack of published standards and user training/background variability. The main purpose of manual curation is to validate structure and function of individual genes based on available lines of evidence: similarity to mapped transcripts, genome conservation, and alignment with other proteins and domains. JGI Mycocosm (http://jgi.doe.gov/fungi) offers tools for community-based manual curation for every hosted genome and enables curators to add, remove, and modify both functional and structural annotations. Some popular tools for manual annotation include GenomeView (http://genomeview.org/) and Apollo (http://gmod.org/wiki/Apollo).

Acknowledgment

The work conducted by the US Department of Energy Joint Genome Institute, a DOE Office of Science User Facility, is supported under Contract No. DE-AC02-05CH11231.

References

1. Grigoriev IV, Nikitin R, Haridas S, Kuo A, Ohm R, Otillar R, Riley R, Salamov A, Zhao X, Korzeniewski F, Smirnova T, Nordberg H, Dubchak I, Shabalov I (2014) MycoCosm portal: gearing up for 1000 fungal genomes. Nucleic Acids Res 42(Database issue):D699–D704. https://doi.org/10.1093/nar/gkt1183
2. Haas BJ, Zeng Q, Pearson MD, Cuomo CA, Wortman JR (2011) Approaches to fungal genome annotation. Mycology 2(3):118–141. https://doi.org/10.1080/21501203.2011.606851
3. Kuo A, Bushnell B, Grigoriev IV (2014) Fungal genomics: sequencing and annotation. Adv Bot Res 70:1–52. https://doi.org/10.1016/b978-0-12-397940-7.00001-x
4. Price AL, Jones NC, Pevzner PA (2005) De novo identification of repeat families in large genomes. Bioinformatics 21(Suppl 1):i351–i358. https://doi.org/10.1093/bioinformatics/bti1018
5. Kent WJ (2002) BLAT – the BLAST-like alignment tool. Genome Res 12(4):656–664. https://doi.org/10.1101/gr.229202. Article published online before March 2002
6. Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, Madden TL (2009) BLAST+: architecture and applications. BMC Bioinformatics 10:421. https://doi.org/10.1186/1471-2105-10-421
7. Li H, Durbin R (2010) Fast and accurate long-read alignment with burrows-wheeler

transform. Bioinformatics 26(5):589–595. https://doi.org/10.1093/bioinformatics/btp698

8. Ter-Hovhannisyan V, Lomsadze A, Chernoff YO, Borodovsky M (2008) Gene prediction in novel fungal genomes using an ab initio algorithm with unsupervised training. Genome Res 18(12):1979–1990. https://doi.org/10.1101/gr.081612.108
9. Korf I (2004) Gene finding in novel genomes. BMC Bioinformatics 5:59. https://doi.org/10.1186/1471-2105-5-59
10. Stanke M, Schoffmann O, Morgenstern B, Waack S (2006) Gene prediction in eukaryotes with a generalized hidden Markov model that uses hints from external sources. BMC Bioinformatics 7:62. https://doi.org/10.1186/1471-2105-7-62
11. Salamov AA, Solovyev VV (2000) Ab initio gene finding in Drosophila genomic DNA. Genome Res 10(4):516–522
12. Birney E, Clamp M, Durbin R (2004) GeneWise and Genomewise. Genome Res 14(5):988–995. https://doi.org/10.1101/gr.1865504
13. Haas BJ, Delcher AL, Mount SM, Wortman JR, Smith RK Jr, Hannick LI, Maiti R, Ronning CM, Rusch DB, Town CD, Salzberg SL, White O (2003) Improving the Arabidopsis genome annotation using maximal transcript alignment assemblies. Nucleic Acids Res 31(19):5654–5666
14. Haas BJ, Salzberg SL, Zhu W, Pertea M, Allen JE, Orvis J, White O, Buell CR, Wortman JR (2008) Automated eukaryotic gene structure annotation using EVidenceModeler and the program to assemble spliced alignments. Genome Biol 9(1):R7. https://doi.org/10.1186/gb-2008-9-1-r7
15. Holt C, Yandell M (2011) MAKER2: an annotation pipeline and genome-database management tool for second-generation genome projects. BMC Bioinformatics 12:491. https://doi.org/10.1186/1471-2105-12-491
16. Finn RD, Coggill P, Eberhardt RY, Eddy SR, Mistry J, Mitchell AL, Potter SC, Punta M, Qureshi M, Sangrador-Vegas A, Salazar GA, Tate J, Bateman A (2016) The Pfam protein families database: towards a more sustainable future. Nucleic Acids Res 44(D1):D279–D285. https://doi.org/10.1093/nar/gkv1344
17. Petersen TN, Brunak S, von Heijne G, Nielsen H (2011) SignalP 4.0: discriminating signal peptides from transmembrane regions. Nat Methods 8(10):785–786. https://doi.org/10.1038/nmeth.1701
18. Claudel-Renard C, Chevalet C, Faraut T, Kahn D (2003) Enzyme-specific profiles for genome annotation: PRIAM. Nucleic Acids Res 31(22):6633–6639
19. Lowe TM, Eddy SR (1997) tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence. Nucleic Acids Res 25(5):955–964
20. Yang JH, Zhang XC, Huang ZP, Zhou H, Huang MB, Zhang S, Chen YQ, Qu LH (2006) snoSeeker: an advanced computational package for screening of guide and orphan snoRNA genes in the human genome. Nucleic Acids Res 34(18):5112–5123. https://doi.org/10.1093/nar/gkl672
21. An J, Lai J, Lehman ML, Nelson CC (2013) miRDeep*: an integrated application tool for miRNA identification from RNA sequencing data. Nucleic Acids Res 41(2):727–737. https://doi.org/10.1093/nar/gks1187
22. Hackenberg M, Rodriguez-Ezpeleta N, Aransay AM (2011) miRanalyzer: an update on the detection and analysis of microRNAs in high-throughput sequencing experiments. Nucleic Acids Res 39(Web Server issue):W132–W138. https://doi.org/10.1093/nar/gkr247
23. Sebastian B, Aggrey SE (2008) Specificity and sensitivity of PROMIR, ERPIN and MIRABELA in predicting pre-microRNAs in the chicken genome. In Silico Biol 8(5–6):377–381
24. Wang X, Zhang J, Li F, Gu J, He T, Zhang X, Li Y (2005) MicroRNA identification based on sequence and structure alignment. Bioinformatics 21(18):3610–3614. https://doi.org/10.1093/bioinformatics/bti562
25. Majoros WH, Pertea M, Salzberg SL (2004) TigrScan and GlimmerHMM: two open source ab initio eukaryotic gene-finders. Bioinformatics 20(16):2878–2879. https://doi.org/10.1093/bioinformatics/bth315
26. Trapnell C, Roberts A, Goff L, Pertea G, Kim D, Kelley DR, Pimentel H, Salzberg SL, Rinn JL, Pachter L (2012) Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. Nat Protoc 7(3):562–578. https://doi.org/10.1038/nprot.2012.016
27. Koonin EV, Fedorova ND, Jackson JD, Jacobs AR, Krylov DM, Makarova KS, Mazumder R, Mekhedov SL, Nikolskaya AN, Rao BS, Rogozin IB, Smirnov S, Sorokin AV, Sverdlov AV, Vasudevan S, Wolf YI, Yin JJ, Natale DA (2004) A comprehensive evolutionary classification of proteins encoded in complete eukaryotic genomes. Genome Biol 5(2):R7. https://doi.org/10.1186/gb-2004-5-2-r7

28. Kanehisa M, Goto S, Hattori M, Aoki-Kinoshita KF, Itoh M, Kawashima S, Katayama T, Araki M, Hirakawa M (2006) From genomics to chemical genomics: new developments in KEGG. Nucleic Acids Res 34(Database issue):D354–D357. https://doi.org/10.1093/nar/gkj102
29. Quevillon E, Silventoinen V, Pillai S, Harte N, Mulder N, Apweiler R, Lopez R (2005) InterProScan: protein domains identifier. Nucleic Acids Res 33(Web Server issue):W116–W120. https://doi.org/10.1093/nar/gki442
30. Emms DM, Kelly S (2015) OrthoFinder: solving fundamental biases in whole genome comparisons dramatically improves orthogroup inference accuracy. Genome Biol 16:157. https://doi.org/10.1186/s13059-015-0721-2
31. Li L, Stoeckert CJ Jr, Roos DS (2003) OrthoMCL: identification of ortholog groups for eukaryotic genomes. Genome Res 13(9):2178–2189. https://doi.org/10.1101/gr.1224503
32. Fischer S, Brunk BP, Chen F, Gao X, Harb OS, Iodice JB, Shanmugam D, Roos DS, Stoeckert CJ Jr (2011) Using OrthoMCL to assign proteins to OrthoMCL-DB groups or to cluster proteomes into new ortholog groups. Curr Protoc Bioinformatics Chapter 6:Unit 6. 12 11–19. https://doi.org/10.1002/0471250953.bi0612s35
33. Laslett D, Canback B (2008) ARWEN: a program to detect tRNA genes in metazoan mitochondrial nucleotide sequences. Bioinformatics 24(2):172–175. https://doi.org/10.1093/bioinformatics/btm573
34. Gautheret D, Lambert A (2001) Direct RNA motif definition and identification from multiple sequence alignments using secondary structure profiles. J Mol Biol 313(5):1003–1011. https://doi.org/10.1006/jmbi.2001.5102
35. Nawrocki EP, Burge SW, Bateman A, Daub J, Eberhardt RY, Eddy SR, Floden EW, Gardner PP, Jones TA, Tate J, Finn RD (2015) Rfam 12.0: updates to the RNA families database. Nucl Acids Res 43(D1):D130–D137. https://doi:10.1093/nar/gku1063
36. Parra G, Bradnam K, Korf I (2007) CEGMA: a pipeline to accurately annotate core genes in eukaryotic genomes. Bioinformatics 23(9):1061–1067. https://doi.org/10.1093/bioinformatics/btm071
37. Simao FA, Waterhouse RM, Ioannidis P, Kriventseva EV, Zdobnov EM (2015) BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. Bioinformatics 31(19):3210–3212. https://doi.org/10.1093/bioinformatics/btv351