Funannotate Documentation

Release 1.8.14

Jon Palmer

Apr 11, 2023

Contents

1 Installation **1**
- 1.1 Dependencies . . . 1

2 Preparing your Assembly **5**
- 2.1 Cleaning your Assembly . . . 5
- 2.2 Sorting/Rename FASTA Headers . . . 6
- 2.3 RepeatMasking your Assembly . . . 6

3 Gene Prediction **9**
- 3.1 Explanation of steps in examples: . . . 9
- 3.2 How are repeats used/dealt with: . . . 11
- 3.3 Explanation of inputs and options: . . . 11
- 3.4 Submitting to NCBI, what should I know? . . . 12
- 3.5 Explanation of the outputs: . . . 12

4 Providing evidence to funannotate **15**

5 Adding UTRs and refining predictions **17**
- 5.1 Why is `funannotate update` so slow?? . . . 17

6 Functional annotation **19**

7 Comparative genomics **23**

8 Annotation Databases **25**

9 Tutorials **27**
- 9.1 Genome assembly and RNA-seq . . . 27
- 9.2 Genome assembly only . . . 29
- 9.3 Non-fungal genomes (higher eukaryotes) . . . 30

10 Funannotate Commands **33**
- 10.1 Funannotate wrapper script . . . 33
- 10.2 Preparing Genome for annotation . . . 34
- 10.3 Training Ab-initio Gene Predictors . . . 40
- 10.4 Gene Prediction . . . 41
- 10.5 Adding Functional Annotation . . . 44
- 10.6 Comparative Genomics . . . 46

10.7 Installation and Database Management 47

11 Utilities **49**
11.1 Generate genome assembly stats 49
11.2 Comparing/contrast annotations to a reference 50
11.3 Format Conversion 50

CHAPTER 1

Installation

1.1 Dependencies

Funannotate has a lot of dependencies. However, it also comes with a few tools to help you get everything installed. The first is that of `funannotate check`. You'll see in the output below that the `fasta` tool is missing, which is Bill Pearsons `fasta36` a dependency of the PASA pipeline. Also the `$PASAHOME`` and `$TRINITYHOME`` variables are not set, that is because on this particular machine they are not installed, i.e. funannotate will alert you at runtime if it is missing a dependency.

```
$ funannotate check --show-versions
-------------------------------------------------------
Checking dependencies for funannotate v1.4.0
-------------------------------------------------------
You are running Python v 2.7.11. Now checking python packages...
biopython: 1.70
goatools: 0.7.11
matplotlib: 2.1.1
natsort: 5.2.0
numpy: 1.12.1
pandas: 0.22.0
psutil: 5.4.3
requests: 2.18.4
scikit-learn: 0.19.0
scipy: 0.19.1
seaborn: 0.8.1
All 11 python packages installed


You are running Perl v 5.026001. Now checking perl modules...
Bio::Perl: 1.007002
Carp: 1.42
Clone: 0.39
DBD::SQLite: 1.56
DBD::mysql: 4.046
```

(continues on next page)

(continued from previous page)

```
DBI: 1.641
DB_File: 1.84
Data::Dumper: 2.167
File::Basename: 2.85
File::Which: 1.22
Getopt::Long: 2.5
Hash::Merge: 0.300
JSON: 2.97001
LWP::UserAgent: 6.33
Logger::Simple: 2.0
POSIX: 1.76
Parallel::ForkManager: 1.19
Pod::Usage: 1.69
Scalar::Util::Numeric: 0.40
Storable: 2.62
Text::Soundex: 3.05
Thread::Queue: 3.12
Tie::File: 1.02
URI::Escape: 3.31
YAML: 1.24
threads: 2.21
threads::shared: 1.58
All 27 Perl modules installed


Checking external dependencies...
RepeatMasker: RepeatMasker 4.0.7
RepeatModeler: RepeatModeler 1.0.11
Trinity: 2.5.1
augustus: 3.2.1
bamtools: bamtools 2.4.0
bedtools: bedtools v2.27.1
blat: BLAT v35
diamond: diamond 0.9.19
emapper.py: emapper-1.0.3
ete3: 3.1.1
exonerate: exonerate 2.4.0
fasta: no way to determine
gmap: 2017-06-20
gmes_petap.pl: 4.30
hisat2: 2.1.0
hmmscan: HMMER 3.1b2 (February 2015)
hmmsearch: HMMER 3.1b2 (February 2015)
java: 1.8.0_92
kallisto: 0.43.1
mafft: v7.313 (2017/Nov/15)
makeblastdb: makeblastdb 2.7.1+
minimap2: 2.10-r761
nucmer: 3.1
pslCDnaFilter: no way to determine
rmblastn: rmblastn 2.2.27+
samtools: samtools 1.8
tRNAscan-SE: 1.23 (April 2002)
tbl2asn: unknown, likely 25.3
tblastn: tblastn 2.7.1+
trimal: trimAl v1.4.rev15 build[2013-12-17]
All 30 external dependencies are installed
```

(continues on next page)

(continued from previous page)

```
Checking Environmental Variables...
$FUNANNOTATE_DB=/usr/local/share/funannotate
$PASAHOME=/Users/jon/software/PASApipeline
$TRINITYHOME=/usr/local/opt/trinity
$EVM_HOME=/Users/jon/software/evidencemodeler
$AUGUSTUS_CONFIG_PATH=/Users/jon/software/augustus/config
$GENEMARK_PATH=/Users/jon/software/gmes_petap
$BAMTOOLS_PATH=/Users/jon/software/bamtools-2.4.0/bin
All 7 environmental variables are set
-------------------------------------------------------
```

Funannotate has a lot of dependencies and therefore installation is the most difficult part of executing the pipeline. The funannotate pipeline is written in python and can be installed with pip, i.e. *pip install funannotate*. You can see a list of *Dependencies*,

Quickest start Docker:

You can use docker to run *funannotate*. Caveats are that GeneMark is not included in the docker image (see licensing below and you can complain to the developers for making it difficult to distribute/use). I've also written a bash script that can run the docker image and auto-detect/include the proper user/volume bindings. This docker image is built off of the latest code in master, so it will be ahead of the tagged releases. The image includes the required databases as well, if you want just funannotate without the databases then that is located on docker hub as well *nextgenusfs/funannotate-slim*. So this route can be achieved with:

```
# download/pull the image from docker hub
$ docker pull nextgenusfs/funannotate

# download bash wrapper script (optional)
$ wget -O funannotate-docker https://raw.githubusercontent.com/nextgenusfs/
↪funannotate/master/funannotate-docker

# might need to make this executable on your system
$ chmod +x /path/to/funannotate-docker

# assuming it is in your PATH, now you can run this script as if it were the␣
↪funannotate executable script
$ funannotate-docker test -t predict --cpus 12
```

Quickstart Bioconda install:

The pipeline can be installed with conda (via [bioconda](https://bioconda.github.io/)):

```
#add appropriate channels
conda config --add channels defaults
conda config --add channels bioconda
conda config --add channels conda-forge

#then create environment
conda create -n funannotate "python>=3.6,<3.9" funannotate
```

If *conda* is taking forever to solve the environment, I would recommend giving [mamba](https://github.com/mamba-org/mamba) a try:

```
#install mamba into base environment
conda install -n base mamba
```

(continues on next page)

(continued from previous page)

```
#then use mamba as drop in replacmeent
mamba create -n funannotate funannotate
```

If you want to use GeneMark-ES/ET you will need to install that manually following developers instructions: http://topaz.gatech.edu/GeneMark/license_download.cgi

Note that you will need to change the shebang line for all perl scripts in GeneMark to use */usr/bin/env perl*. You will then also need to add *gmes_petap.pl* to the $PATH or set the environmental variable $GENEMARK_PATH to the gmes_petap directory.

To install just the python funannotate package, you can do this with pip:

```
python -m pip install funannotate
```

To install the most updated code in master you can run:

```
python -m pip install git+https://github.com/nextgenusfs/funannotate.git
```

Please setup database and test your installation locally using the following:

```
#start up conda ENV
conda activate funannotate

#check that all modules are installed
funannotate check --show-versions

#download/setup databases to a writable/readable location
funannotate setup -d $HOME/funannotate_db

#set ENV variable for $FUNANNOTATE_DB
echo "export FUNANNOTATE_DB=$HOME/funannotate_db" > /conda/installation/path/envs/
↪funannotate/etc/conda/activate.d/funannotate.sh
echo "unset FUNANNOTATE_DB" > /conda/installation/path/envs/funannotate/etc/conda/
↪deactivate.d/funannotate.sh

#run tests -- requires internet connection to download data
funannotate test -t all --cpus X
```

CHAPTER 2

Preparing your Assembly

There are a few things that you can do to your multi-FASTA assembly to get it "ready" to be annotated. These steps include methods for removing small repetitive contigs from an assembly, sorting/renaming contig headers so they do not cause problems during prediction step, and repeatmasking your assembely (required).

2.1 Cleaning your Assembly

When working with haploid assemblies, sometimes you want to remove some repetitive contigs that are contained in other scaffolds of the assembly. If the repeats are indeed unique, then we want to keep them in the assembly. Funannotate can help "clean" up repetitive contigs in your assembly. This is done using a "leave one out" methodology using minimap2 or mummer (nucmer), where the the shortest contigs/scaffolds are aligned to the rest of the assembly to determine if it is repetitive. The script loops through the contigs starting with the shortest and workings its way to the N50 of the assembly, dropping contigs/scaffolds that are greater than the percent coverage of overlap (`--cov`) and the percent identity of overlap (`--pident`).

```
$ funannotate clean

    Usage:       funannotate clean <arguments>
    version:     1.8.14

    Description: The script sorts contigs by size, starting with shortest contigs it␣
↪uses minimap2
                 to find contigs duplicated elsewhere, and then removes duplicated␣
↪contigs.

    Arguments:
      -i, --input    Multi-fasta genome file (Required)
      -o, --out      Cleaned multi-fasta output file (Required)
      -p, --pident   Percent identity of overlap. Default = 95
      -c, --cov      Percent coverage of overlap. Default = 95
      -m, --minlen   Minimum length of contig to keep. Default = 500
      --exhaustive   Test every contig. Default is to stop at N50 value.
```

2.2 Sorting/Rename FASTA Headers

NCBI limits the number of characters in a FASTA header for submission to 16 characters and Augustus also has problems with longer contig/scaffold names. You can use this simple script to sort your assembly by length and then rename the FASTA headers.

```
$funannotate sort

    Usage:       funannotate sort <arguments>
    version:     1.8.14

    Description: This script sorts the input contigs by size (longest->shortest) and␣
↪then relabels
                 the contigs with a simple name (e.g. scaffold_1).  Augustus can have␣
↪problems with
                 some complicated contig names.

    Arguments:
      -i, --input    Multi-fasta genome file. (Required)
      -o, --out      Sorted by size and relabeled output file. (Required)
      -b, --base     Base name to relabel contigs. Default: scaffold
      --minlen       Shorter contigs are discarded. Default: 0
```

2.3 RepeatMasking your Assembly

This is an essential step in the annotation process. As of v1.4.0 repeatmasking has been decoupled from `funannotate predict` in order to make it more flexible and accomodate those users that don't have access to the RepBase library (a requirement of RepeatMasker). The `funannotate mask` command default is to run simple masking using tantan. The script is a wrapper for RepeatModeler and RepeatMasker, however you can use any external program to softmask your assembly. Softmasking is where repeats are represented by lowercase letters and all non-repetitive regions are uppercase letters. One alternative to RepeatMasker is RED (REpeat Detector) you can find a wrapper for this program Redmask.

```
$funannotate mask

    Usage:       funannotate mask <arguments>
    version:     1.8.14

    Description: This script is a wrapper for repeat masking. Default is to run very␣
↪simple
                 repeat masking with tantan. The script can also run RepeatMasker and/
↪or
                 RepeatModeler. It will generate a softmasked genome. Tantan is␣
↪probably not
                 sufficient for soft-masking an assembly, but with RepBase no longer␣
↪being
                 available RepeatMasker/Modeler may not be functional for many users.

    Arguments:
      -i, --input                 Multi-FASTA genome file. (Required)
      -o, --out                   Output softmasked FASTA file. (Required)

    Optional:
      -m, --method                Method to use. Default: tantan [repeatmasker,␣
↪repeatmodeler]
```

(continues on next page)

(continued from previous page)

```
    -s, --repeatmasker_species   Species to use for RepeatMasker
    -l, --repeatmodeler_lib      Custom repeat database (FASTA format)
    --cpus                       Number of cpus to use. Default: 2
    --debug                      Keep intermediate files
```

CHAPTER 3

Gene Prediction

Gene prediction in funannotate is dynamic in the sense that it will adjust based on the input parameters passed to the `funannotate predict` script. At the core of the prediction algorithm is Evidence Modeler, which takes several different gene prediction inputs and outputs consensus gene models. The *ab initio* gene predictors are Augustus, snap, glimmerHMM, CodingQuarry and GeneMark-ES/ET (optional due to licensing). An important component of gene prediction in funannotate is providing "evidence" to the script, you can read more about *Providing evidence to funannotate*. To explain how `funannotate predict` works, I will walk-through a few examples and describe step-by-step what is happening.

Note that as of funannotate v1.4.0, repeat masking is decoupled from `funannotate predict`, thus predict is expecting that your genome input (`-i`) is softmasked multi-FASTA file. RepeatModeler/RepeatMasker mediated masking is now done with the `funannotate mask` command. You can read more about repeatmasking

3.1 Explanation of steps in examples:

1. Genome fasta file, Trinity transcripts, RNAseq BAM file and PASA/transdecoder data.

```
funannotate predict -i genome.fasta --species "Genome awesomenous" --isolate T12345 \
    --transcript_evidence trinity.fasta --rna_bam alignments.bam --pasa_gff pasa.gff3
```

- **In this example, funannotate will run the following steps:**
    1. Align Transcript Evidence to genome using minimap2
    2. Align Protein Evidence to genome using Diamond/Exonerate.
    3. Parse BAM alignments generating hints file
    4. Parse PASA gene models and use to train/run Augustus, snap, GlimmerHMM
    5. Extract high-quality Augustus predictions (HiQ)
    6. Run Stringtie on BAM alignments, use results to run CodingQuarry
    7. Pass all data to Evidence Modeler and run

8. Filter gene models (length filtering, spanning gaps, and transposable elements)
9. Predict tRNA genes using tRNAscan-SE
10. Generate an NCBI annotation table (.tbl format)
11. Convert to GenBank format using tbl2asn
12. Parse NCBI error reports and alert user to invalid gene models

2. Genome fasta file and EST transcripts.

```
funannotate predict -i genome.fasta --species "Genome awesomenous" --isolate T12345 \
    --transcript_evidence my_ests.fa
```

- **Funannotate will now run the following steps:**
    1. Align Transcript Evidence (ESTs) to genome using minimap2
    2. Align Protein Evidence to genome using Diamond/Exonerate.
    3. Run GeneMark-ES (self-training) on genome fasta file
    4. Run a modified BUSCO2 script to identify conserved orthologs
    5. Combined GeneMark and BUSCO2 results, feed into Evidence Modeler
    6. Double-check EVM BUSCO2 consensus models are accurate –> use to train Augustus
    7. Run Augustus using training set derived from BUSCO2 orthologs
    8. Extract high-quality Augustus predictions (HiQ)
    9. Use BUSCO2 training set to train/run snap and GlimmerHMM
    10. Pass GeneMark, Augustus, HiQ, transcript align, protein align –> to EVM
    11. Filter gene models (length filtering, spanning gaps, and transposable elements)
    12. Predict tRNA genes using tRNAscan-SE
    13. Generate an NCBI annotation table (.tbl format)
    14. Convert to GenBank format using tbl2asn
    15. Parse NCBI error reports and alert user to invalid gene models

3. Re-using a parameters JSON file containing training data for ab-initio prediction software. As of v1.7.0 'funannotate species' now saves training data for all of the ab-initio predictors, this can be re-used by using the parameters.json file. Passing the '-p, –parameters' file will over-rule any existing training sets from 'funannotate species', ie:

```
funannotate predict -i genome.fasta --species "Genome awesomenous" --isolate T12345 \
    -p genome_awesomenous.parameters.json
```

- **Funannotate will now run the following steps:** 1. Align Protein Evidence to genome using Diamond/Exonerate. 3. Run GeneMark (if found) using *mod* HMM file found in *–parameters* 4. Run Augustus using training parameters found in *–parameters* 5. Run snap using training parameters found in *–parameters* 6. Run GlimmerHMM using training parameters found in *–parameters* 7. Run CodingQuarry (if found) using training parameters found in *–parameters* 8. Extract high-quality Augustus predictions (HiQ) 9. Pass gene models and protein evidence –> to EVM 10. Filter gene models (length filtering, spanning gaps, and transposable elements) 11. Predict tRNA genes using tRNAscan-SE 12. Generate an NCBI annotation table (.tbl format) 13. Convert to GenBank format using tbl2asn 14. Parse NCBI error reports and alert user to invalid gene models

4. Genome fasta file and Maker GFF output. Note this option is provided out of convenience for the user, however, it won't provide the best results.

```
funannotate predict -i genome.fasta --species "Genome awesomenous" --isolate T12345 \
    --maker_gff my_previous_maker.gff
```

- **Funannotate will now run the following steps:**
    1. Parse –pasa_gff and/or –other_gff
    2. Extract gene models from Maker gff
    3. Pass Maker, pasa, other models –> to EVM
    4. Filter gene models (length filtering, spanning gaps, and transposable elements)
    5. Predict tRNA genes using tRNAscan-SE
    6. Generate an NCBI annotation table (.tbl format)
    7. Convert to GenBank format using tbl2asn
    8. Parse NCBI error reports and alert user to invalid gene models

3.2 How are repeats used/dealt with:

Repetitive regions are parsed from the softmasked genome fasta file – these data are then turned into a BED file. The softmasked genomes are then passed to the *ab initio* predictors Augustus and GeneMark which each have their internal ways of working with the data – which according to the developers is preferential than hard masking the sequences.

- `--soft_mask` option controls how GeneMark deals with repetitive regions. By default this set to *2000* which means that GeneMark skips prediction on repeat regions shorter than 2 kb.
- `--repeats2evm` option passes the repeat GFF3 file to Evidence Modeler. This option is by default turned off this can too stringent for many fungal genomes that have high gene density. You might want to turn this option on for larger genomes or those that have a high repeat content.
- `--repeat_filter` is an option that controls how funannotate filters out repetitive gene models. Default is to use both overlap and blast filtering – overlap filtering uses the repeat BED file and drops gene models that are more than 90% contained within a repeat region while the blast filtering compares the amino acid sequences to a small database of known transposons.

3.3 Explanation of inputs and options:

What are the inputs?

The simplest way to run `funannotate predict` is to provide a softmasked genome fasta file, an output folder, and a species name (binomial), i.e. this would look like:

```
funannotate predict -i mygenome.fa -o output_folder -s "Aspergillus nidulans"
```

I already trained Augustus or training set is available.

In this case you can use the pre-trained parameters directly which will save a lot of time. To use this option you can see which species are pre-trained on your system with the `funannotate species` option. Then you can specify which species parameters to use with the `--augustus_species` option.

```
funannotate predict -i mygenome.fa -o output_folder -s "Aspergillus nidulans"
    --augustus_species anidulans
```

I already have Augustus and/or GeneMark predictions.

You can pass these predictions directly to funannotate using the `--augustus_gff` and the `--genemark_gtf` options. Note you need to run Augustus with the `--stopCodonExcludedFromCDS=False` for it to be properly parsed by funannotate.

```
funannotate predict -i mygenome.fa -o output_folder -s "Aspergillus nidulans"
    --augustus_gff augustus.gff --genemark_gtf genemark.gtf
```

How can I control the weights given to Evidence Modeler?

Evidence Modeler builds consensus gene models and in addition to providing EVM with the predictions/evidence it also requires "weights" for each set of evidence. By default the inputs are set to 1 for *ab initio* predictions and transcript/protein alignments. If high quality gene models from PASA are passed `--pasa_gff`, they default to a weight of 6. While if evidence from another GFF file is passed via `--other_gff` those models are set to 1 by default. You can control the weight of both the PASA evidence as well as the OTHER evidence by using a colon in the input. You now also control the weights for the ab-initio tools by utilizing the *-w, –weights* option i.e.

```
funannotate predict -i mygenome.fa -o output_folder -s "Aspergillus nidulans"
    --pasa_gff mypasamodels.gff3:8 --other_gff prediction.gff3:5

#multiple GFF files can be passed to --other_gff
funannotate predict -i mygenome.fa -o output_folder -s "Aspergillus nidulans"
    --pasa_gff mypasamodels.gff3:8 --other_gff prediction1.gff3:5 prediction2.gff3:1

#controlling the weights directly
funannotate predict -i mygenome.fa -o output_folder -s "Aspergillus nidulans"
    --weights augustus:2 pasa:8 snap:1
```

3.4 Submitting to NCBI, what should I know?

Funannotate will produce NCBI/GeneBank-submission ready output, however, there are a few things you should do if planning on submitting to NCBI.

1. **Get a locus_tag number for your genome.** You do this by starting a WGS genome submission and either specifying a locus tag or one will be assigned to you. The default in funannotate is to use "FUN".
2. **Pre-submission inquiry of unannotated genome.** If you are new to genome assembly/annotation submission, be aware that your assembly will have to undergo some quality checks before being accepted by NCBI. Sometimes this results in you have to update your assembly, i.e. remove contigs, split contigs where you have adapter contamination, etc. If you have already done your annotation and then have to make these changes it can be very difficult. Instead, you can start your WGS submission and request that the GenBank curators do a quality check on your assembly and fix any problems prior to generating annotation with funannotate.
3. Generated an SBT template file. https://submit.ncbi.nlm.nih.gov/genbank/template/submission/

3.5 Explanation of the outputs:

The output of `funannotate predict` is written to the output/predict_results folder, which contains:

File Name	Description
Basename.gbk	Annotated Genome in GenBank Flat File format
Basename.tbl	NCBI tbl annotation file
Basename.gff3	Genome annotation in GFF3 format
Basename.scaffolds.fa	Multi-fasta file of scaffolds
Basename.proteins.fa	Multi-fasta file of protein coding genes
Basename.transcripts.fa	Multi-fasta file of transcripts (mRNA)
Basename.discrepency.report.txt	tbl2asn summary report of annotated genome
Basename.error.summary.txt	tbl2asn error summary report
Basename.validation.txt	tbl2asn genome validation report
Basename.parameters.json	ab-initio training parameters

CHAPTER 4

Providing evidence to funannotate

Funannotate uses Evidence Modeler to combine *ab initio* gene model predictions with evidence (transcripts or proteins) aligned to the genome. Therefore, the evidence that you supply at runtime for `--transcript_evidence` and `--protein_evidence` are important. By default, funannotate will use the UniProtKb/SwissProt curated protein database for protein evidence. However, you can specify other forms of protein evidence, perhaps from a well-annotated closely related species, using the `--protein_evidence` option. Multiple files can be passed to both `--transcript_evidence` or `--protein_evidence` by separating the files by spaces, for example:

```
funannotate predict -i genome.fa -s "Awesome species" --transcript_evidence trinity.
↪fasta myESTs.fa \
    -o output --protein_evidence closely_related.fasta $FUNANNOTATE_DB/uniprot_sprot.
↪fasta
```

You'll notice in this example, I also added the UniProt/SwissProt protein models located in the funannotate database. I should also note that adding protein evidence from ab initio predictors of closely related species should be avoided, this is because those models have not been validated. What you are trying to do here is to provide the software with high-quality protein models so that information can be used to direct the *ab initio* gene prediction algorithms, so providing them with incorrect/truncated proteins isn't going to help your accuracy and in many cases it may hurt. It is often okay to just stick with the default UniProtKb/SwissProt protein evidence.

Sources of Evidence that work well:

1. De-novo RNA-seq assemblies (i.e. output of Trinity)
2. ESTs (for fungal genomes ESTs from related species can be downloaded from JGI Mycocosm)
3. Curated Protein models from closely related species

CHAPTER 5

Adding UTRs and refining predictions

If you have RNA-seq data and would like to use the PASA-mediated "annotation comparison" to add UTRs and refine gene model predictions, this can be accomplished using the `funannotate update` command. This script can also be run as a stand-alone to re-align RNA-seq data and/or update an existing GenBank genome.

If you have run `funannotate train` and then `funannotate predict`, this script will re-use those data and you can simply pass `funannotate update -i folder --cpus 12`. This will add the gene predictions to the SQL database and then walk through each gene comparing to existing PASA alignments, PASA will make some adjustments to the gene models. As recommended by PASA developers, this is run twice in `funannotate update`.

5.1 Why is `funannotate update` so slow??

The default SQL database for PASA is set to use SQLite – this is for compatibility. However, the limitation is that SQLite database in PASA is single threaded due to SQLite database lock issue. Thus even if you pass multiple cpus to the script, it will run all of the PASA steps single threaded, which can take a long time depending on PASA alignments and genome size. If you setup PASA to use MySQL, then the scripts can run PASA multi-threaded and `funannotate update` will run much faster.

```
    Usage:       funannotate update <arguments>
    version:     1.8.14

    Description: Script will run PASA mediated update of gene models. It can directly␣
↪update
    the annotation from an NCBI downloaded GenBank file using RNA-seq data or can be
    used after funannotate predict to refine UTRs and gene model predictions. Kallisto
    is used to evidence filter most likely PASA gene models. Dependencies are
    hisat2, Trinity, samtools, fasta, minimap2, PASA, kallisto, bedtools.

    Required:
      -i, --input             Funannotate folder or Genome in GenBank format (.gbk,.
↪gbff).
            or
```

(continues on next page)

(continued from previous page)

```
      -f, --fasta              Genome in FASTA format
      -g, --gff                Annotation in GFF3 format
      --species                Species name, use quotes for binomial, e.g.
↪"Aspergillus fumigatus"

    Optional:
      -o, --out                Output folder name
      -l, --left               Left/Forward FASTQ Illumina reads (R1)
      -r, --right              Right/Reverse FASTQ Illumina reads (R2)
      -s, --single             Single ended FASTQ reads
      --stranded               If RNA-seq library stranded. [RF,FR,F,R,no]
      --left_norm              Normalized left FASTQ reads (R1)
      --right_norm             Normalized right FASTQ reads (R2)
      --single_norm            Normalized single-ended FASTQ reads
      --pacbio_isoseq          PacBio long-reads
      --nanopore_cdna          Nanopore cDNA long-reads
      --nanopore_mrna          Nanopore mRNA direct long-reads
      --trinity                Pre-computed Trinity transcripts (FASTA)
      --jaccard_clip           Turn on jaccard clip for dense genomes [Recommended␣
↪for fungi]
      --no_normalize_reads     Skip read Normalization
      --no_trimmomatic         Skip Quality Trimming of reads
      --memory                 RAM to use for Jellyfish. Default: 50G
      -c, --coverage           Depth to normalize reads. Default: 50
      -m, --min_coverage       Min depth for normalizing reads. Default: 5
      --pasa_config            PASA assembly config file, i.e. from previous PASA run
      --pasa_db                Database to use. Default: sqlite [mysql,sqlite]
      --pasa_alignment_overlap PASA --stringent_alignment_overlap. Default: 30.0
--aligners               Aligners to use with PASA: Default: minimap2 blat [gmap]
--pasa_min_avg_per_id    PASA --MIN_AVG_PER_ID. Default: 95
--pasa_num_bp_splice     PASA --NUM_BP_PERFECT_SPLICE_BOUNDARY. Default: 3
      --max_intronlen          Maximum intron length. Default: 3000
      --min_protlen            Minimum protein length. Default: 50
      --alt_transcripts        Expression threshold (percent) to keep alt transcripts.
↪ Default: 0.1 [0-1]
      --p2g                    NCBI p2g file (if updating NCBI annotation)
      -t, --tbl2asn            Assembly parameters for tbl2asn. Example: "-l paired-
↪ends"
      --name                   Locus tag name (assigned by NCBI?). Default: use␣
↪existing
      --sbt                    NCBI Submission file
      --species                Species name, use quotes for binomial, e.g.
↪"Aspergillus fumigatus"
      --strain                 Strain name
      --isolate                Isolate name
      --SeqCenter              Sequencing facilty for NCBI tbl file. Default: CFMR
      --SeqAccession           Sequence accession number for NCBI tbl file. Default:␣
↪12345
      --cpus                   Number of CPUs to use. Default: 2

    ENV Vars:  If not passed, will try to load from your $PATH.
      --PASAHOME
      --TRINITYHOME
```

CHAPTER 6

Functional annotation

After your genome has gone through the gene prediction module and you have gene models that pass NCBI specs the next step is to add functional annotate to the protein-coding genes. Funannotate accomplishes this using several curated databases and is run using the `funannotate annotate` command.

Funannotate will parse the protein-coding models from the annotation and identify Pfam domains, CAZYmes, secreted proteins, proteases (MEROPS), and BUSCO groups. If you provide the script with InterProScan5 data `--iprscan`, funannotate will also generate additional annotation: InterPro terms, GO ontology, and fungal transcription factors. If Eggnog-mapper is installed locally or you pass eggnog results via `--eggnog`, then Eggnog annotations and COGs will be added to the functional annotation. The scripts will also parse UniProtKb/SwissProt searches with Eggnog-mapper searches (optional) to generate gene names and product descriptions.

InterProScan5 and Eggnog-Mapper are two functional annotation pipelines that can be parsed by funannotate, however due to the large database sizes they are not run directly. If `emapper.py` (Eggnog-mapper) is installed, then it will be run automatically during the functional annotation process. Because InterProScan5 is Linux only, it must be run outside funannotate and the results passed to the script. If you are on Mac, I've included a method to run InterProScan5 using Docker and the `funannotate predict` output will let the user know how to run this script. Alternatively, you can run the InterProScan5 search remotely using the `funannotate remote` command.

Phobius and SignalP will be run automatically if they are installed (i.e. in the PATH), however, Phobius will not run on Mac. If you are on Mac you can run Phobius with the `funannotate remote` script.

If you are annotating a fungal genome, you can run Secondary Metabolite Gene Cluster prediction using antiSMASH. This can be done on the webserver, submit your GBK file from predict (predict_results/yourGenome.gbk) or alternatively you can submit from the command line using `funannotate remote`. Of course, if you are on Linux you can install the antiSMASH program locally and run that way as well. The annotated GBK file is fed back to this script with the `--antismash` option.

Similarily to `funannotate predict`, the output from `funannotate annotate` will be populated in the output/annotate_results folder. The output files are:

File Name	Description
Basename.gbk	Annotated Genome in GenBank Flat File format
Basename.contigs.fsa	Multi-fasta file of contigs, split at gaps (use for NCBI submission)
Basename.agp	AGP file; showing linkage/location of contigs (use for NCBI submission)
Basename.tbl	NCBI tbl annotation file (use for NCBI submission)
Basename.sqn	NCBI Sequin genome file (use for NCBI submission)
Basename.scaffolds.fa	Multi-fasta file of scaffolds
Basename.proteins.fa	Multi-fasta file of protein coding genes
Basename.transcripts.fa	Multi-fasta file of transcripts (mRNA)
Basename.discrepency.report.txt	tbl2asn summary report of annotated genome
Basename.annotations.txt	TSV file of all annotations added to genome. (i.e. import into excel)
Gene2Products.must-fix.txt	TSV file of Gene Name/Product deflines that failed to pass tbl2asn checks and must be fixed
Gene2Products.need-curating.txt	TSV file of Gene Name/Product defines that need to be curated
Gene2Products.new-names-passed.txt	TSV file of Gene Name/Product deflines that passed tbl2asn but are not in Gene2Products database. Please submit a PR with these.

```
      $ funannotate annotate

Usage:       funannotate annotate <arguments>
version:     1.8.14

Description: Script functionally annotates the results from funannotate predict.  It␣
↪pulls
             annotation from PFAM, InterPro, EggNog, UniProtKB, MEROPS, CAZyme, and␣
↪GO ontology.

Required:
  -i, --input          Folder from funannotate predict
    or
  --genbank            Genome in GenBank format
  -o, --out            Output folder for results
    or
  --gff                Genome GFF3 annotation file
  --fasta              Genome in multi-fasta format
  -s, --species        Species name, use quotes for binomial, e.g. "Aspergillus␣
↪fumigatus"
  -o, --out            Output folder for results

Optional:
  --sbt                NCBI submission template file. (Recommended)
  -a, --annotations    Custom annotations (3 column tsv file)
  -m, --mito-pass-thru Mitochondrial genome/contigs. append with :mcode
  --eggnog             Eggnog-mapper annotations file (if NOT installed)
  --antismash          antiSMASH secondary metabolism results (GBK file from output)
  --iprscan            InterProScan5 XML file
  --phobius            Phobius pre-computed results (if phobius NOT installed)
  --signalp            SignalP pre-computed results (-org euk -format short)
  --isolate            Isolate name
  --strain             Strain name
  --rename             Rename GFF gene models with locus_tag from NCBI.
```

(continues on next page)

(continued from previous page)

```
  --fix                 Gene/Product names fixed (TSV: GeneID  Name    Product)
  --remove              Gene/Product names to remove (TSV: Gene        Product)
  --busco_db            BUSCO models. Default: dikarya
  -t, --tbl2asn         Additional parameters for tbl2asn. Default: "-l paired-ends"
  -d, --database        Path to funannotate database. Default: $FUNANNOTATE_DB
  --force               Force over-write of output folder
  --cpus                Number of CPUs to use. Default: 2
  --tmpdir              Volume/location to write temporary files. Default: /tmp
  --p2g                 protein2genome pre-computed results
  --header_length       Maximum length of FASTA headers. Default: 16
  --no-progress         Do not print progress to stdout for long sub jobs
```

CHAPTER 7

Comparative genomics

A typical workflow in a genomics project would be to compare your newly sequenced/assembled/annotated genome to other organisms. The impetus behind `funannotate compare` was that there was previously no way to easily compare multiple genomes. Funannotate stores all annotation in GenBank flat file format, while some people don't like this format as it is difficult to parse with standard unix tools, the main advantage is that the annotation can be stored in a standardized format and retrieved in the same way for each genome. GFF3 is the common output of many annotation tools, however, this doesn't work well for functional annotation as all of the "information" is stored in a single column. At any rate, `funannotate compare` can take either folders containing "funannotated" genomes or GBK files –> the output is stats, graphs, CSV files, phylogeny, etc all summarized in HTML format.

```
Usage:       funannotate compare <arguments>
version:     1.8.14

Description: Script does light-weight comparative genomics between funannotated␣
↪genomes.  Output
                        is graphs, phylogeny, CSV files, etc --> visualized in web-
↪browser.

Required:
  -i, --input         List of funannotate genome folders or GBK files

Optional:
  -o, --out           Output folder name. Default: funannotate_compare
  -d, --database      Path to funannotate database. Default: $FUNANNOTATE_DB
  --cpus              Number of CPUs to use. Default: 2
  --run_dnds          Calculate dN/dS ratio on all orthologs. [estimate,full]
  --go_fdr            P-value for FDR GO-enrichment. Default: 0.05
  --heatmap_stdev     Cut-off for heatmap. Default: 1.0
  --num_orthos        Number of Single-copy orthologs to use for ML. Default: 500
  --bootstrap         Number of boostrap replicates to run with RAxML. Default: 100
  --outgroup          Name of species to use for ML outgroup. Default: no outgroup
  --proteinortho      ProteinOrtho5 POFF results.
  --ml_method         Maxmimum Likelihood method: Default: raxml [raxml,iqtree]
  --ml_model          Substitution model for IQtree. Default: modelfinder
  --no-progress       Do not print progress to stdout for long sub jobs
```

CHAPTER 8

Annotation Databases

Funannotate uses several publicly available databases, they can be installed with the `funannotate setup` command. The currently installed databases and version numbers can be displayed with the `funannotate database` command.

Initial setup is simple and requires only a path to a database location, this can (should) be set using the $FUNANNOTATE_DB environmental variable. If $FUNANNOTATE_DB is set, then the script will use that location by default, otherwise you will need to specify a location to the script i.e.:

```
funannotate setup -d $HOME/funannotate_db
```

You could then update the databases if $FUNANNOTATE_DB is set like this:

```
funannotate setup -i all --update

#or force update of just one database
funannotate setup -i uniprot --force
```

This will download and format the databases, they can be displayed like so:

```
$ funannotate database

    Funannotate Databases currently installed:

      Database          Type        Version      Date         Num_Records  ␣
↪Md5checksum
      pfam              hmmer3      32.0         2018-08            17929  ␣
↪de7496fad69c1040fd74db1cb5eef0fc
      gene2product      text        1.45         2019-07-31         30103  ␣
↪657bb30cf3247fcb74ca4f51a4ab7c18
      interpro          xml         76.0         2019-09-18         37113  ␣
↪328f66a791f9866783764f24a74a5aa3
      dbCAN             hmmer3      8.0          2019-08-08           607  ␣
↪51c724c1f9ac45687f08d0faa689ed58
      busco_outgroups   outgroups   1.0          2019-10-20             7  ␣
↪6795b1d4545850a4226829c7ae8ef058
```

(continues on next page)

(continued from previous page)

```
     merops            diamond     12.0         2017-10-04          5009  ␣
↪a6dd76907896708f3ca5335f58560356
     mibig             diamond     1.4          2019-10-20          31023 ␣
↪118f2c11edde36c81bdea030a0228492
     uniprot           diamond     2019_09      2019-10-16          561176 ␣
↪9fc7871b8c4e3b755fe2086d77ed0645
     go                text        2019-10-07   2019-10-07          47375 ␣
↪3bc9ba43a98bf8fcd01db6e7e7813dd2
     repeats           diamond     1.0          2019-10-20          11950 ␣
↪4e8cafc3eea47ec7ba505bb1e3465d21

   To update a database type:
           funannotate setup -i DBNAME -d $HOME/funannotate_db --force

   To see install BUSCO outgroups type:
           funannotate database --show-outgroups

   To see BUSCO tree type:
           funannotate database --show-buscos
```

Similarly, database sources can be updated with the `funannotate setup` command, for example to update the gene2product database to its most recent version you would run:

```
$ funannotate setup -d $HOME/funannotate_db -i gene2product --update
```

CHAPTER 9

Tutorials

Funannotate can accommodate a variety of input data and depending on the data you have available you will use funannotate slightly differently, although the core modules are used in the following order:

clean –> sort –> mask –> train –> predict –> update –> annotate –> compare

The following sections will walk-through usage of funannotate for some common data types.

Genome and RNA sequencing data. Genome only. Options for non-fungal genomes.

9.1 Genome assembly and RNA-seq

This the "gold standard" in a sense and if the RNA-seq data is high quality should lead to a high quality annotation. Paired-end stranded RNA-seq data will yield the best results, although unstranded and/or single ended reads can also be used. Funannotate can also handle long-read RNA-seq data from PacBio or nanopore sequencers.

For this walkthrough, lets assume you have stranded RNA-seq data from 3 different time points, you have Nanopore direct mRNA sequences, and you have a genome assembly built from Spades. You have the following files in your folder:

```
Spades.genome.fa
liquid_R1.fq.gz
liquid_R2.fq.gz
solid_R1.fq.gz
solid_R2.fq.gz
medium_R1.fq.gz
medium_R2.fq.gz
nanopore_mRNA.fq.gz
```

1. Haploid fungal genome? This step is optional. Then run `funannotate clean`. Will also run `funannotate sort` to rename fasta headers.

```
funannotate clean -i Spades.genome.fa --minlen 1000 -o Spades.genome.cleaned.fa
```

2. Now sort your scaffolds by length and rename with a simple fasta header to avoid downstream problems.

```
funannotate sort -i Spades.genome.cleaned.fa -b scaffold -o Spades.genome.cleaned.
↪sorted.fa
```

3. Now we want to softmask the repetitive elements in the assembly.

```
funannotate mask -i Spades.genome.cleaned.fa --cpus 12 -o MyAssembly.fa
```

4. Now you have a cleaned up/renamed assembly where repeats have been softmasked, run `funannotate train` to align RNA-seq data, run Trinity, and then run PASA.

```
funannotate train -i MyAssembly.fa -o fun \
    --left liquid_R1.fq.gz solid_R1.fq.gz medium_R1.fq.gz \
    --right liquid_R2.fq.gz solid_R2.fq.gz medium_R2.fq.gz \
    --nanopore_mrna nanopore_mRNA.fq.gz \
    --stranded RF --jaccard_clip --species "Pseudogenus specicus" \
    --strain JMP12345 --cpus 12
```

You'll notice that I flipped on the `--jaccard_clip` option, since we have a fungal genome we are expected high gene density. This script will run and produce an output directory called `fun` and sub-directory called `training` where it will house the intermediate files.

5. After training is run, the script will tell you what command to run next, in this case it is `funannotate predict` with the following options:

```
funannotate predict -i MyAssembly.fa -o fun \
    --species "Pseudogenus specicus" --strain JMP12345 \
    --cpus 12
```

The script will run through the gene prediction pipeline. Note that the scripts will automatically identify and reuse data from `funannotate train`, including using the PASA gene models to train Augustus. If some gene models are unable to be fixed automatically, it will warn you at the end of the script which gene models need to be manually fixed (there might be some errors in tbl2asn I've not seen yet or cannot be fixed without manual intervention).

6. Since we have RNA-seq data, we will use the `funannotate update` command to add UTR data to the predictions and fix gene models that are in disagreement with the RNA-seq data.

```
funannotate update -i fun --cpus 12
```

Since we ran `funannotate train` those data will be automatically parsed and used to update the UTR data using PASA comparison method. The script will then choose the best gene model at each locus using the RNA-seq data and pseudoalignment with Kallisto. The outputs from this script are located in the `fun/update_results` folder. User will be alerted to any gene models that need to be fixed before moving onto functional annotation.

7. Now we have NCBI compatible gene models, we can now add functional annotation to the protein coding gene models. This is done with the `funannotate annotate` command. But first we want to run InterProScan, Eggnog-mapper, and antiSMASH.

    1. Running InterProScan5. You could install this locally and run with protein sequences. Otherwise I've built two other options, run from docker or run remotely using EBI servers.

```
#run using docker
funannotate iprscan -i fun -m docker --cpus 12

#run locally (Linux only)
funannotate iprscan -i fun -m local --iprscan_path /my/path/to/
↪interproscan.sh
```

2. Now we want to run Eggnog-mapper. You can run this on their webserver http://eggnogdb.embl.de/#/app/emapper or if you have it installed locally then `funannotate annotate` will run it for you.

3. If annotating a fungal genome and you are interested in secondary metabolism gene clusters you can run antiSMASH

```
funannotate remote -i fun -m antismash -e your-email@domain.edu
```

4. If you are on a Mac or you don't have phobius installed, you can also run this as a remote search

```
funannotate remote -i fun -m phobius -e your-email@domain.edu

#note you could run multiple searches at once
funannotate remote -i fun -m phobius antismash -e your-email@domain.edu
```

8. Finally you can run the `funannotate annotate` script incorporating the data you generated. Passing the funannotate folder will automatically incorporate the interproscan, antismash, phobius results.

```
funannotate annotate -i fun --cpus 12
```

Your results will be in the `fun/annotate_results` folder.

9.2 Genome assembly only

If you don't have any RNA-seq data that is okay as you can still generate a high quality annotation using funannotate. If you are able to get some transcript evidence from closely related species this can also be helpful, if not, funannotate is flexible and can still generate annotation.

1. First we want to softmask the repetitive elements in the assembly.

```
funannotate mask -i Spades.assembly.fa --cpus 12 -o MyAssembly.fa
```

2. Now you have an assembly where repeats have been softmasked, run `funannotate predict` to find genes.

```
funannotate predict -i MyAssembly.fa -o fun \
    --species "Pseudogenus specicus" --strain JMP12345 \
    --busco_seed_species botrytis_cinerea --cpus 12
```

The script will run through the gene prediction pipeline. It will use BUSCO2 to train Augustus and use self-training GeneMark-ES, note the `--busco_seed_species` option which corresponds to a pre-trained parameters for Augustus (`funannotate species` will display the local pre-trained options) - you want to pick a species that is close to the one you are annotating. If some gene models are unable to be fixed automatically, it will warn you at the end of the script which gene models need to be manually fixed (there might be some errors in tbl2asn I've not seen yet or cannot be fixed without manual intervention).

3. Now we have NCBI compatible gene models, we can now add functional annotation to the protein coding gene models. This is done with the `funannotate annotate` command. But first we want to run InterProScan, Eggnog-mapper, and antiSMASH.

    1. Running InterProScan5. You could install this locally and run with protein sequences. Otherwise I've built two other options, run from docker or run remotely using EBI servers.

```
#run using docker
funannotate iprscan -i fun -m docker --cpus 12
```

(continues on next page)

(continued from previous page)

```
#run locally (Linux only)
funannotate iprscan -i fun -m local --iprscan_path /my/path/to/
↪interproscan.sh

#using remote search
funannotate remote -i fun -m interproscan -e your-email@domain.edu
```

2. Now we want to run Eggnog-mapper. You can run this on their webserver http://eggnogdb.embl.de/#/app/emapper or if you have it installed locally then `funannotate annotate` will run it for you.
3. If annotating a fungal genome and you are interested in secondary metabolism gene clusters you can run antiSMASH

```
funannotate remote -i fun -m antismash -e your-email@domain.edu
```

4. If you are on a Mac or you don't have phobius installed, you can also run this as a remote search

```
funannotate remote -i fun -m phobius -e your-email@domain.edu

#note you could run multiple searches at once
funannotate remote -i fun -m phobius antismash -e your-email@domain.edu
```

4. Finally you can run the `funannotate annotate` script incorporating the data you generated. Passing the funannotate folder will automatically incorporate the interproscan, antismash, phobius results.

```
funannotate annotate -i fun --cpus 12
```

9.3 Non-fungal genomes (higher eukaryotes)

Since funannotate was originally written for fungal genomes, there are a few default values that you will want to pay attention to if you are not annotating a fungal genome.

1. Maximum intron length, this parameter is set by default to 3000 bp throughout the scripts, to adjust you can use the `--max_intronlen` flag.
2. In the `funannotate predict` menu there is a parameter for some fungal specific GeneMark options, these can be turned off by passing `--organism other` at runtime.
3. In larger genomes (i.e. > 100 MB?) you may get better results to pass the `--repeats2evm` option to `funannotate predict`, this will use the repeat GFF3 file in Evidence Modeler and will reduce the number of gene predictions. Note you could run the pipeline once without this flag to see the results and then run it again adding the option to compare results. If you see a large discrepancy between GeneMark and Augustus predictions, this seems to be associated with repeat regions (where one of the ab initio predictors gets hung up on repeats), then adding the `--repeats2evm` option will be beneficial.
4. Pay attention to the `--busco_db` option in all scripts. The default is set for `--busco_db dikarya` (default is specifically for dikaryotic fungi). Thus for other organisms `--busco_db` needs to be properly set for each script where it is an option. You can see the available busco databases with the following command:

```
$ funannotate database --show_buscos
------------------------------
BUSCO DB tree: (# of models)
------------------------------
```

(continues on next page)

(continued from previous page)

```
eukaryota (303)
        metazoa (978)
                nematoda (982)
                arthropoda (1066)
                        insecta (1658)
                        endopterygota (2442)
                        hymenoptera (4415)
                        diptera (2799)
                vertebrata (2586)
                        actinopterygii (4584)
                        tetrapoda (3950)
                        aves (4915)
                        mammalia (4104)
                euarchontoglires (6192)
                        laurasiatheria (6253)
        fungi (290)
                dikarya (1312)
                        ascomycota (1315)
                                pezizomycotina (3156)
                                        eurotiomycetes (4046)
                                        sordariomycetes (3725)
                                        saccharomycetes (1759)
                                                saccharomycetales (1711)
                        basidiomycota (1335)
                microsporidia (518)
        embryophyta (1440)
        protists (215)
                alveolata_stramenophiles (234)
```

CHAPTER 10

Funannotate Commands

A description for all funannotate commands.

10.1 Funannotate wrapper script

Funannotate is a series of Python scripts that are launched from a Python wrapper script. Each command has a help menu which you can print to the terminal by issuing the command without any arguments, i.e. `funannotate` yields the following.

```
$ funannotate

    Usage:       funannotate <command> <arguments>
    version:     1.8.14

    Description: Funannotate is a genome prediction, annotation, and comparison␣
↪pipeline.

    Commands:
      clean       Find/remove small repetitive contigs
      sort        Sort by size and rename contig headers
      mask        Repeatmask genome assembly

      train       RNA-seq mediated training of Augustus/GeneMark
      predict     Run gene prediction pipeline
      fix         Fix annotation errors (generate new GenBank file)
      update      RNA-seq/PASA mediated gene model refinement
      remote      Partial functional annotation using remote servers
      iprscan     InterProScan5 search (Docker or local)
      annotate    Assign functional annotation to gene predictions
      compare     Compare funannotated genomes

      util        Format conversion and misc utilities
      setup       Setup/Install databases
```

(continues on next page)

(continued from previous page)

```
      test        Download/Run funannotate installation tests
      check       Check Python, Perl, and External dependencies [--show-versions]
      species     list pre-trained Augustus species
      database    Manage databases
      outgroups   Manage outgroups for funannotate compare

    Written by Jon Palmer (2016-2022) nextgenusfs@gmail.com
```

10.2 Preparing Genome for annotation

10.2.1 funannotate clean

Script "cleans" an assembly by looking for duplicated contigs. The script first sorts the contigs by size, then starting with the shortest contig it runs a "leave one out" alignment using Mummer to determine if contig is duplicated elsewhere. This script is meant to be run with a haploid genome, it has not been tested as a method to haplodize a polyploid assembly.

```
Usage:       funannotate clean <arguments>
version:     1.8.14

Description: The script sorts contigs by size, starting with shortest contigs it uses␣
↪minimap2
                         to find contigs duplicated elsewhere, and then removes␣
↪duplicated contigs.

Arguments:
  -i, --input    Multi-fasta genome file (Required)
  -o, --out      Cleaned multi-fasta output file (Required)
  -p, --pident   Percent identity of overlap. Default = 95
  -c, --cov      Percent coverage of overlap. Default = 95
  -m, --minlen   Minimum length of contig to keep. Default = 500
  --exhaustive   Test every contig. Default is to stop at N50 value.
```

10.2.2 funannotate sort

Simple script to sort and rename a genome assembly. Often assemblers output contig/scaffold names that are incompatible with NCBI submission rules. Use this script to rename and/or drop scaffolds that are shorter than a minimum length.

```
Usage:       funannotate sort <arguments>
version:     1.8.14

Description: This script sorts the input contigs by size (longest->shortest) and then␣
↪relabels
                         the contigs with a simple name (e.g. scaffold_1).  Augustus␣
↪can have problems with
                         some complicated contig names.

Arguments:
  -i, --input    Multi-fasta genome file. (Required)
  -o, --out      Sorted by size and relabeled output file. (Required)
```

(continues on next page)

(continued from previous page)

```
  -b, --base      Base name to relabel contigs. Default: scaffold
  --minlen        Shorter contigs are discarded. Default: 0
```

10.2.3 funannotate species

This function will output the current trained species in Augustus.

```
$ funannotate species

      Species                                    Augustus                GeneMark  ␣
↪Snap   GlimmerHMM   CodingQuarry   Date
      E_coli_K12                                 augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      elegans                                    augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      awesome_testicus                           augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      thermoanaerobacter_tengcongensis           augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      pfalciparum                                augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      s_pneumoniae                               augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      culex                                      augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      bombus_impatiens1                          augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      cryptococcus                               augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      histoplasma                                augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      neurospora_crassa                          augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      schistosoma                                augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      schistosoma                                augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      pichia_stipitis                            augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      candida_tropicalis                         augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      histoplasma_capsulatum                     augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      honeybee1                                  augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      elephant_shark                             augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      cryptococcus_neoformans_neoformans_JEC21   augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      coprinus                                   augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      chlamy2011                                 augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
      verticillium_longisporum1                  augustus pre-trained    None      ␣
↪None   None         None           2019-10-24
```

(continues on next page)

(continued from previous page)

```
      arabidopsis                                augustus pre-trained   None
↪None    None          None           2019-10-24
      galdieria                                  augustus pre-trained   None
↪None    None          None           2019-10-24
      rice                                       augustus pre-trained   None
↪None    None          None           2019-10-24
      fly                                        augustus pre-trained   None
↪None    None          None           2019-10-24
      adorsata                                   augustus pre-trained   None
↪None    None          None           2019-10-24
      c_elegans_trsk                             augustus pre-trained   None
↪None    None          None           2019-10-24
      pseudogymnoascus_destructans_20631-21      augustus pre-trained   None
↪None    None          None           2019-10-24
      parasteatoda                               augustus pre-trained   None
↪None    None          None           2019-10-24
      saccharomyces_cerivisiae_1234              augustus pre-trained   None
↪None    None          None           2019-10-24
      template_prokaryotic                       augustus pre-trained   None
↪None    None          None           2019-10-24
      s_aureus                                   augustus pre-trained   None
↪None    None          None           2019-10-24
      testicus_genome                            augustus pre-trained   None
↪None    None          None           2019-10-24
      chaetomium_globosum                        augustus pre-trained   None
↪None    None          None           2019-10-24
      caenorhabditis                             augustus pre-trained   None
↪None    None          None           2019-10-24
      rhizopus_oryzae                            augustus pre-trained   None
↪None    None          None           2019-10-24
      rhodnius                                   augustus pre-trained   None
↪None    None          None           2019-10-24
      lodderomyces_elongisporus                  augustus pre-trained   None
↪None    None          None           2019-10-24
      tetrahymena                                augustus pre-trained   None
↪None    None          None           2019-10-24
      coyote_tobacco                             augustus pre-trained   None
↪None    None          None           2019-10-24
      chlamydomonas                              augustus pre-trained   None
↪None    None          None           2019-10-24
      b_pseudomallei                             augustus pre-trained   None
↪None    None          None           2019-10-24
      pneumocystis                               augustus pre-trained   None
↪None    None          None           2019-10-24
      eremothecium_gossypii                      augustus pre-trained   None
↪None    None          None           2019-10-24
      phanerochaete_chrysosporium                augustus pre-trained   None
↪None    None          None           2019-10-24
      fusarium                                   augustus pre-trained   None
↪None    None          None           2019-10-24
      cryptococcus_neoformans_gattii             augustus pre-trained   None
↪None    None          None           2019-10-24
      seahare                                    augustus pre-trained   None
↪None    None          None           2019-10-24
      ustilago_maydis                            augustus pre-trained   None
↪None    None          None           2019-10-24
      lamprey                                    augustus pre-trained   None
↪None    None          None           2019-10-24
```

(continues on next page)

(continued from previous page)

```
     nasonia                                    augustus pre-trained   None
↪None   None         None           2019-10-24
     tribolium2012                              augustus pre-trained   None
↪None   None         None           2019-10-24
     aspergillus_nidulans                       augustus pre-trained   None
↪None   None         None           2019-10-24
     cryptococcus_neoformans_neoformans_B       augustus pre-trained   None
↪None   None         None           2019-10-24
     verticillium_albo_atrum1                   augustus pre-trained   None
↪None   None         None           2019-10-24
     wheat                                      augustus pre-trained   None
↪None   None         None           2019-10-24
     test_genome                                augustus pre-trained   None
↪None   None         None           2019-10-24
     schizosaccharomyces_pombe                  augustus pre-trained   None
↪None   None         None           2019-10-24
     amphimedon                                 augustus pre-trained   None
↪None   None         None           2019-10-24
     saccharomyces_cerevisiae_rm11-1a_1         augustus pre-trained   None
↪None   None         None           2019-10-24
     aspergillus_fumigatus                      augustus pre-trained   None
↪None   None         None           2019-10-24
     aedes                                      augustus pre-trained   None
↪None   None         None           2019-10-24
     aspergillus_terreus                        augustus pre-trained   None
↪None   None         None           2019-10-24
     rubicus_maboogago                          augustus pre-trained   None
↪None   None         None           2019-10-24
     awe_test                                   augustus pre-trained   None
↪None   None         None           2019-10-24
     neurospora                                 augustus pre-trained   None
↪None   None         None           2019-10-24
     ancylostoma_ceylanicum                     augustus pre-trained   None
↪None   None         None           2019-10-24
     saccharomyces_cerevisiae_S288C             augustus pre-trained   None
↪None   None         None           2019-10-24
     yarrowia_lipolytica                        augustus pre-trained   None
↪None   None         None           2019-10-24
     Conidiobolus_coronatus                     augustus pre-trained   None
↪None   None         None           2019-10-24
     rubeus_macgubis                            augustus pre-trained   None
↪None   None         None           2019-10-24
     botrytis_cinerea                           augustus pre-trained   None
↪None   None         None           2019-10-24
     candida_guilliermondii                     augustus pre-trained   None
↪None   None         None           2019-10-24
     anidulans                                  augustus pre-trained   None
↪None   None         None           2019-10-24
     trichinella                                augustus pre-trained   None
↪None   None         None           2019-10-24
     candida_albicans                           augustus pre-trained   None
↪None   None         None           2019-10-24
     aspergillus_oryzae                         augustus pre-trained   None
↪None   None         None           2019-10-24
     fusarium_graminearum                       augustus pre-trained   None
↪None   None         None           2019-10-24
     chlorella                                  augustus pre-trained   None
↪None   None         None           2019-10-24
```

(continues on next page)

(continued from previous page)

```
     saccharomyces                                augustus pre-trained   None
↪None   None         None          2019-10-24
     chicken                                      augustus pre-trained   None
↪None   None         None          2019-10-24
     magnaporthe_grisea                           augustus pre-trained   None
↪None   None         None          2019-10-24
     bombus_terrestris2                           augustus pre-trained   None
↪None   None         None          2019-10-24
     laccaria_bicolor                             augustus pre-trained   None
↪None   None         None          2019-10-24
     cacao                                        augustus pre-trained   None
↪None   None         None          2019-10-24
     generic                                      augustus pre-trained   None
↪None   None         None          2019-10-24
     maize5                                       augustus pre-trained   None
↪None   None         None          2019-10-24
     debaryomyces_hansenii                        augustus pre-trained   None
↪None   None         None          2019-10-24
     heliconius_melpomene1                        augustus pre-trained   None
↪None   None         None          2019-10-24
     toxoplasma                                   augustus pre-trained   None
↪None   None         None          2019-10-24
     kluyveromyces_lactis                         augustus pre-trained   None
↪None   None         None          2019-10-24
     camponotus_floridanus                        augustus pre-trained   None
↪None   None         None          2019-10-24
     coprinus_cinereus                            augustus pre-trained   None
↪None   None         None          2019-10-24
     my_genome                                    augustus pre-trained   None
↪None   None         None          2019-10-24
     ustilago                                     augustus pre-trained   None
↪None   None         None          2019-10-24
     encephalitozoon_cuniculi_GB                  augustus pre-trained   None
↪None   None         None          2019-10-24
     human                                        augustus pre-trained   None
↪None   None         None          2019-10-24
     tomato                                       augustus pre-trained   None
↪None   None         None          2019-10-24
     brugia                                       augustus pre-trained   None
↪None   None         None          2019-10-24
     pea_aphid                                    augustus pre-trained   None
↪None   None         None          2019-10-24
     yeast                                        augustus pre-trained   None
↪None   None         None          2019-10-24
     zebrafish                                    augustus pre-trained   None
↪None   None         None          2019-10-24
     sulfolobus_solfataricus                      augustus pre-trained   None
↪None   None         None          2019-10-24
     Xipophorus_maculatus                         augustus pre-trained   None
↪None   None         None          2019-10-24
     schistosoma2                                 augustus pre-trained   None
↪None   None         None          2019-10-24
     pchrysosporium                               augustus pre-trained   None
↪None   None         None          2019-10-24
     leishmania_tarentolae                        augustus pre-trained   None
↪None   None         None          2019-10-24
     coccidioides_immitis                         augustus pre-trained   None
↪None   None         None          2019-10-24
```

(continues on next page)

(continued from previous page)

```
      ophidiomyces_ophiodiicola_cbs-122913       augustus pre-trained   None       ␣
↪None    None          None           2019-10-24
      maize                                      augustus pre-trained   None       ␣
↪None    None          None           2019-10-24


    Options for this script:
     To print a parameter file to terminal:
       funannotate species -p myparameters.json
     To print the parameters details from a species in the database:
       funannotate species -s aspergillus_fumigatus
     To add a new species to database:
       funannotate species -s new_species_name -a new_species_name.parameters.json
```

10.2.4 funannotate mask

Repetitive elements should be soft-masked from a genome assembly to help direct the ab-initio gene predictors. This can be accomplished with the often used RepeatModeler/RepeatMasker programs. A wrapper for RepeatModeler/RepeatMasker is the `funannotate mask` script. Note you can use any other software to soft-mask your genome prior to running the gene prediction script.

```
Usage:       funannotate mask <arguments>
version:     1.8.14

Description: This script is a wrapper for repeat masking. Default is to run very␣
↪simple
                        repeat masking with tantan. The script can also run␣
↪RepeatMasker and/or
                        RepeatModeler. It will generate a softmasked genome. Tantan␣
↪is probably not
                        sufficient for soft-masking an assembly, but with RepBase no␣
↪longer being
                        available RepeatMasker/Modeler may not be functional for␣
↪many users.

Arguments:
  -i, --input                   Multi-FASTA genome file. (Required)
  -o, --out                     Output softmasked FASTA file. (Required)

Optional:
  -m, --method                  Method to use. Default: tantan [repeatmasker,␣
↪repeatmodeler]
  -s, --repeatmasker_species    Species to use for RepeatMasker
  -l, --repeatmodeler_lib       Custom repeat database (FASTA format)
  --cpus                        Number of cpus to use. Default: 2
  --debug                       Keep intermediate files
```

10.3 Training Ab-initio Gene Predictors

10.3.1 funannotate train

In order to use this script you will need RNA-seq data from the genome you are annotating, if you don't have RNA-seq data then `funannotate predict` will train Augustus during runtime. This script is a wrapper for genome-guided Trinity RNA-seq assembly followed by PASA assembly. These methods will generate the input data to `funannotate predict`, i.e. coord-sorted BAM alignments, trinity transcripts, and high quality PASA GFF3 annotation. This script unfortunately has lots of dependencies that include Hisat2, Trinity, Samtools, Fasta, GMAP, Blat, MySQL, PASA, and RapMap. The $PASAHOME and $TRINITYHOME environmental variables need to be set or passed at runtime.

```
Usage:       funannotate train <arguments>
version:     1.8.14

Description: Script is a wrapper for de novo genome-guided transcriptome assembly␣
↪using
             Trinity followed by PASA. Illumina and Long-read (nanopore/pacbio) RNA-
↪seq
             is also supported. Dependencies are hisat2, Trinity, samtools, fasta,
             minimap2, PASA.

Required:
  -i, --input              Genome multi-fasta file
  -o, --out                Output folder name
  -l, --left               Left/Forward FASTQ Illumina reads (R1)
  -r, --right              Right/Reverse FASTQ Illumina reads (R2)
  -s, --single             Single ended FASTQ reads

Optional:
  --stranded               If RNA-seq library stranded. [RF,FR,F,R,no]
  --left_norm              Normalized left FASTQ reads (R1)
  --right_norm             Normalized right FASTQ reads (R2)
  --single_norm            Normalized single-ended FASTQ reads
  --pacbio_isoseq          PacBio long-reads
  --nanopore_cdna          Nanopore cDNA long-reads
  --nanopore_mrna          Nanopore mRNA direct long-reads
  --trinity                Pre-computed Trinity transcripts (FASTA)
  --jaccard_clip           Turn on jaccard clip for dense genomes [Recommended for␣
↪fungi]
  --no_normalize_reads     Skip read Normalization
  --no_trimmomatic         Skip Quality Trimming of reads
  --memory                 RAM to use for Jellyfish. Default: 50G
  -c, --coverage           Depth to normalize reads. Default: 50
  -m, --min_coverage       Min depth for normalizing reads. Default: 5
  --pasa_db                Database to use. Default: sqlite [mysql,sqlite]
  --pasa_alignment_overlap PASA --stringent_alignment_overlap. Default: 30.0
  --aligners               Aligners to use with PASA: Default: minimap2 blat [gmap]
  --pasa_min_pct_aligned   PASA --MIN_PERCENT_ALIGNED. Default: 90
  --pasa_min_avg_per_id    PASA --MIN_AVG_PER_ID. Default: 95
  --pasa_num_bp_splice     PASA --NUM_BP_PERFECT_SPLICE_BOUNDARY. Default: 3
  --max_intronlen          Maximum intron length. Default: 3000
  --species                Species name, use quotes for binomial, e.g. "Aspergillus␣
↪fumigatus"
  --strain                 Strain name
  --isolate                Isolate name
```

(continues on next page)

(continued from previous page)

```
  --cpus                    Number of CPUs to use. Default: 2
  --no-progress             Do not print progress to stdout for long sub jobs

ENV Vars:  If not passed, will try to load from your $PATH.
  --PASAHOME
  --TRINITYHOME
```

10.4 Gene Prediction

10.4.1 funannotate predict

This script is the "meat and potatoes" of funannotate. It will parse the data you provide and choose the best method to train the ab-initio gene predictors Augustus and GeneMark. After the predictors are trained, it runs Evidence Modeler to generate consensus gene models from all of the data present. Finally, the GFF3 file is converted to NCBI GenBank format.

```
Usage:       funannotate predict <arguments>
version:     1.8.14

Description: Script takes genome multi-fasta file and a variety of inputs to do a␣
↪comprehensive whole
             genome gene prediction.  Uses AUGUSTUS, GeneMark, Snap, GlimmerHMM,␣
↪BUSCO, EVidence Modeler,
             tbl2asn, tRNAScan-SE, Exonerate, minimap2.
Required:
  -i, --input               Genome multi-FASTA file (softmasked repeats)
  -o, --out                 Output folder name
  -s, --species             Species name, use quotes for binomial, e.g. "Aspergillus␣
↪fumigatus"

Optional:
  -p, --parameters          Ab intio parameters JSON file to use for gene predictors
  --isolate                 Isolate name, e.g. Af293
  --strain                  Strain name, e.g. FGSCA4
  --name                    Locus tag name (assigned by NCBI?). Default: FUN_
  --numbering               Specify where gene numbering starts. Default: 1
  --maker_gff               MAKER2 GFF file. Parse results directly to EVM.
  --pasa_gff                PASA generated gene models. filename:weight
  --other_gff               Annotation pass-through to EVM. filename:weight
  --rna_bam                 RNA-seq mapped to genome to train Augustus/GeneMark-ET
  --stringtie               StringTie GTF result
  -w, --weights             Ab-initio predictor and EVM weight. Example: augustus:2 or␣
↪pasa:10
  --augustus_species        Augustus species config. Default: uses species name
  --min_training_models     Minimum number of models to train Augustus. Default: 200
  --genemark_mode           GeneMark mode. Default: ES [ES,ET]
  --genemark_mod            GeneMark ini mod file
  --busco_seed_species      Augustus pre-trained species to start BUSCO. Default:␣
↪anidulans
  --optimize_augustus       Run 'optimze_augustus.pl' to refine training (long runtime)
  --busco_db                BUSCO models. Default: dikarya. `funannotate outgroups --
↪show_buscos`
  --organism                Fungal-specific options. Default: fungus. [fungus,other]
```

(continues on next page)

(continued from previous page)

```
  --ploidy                  Ploidy of assembly. Default: 1
  -t, --tbl2asn             Assembly parameters for tbl2asn. Default: "-l paired-ends"
  -d, --database            Path to funannotate database. Default: $FUNANNOTATE_DB

  --protein_evidence        Proteins to map to genome (prot1.fa prot2.fa uniprot.fa).␣
↪Default: uniprot.fa
  --protein_alignments      Pre-computed protein alignments in GFF3 format
  --p2g_pident              Exonerate percent identity. Default: 80
  --p2g_diamond_db          Premade diamond genome database for protein2genome mapping
  --p2g_prefilter           Pre-filter hits software selection. Default: diamond␣
↪[tblastn]
  --transcript_evidence     mRNA/ESTs to align to genome (trans1.fa ests.fa trinity.
↪fa). Default: none
  --transcript_alignments   Pre-computed transcript alignments in GFF3 format
  --augustus_gff            Pre-computed AUGUSTUS GFF3 results (must use --
↪stopCodonExcludedFromCDS=False)
  --genemark_gtf            Pre-computed GeneMark GTF results
  --trnascan                Pre-computed tRNAscanSE results

  --min_intronlen           Minimum intron length. Default: 10
  --max_intronlen           Maximum intron length. Default: 3000
  --soft_mask               Softmasked length threshold for GeneMark. Default: 2000
  --min_protlen             Minimum protein length. Default: 50
  --repeats2evm             Use repeats in EVM consensus model building
  --keep_evm                Keep existing EVM results (for rerunning pipeline)
  --evm-partition-interval  Min length between genes to make a partition: Default: 1500
  --no-evm-partitions       Do not split contigs into partitions
  --repeat_filter           Repetitive gene model filtering. Default: overlap blast␣
↪[overlap,blast,none]
  --keep_no_stops           Keep gene models without valid stops
  --SeqCenter               Sequencing facilty for NCBI tbl file. Default: CFMR
  --SeqAccession            Sequence accession number for NCBI tbl file. Default: 12345
  --force                   Annotated unmasked genome
  --cpus                    Number of CPUs to use. Default: 2
  --no-progress             Do not print progress to stdout for long sub jobs
  --tmpdir                  Volume/location to write temporary files. Default: /tmp
  --header_length           Maximum length of FASTA headers. Default: 16

ENV Vars:  If not specified at runtime, will be loaded from your $PATH
  --EVM_HOME
  --AUGUSTUS_CONFIG_PATH
  --GENEMARK_PATH
  --BAMTOOLS_PATH
```

10.4.2 funannotate fix

While funannotate predict does its best to generate gene models that will pass NCBI annotation specs, occasionally gene models fall through the cracks (i.e. they are errors that the author has not seen yet). Gene models that generate submission errors are automatically flagged by funannotate predict and alerted to the user. The user must manually fix the .tbl annotation file to fix these models. This script is a wrapper for archiving the previous genbank annotations and generating a new set with the supplied .tbl annotation file.

```
Usage:       funannotate fix <arguments>
version:     1.8.14
```

(continues on next page)

(continued from previous page)

```
Description: Script takes a GenBank genome annotation file and an NCBI tbl file to
                        generate updated annotation. Script is used to fix␣
↪problematic gene models
                        after running funannotate predict or funannotate update.

Required:
  -i, --input     Annotated genome in GenBank format.
  -t, --tbl       NCBI tbl annotation file.
  -d, --drop      Gene models to remove/drop from annotation. File with locus_tag 1␣
↪per line.

Optional:
  -o, --out       Output folder
  --tbl2asn       Parameters for tbl2asn. Default: "-l paired-ends"
```

10.4.3 funannotate update

This script updates gene models from *funannotate predict* using RNA-seq data. The method relies on RNA-seq –> Trinity –> PASA –> Kallisto. Using this script you can also update an NCBI GenBank genome using RNA-seq data, i.e. you can update gene models on a pre-existing submission and the script will maintain proper annotation naming/updating in accordance with NCBI rules.

```
Usage:       funannotate update <arguments>
version:     1.8.14

Description: Script will run PASA mediated update of gene models. It can directly␣
↪update
                        the annotation from an NCBI downloaded GenBank file using␣
↪RNA-seq data or can be
                        used after funannotate predict to refine UTRs and gene model␣
↪predictions. Kallisto
                        is used to evidence filter most likely PASA gene models.␣
↪Dependencies are
                        hisat2, Trinity, samtools, fasta, minimap2, PASA, kallisto,␣
↪bedtools.

Required:
  -i, --input             Funannotate folder or Genome in GenBank format (.gbk,.
↪gbff).
        or
  -f, --fasta             Genome in FASTA format
  -g, --gff               Annotation in GFF3 format
  --species               Species name, use quotes for binomial, e.g. "Aspergillus␣
↪fumigatus"

Optional:
  -o, --out               Output folder name
  -l, --left              Left/Forward FASTQ Illumina reads (R1)
  -r, --right             Right/Reverse FASTQ Illumina reads (R2)
  -s, --single            Single ended FASTQ reads
  --stranded              If RNA-seq library stranded. [RF,FR,F,R,no]
  --left_norm             Normalized left FASTQ reads (R1)
  --right_norm            Normalized right FASTQ reads (R2)
  --single_norm           Normalized single-ended FASTQ reads
  --pacbio_isoseq         PacBio long-reads
```

(continues on next page)

(continued from previous page)

```
  --nanopore_cdna          Nanopore cDNA long-reads
  --nanopore_mrna          Nanopore mRNA direct long-reads
  --trinity                Pre-computed Trinity transcripts (FASTA)
  --jaccard_clip           Turn on jaccard clip for dense genomes [Recommended for␣
↪fungi]
  --no_normalize_reads     Skip read Normalization
  --no_trimmomatic         Skip Quality Trimming of reads
  --memory                 RAM to use for Jellyfish. Default: 50G
  -c, --coverage           Depth to normalize reads. Default: 50
  -m, --min_coverage       Min depth for normalizing reads. Default: 5
  --pasa_config            PASA assembly config file, i.e. from previous PASA run
  --pasa_db                Database to use. Default: sqlite [mysql,sqlite]
  --pasa_alignment_overlap PASA --stringent_alignment_overlap. Default: 30.0
  --aligners               Aligners to use with PASA: Default: minimap2 blat [gmap]
  --pasa_min_pct_aligned   PASA --MIN_PERCENT_ALIGNED. Default: 90
  --pasa_min_avg_per_id    PASA --MIN_AVG_PER_ID. Default: 95
  --pasa_num_bp_splice     PASA --NUM_BP_PERFECT_SPLICE_BOUNDARY. Default: 3
  --max_intronlen          Maximum intron length. Default: 3000
  --min_protlen            Minimum protein length. Default: 50
  --alt_transcripts        Expression threshold (percent) to keep alt transcripts.␣
↪Default: 0.1 [0-1]
  --p2g                    NCBI p2g file (if updating NCBI annotation)
  -t, --tbl2asn            Assembly parameters for tbl2asn. Example: "-l paired-ends"
  --name                   Locus tag name (assigned by NCBI?). Default: use existing
  --sbt                    NCBI Submission file
  --species                Species name, use quotes for binomial, e.g. "Aspergillus␣
↪fumigatus"
  --strain                 Strain name
  --isolate                Isolate name
  --SeqCenter              Sequencing facilty for NCBI tbl file. Default: CFMR
  --SeqAccession           Sequence accession number for NCBI tbl file. Default: 12345
  --cpus                   Number of CPUs to use. Default: 2
  --no-progress            Do not print progress to stdout for long sub jobs

ENV Vars:  If not passed, will try to load from your $PATH.
  --PASAHOME
  --TRINITYHOME
```

10.5 Adding Functional Annotation

10.5.1 funannotate remote

Some programs are Linux-only and not compatible on Mac OSX, to accomodate all users there are a series of remote based searches that can be done from the command line. anitSMASH secondary metabolite gene cluster prediction, Phobius, and InterProScan5 can be done from this interface. Note that if you can install these tools locally, those searches will likely be much faster and thus preferred.

```
Usage:       funannotate remote <arguments>
version:     1.8.14

Description: Script runs remote server functional annotation for Phobius and
                        antiSMASH (fungi).  These searches are slow, if you can␣
↪setup these services
```

(continues on next page)

(continued from previous page)

```
                        locally it will be much faster to do that.  PLEASE do not␣
↪abuse services!

Required:
  -m, --methods       Which services to run, space separated [phobius,antismash,all]
  -e, --email         Email address to identify yourself to services.

  -i, --input         Funannotate input folder.
      or
  -g, --genbank       GenBank file (must be annotated).
  -o, --out           Output folder name.

  --force             Force query even if antiSMASH server looks busy
```

10.5.2 funannotate iprscan

This script is a wrapper for a local InterProScan5 run or a local Docker-based IPR run. The Docker build uses the blaxterlab/interproscan image.

```
Usage:       funannotate iprscan <arguments>
version:     1.8.14

Description: This script is a wrapper for running InterProScan5 using Docker or from a
             local installation. The script splits proteins into smaller chunks and␣
↪then
             launches several interproscan.sh "processes". It then combines the␣
↪results.

Arguments:
  -i, --input        Funannotate folder or FASTA protein file. (Required)
  -m, --method       Search method to use: [local, docker] (Required)
  -n, --num          Number of fasta files per chunk. Default: 1000
  -o, --out          Output XML InterProScan5 file

Docker arguments:
  -c, --cpus         Number of CPUs (total). Default: 12
  --cpus_per_chunk   Number of cpus per Docker instance. Default: 4

Local arguments:
  --iprscan_path     Path to interproscan.sh. Default: which(interproscan.sh)
  -c, --cpus         Number of InterProScan instances to run
                     (configure cpu/thread control in interproscan.properties file)
```

10.5.3 funannotate annotate

This script is run after *funannotate predict* or *funannotate update* and assigns functional annotation to the protein coding gene models. The best functional annotation is done when InterProScan 5 is run on your protein prior to running this script.

```
Usage:       funannotate annotate <arguments>
version:     1.8.14

Description: Script functionally annotates the results from funannotate predict.  It␣
↪pulls
```

(continues on next page)

(continued from previous page)

```
             annotation from PFAM, InterPro, EggNog, UniProtKB, MEROPS, CAZyme, and␣
↪GO ontology.

Required:
  -i, --input        Folder from funannotate predict
        or
  --genbank          Genome in GenBank format
  -o, --out          Output folder for results
        or
  --gff              Genome GFF3 annotation file
  --fasta            Genome in multi-fasta format
  -s, --species      Species name, use quotes for binomial, e.g. "Aspergillus␣
↪fumigatus"
  -o, --out          Output folder for results

Optional:
  --sbt              NCBI submission template file. (Recommended)
  -a, --annotations  Custom annotations (3 column tsv file)
  -m, --mito-pass-thru Mitochondrial genome/contigs. append with :mcode
  --eggnog           Eggnog-mapper annotations file (if NOT installed)
  --antismash        antiSMASH secondary metabolism results (GBK file from output)
  --iprscan          InterProScan5 XML file
  --phobius          Phobius pre-computed results (if phobius NOT installed)
  --signalp            SignalP pre-computed results (-org euk -format short)
  --isolate          Isolate name
  --strain           Strain name
  --rename           Rename GFF gene models with locus_tag from NCBI.
  --fix              Gene/Product names fixed (TSV: GeneID      Name    Product)
  --remove           Gene/Product names to remove (TSV: Gene    Product)
  --busco_db         BUSCO models. Default: dikarya
  -t, --tbl2asn      Additional parameters for tbl2asn. Default: "-l paired-ends"
  -d, --database     Path to funannotate database. Default: $FUNANNOTATE_DB
  --force            Force over-write of output folder
  --cpus             Number of CPUs to use. Default: 2
  --tmpdir             Volume/location to write temporary files. Default: /tmp
  --p2g                protein2genome pre-computed results
  --header_length      Maximum length of FASTA headers. Default: 16
  --no-progress        Do not print progress to stdout for long sub jobs
```

10.6 Comparative Genomics

10.6.1 funannotate compare

This script takes "funannotate" genomes (output from multiple *funannotate annotate*) and runs some comparative genomic operations. The script compares the annotation and generates graphs, CSV files, GO enrichment, dN/dS ratios, orthology, etc –> the output is visualized HTML format in a web browser.

```
Usage:       funannotate compare <arguments>
version:     1.8.14

Description: Script does light-weight comparative genomics between funannotated␣
↪genomes.  Output
                        is graphs, phylogeny, CSV files, etc --> visualized in web-
↪browser.
```

(continues on next page)

(continued from previous page)

```
Required:
  -i, --input         List of funannotate genome folders or GBK files

Optional:
  -o, --out           Output folder name. Default: funannotate_compare
  -d, --database      Path to funannotate database. Default: $FUNANNOTATE_DB
  --cpus              Number of CPUs to use. Default: 2
  --run_dnds          Calculate dN/dS ratio on all orthologs. [estimate,full]
  --go_fdr            P-value for FDR GO-enrichment. Default: 0.05
  --heatmap_stdev     Cut-off for heatmap. Default: 1.0
  --num_orthos        Number of Single-copy orthologs to use for ML. Default: 500
  --bootstrap         Number of boostrap replicates to run with RAxML. Default: 100
  --outgroup          Name of species to use for ML outgroup. Default: no outgroup
  --proteinortho      ProteinOrtho5 POFF results.
  --ml_method         Maxmimum Liklihood method: Default: raxml [raxml,iqtree]
  --ml_model          Substitution model for IQtree. Default: modelfinder
  --no-progress       Do not print progress to stdout for long sub jobs
```

10.7 Installation and Database Management

10.7.1 funannotate setup

This command needs to be run to download required databases. It requires the user to specify a location to save the database files. This location can then be added to the ~/.bash_profile so funannotate knows where to locate the database files.

```
Usage:       funannotate setup <arguments>
version:     1.8.14

Description: Script will download/format necessary databases for funannotate.

Options:
  -i, --install    Download format databases. Default: all
                                        [merops,uniprot,dbCAN,pfam,repeats,go,
                                         mibig,interpro,busco_outgroups,gene2product]
  -b, --busco_db   Busco Databases to install. Default: dikarya [all,fungi,aves,etc]
  -d, --database   Path to funannotate database
  -u, --update     Check remote md5 and update if newer version found
  -f, --force      Force overwriting database
  -w, --wget       Use wget to download instead of python requests
  -l, --local      Use local resource JSON file instead of current on github
```

10.7.2 funannotate database

Simple script displays the currently installed databases.

```
$ funannotate database

    Funannotate Databases currently installed:

      Database          Type        Version      Date         Num_Records
↪Md5checksum
```

(continues on next page)

(continued from previous page)

```
      pfam              hmmer3      35.0         2021-11           19632  ␣
↪c78ab387de299860bd242d6f57930c7f
      gene2product      text        1.82         2022-09-25        34212  ␣
↪23a8436fb3a7d09c87febc7f2ee86615
      interpro          xml         90.0         2022-08-04        40597  ␣
↪0cd0aff2b5df0d5c57a888e5953a754e
      dbCAN             hmmer3      10.0         2021-10-03          641  ␣
↪04696dfba1c3bb82ff9b72cfbb3e4a65
      busco_outgroups   outgroups   1.0          2022-09-25            8  ␣
↪6795b1d4545850a4226829c7ae8ef058
      merops            diamond     12.0         2017-10-04         5009  ␣
↪a6dd76907896708f3ca5335f58560356
      mibig             diamond     1.4          2019-10-20        31023  ␣
↪118f2c11edde36c81bdea030a0228492
      uniprot           diamond     2022_03      2022-08-03       568002  ␣
↪30ad53c6d2b4bc36b75ed2814a3708f7
      go                text        2022-09-19   2022-09-19        47343  ␣
↪8f0f6557c8140bc68af67ac57239236d
      repeats           diamond     1.0          2019-10-20        11950  ␣
↪4e8cafc3eea47ec7ba505bb1e3465d21

    To update a database type:
            funannotate setup -i DBNAME -d /usr/local/share/funannotate --force

    To see install BUSCO outgroups type:
            funannotate database --show-outgroups

    To see BUSCO tree type:
            funannotate database --show-buscos
```

10.7.3 funannotate outgroups

This script is a helper function to manage and update outgroups for *funannotate compare*. Outgroup species can be specified in *funannotate compare* to use as a reference for BUSCO-mediated maximum likelihood phylogeny. This script allows the user to add a genome to the available outgroups folder by running BUSCO and formatting it appropriately.

```
Usage:       funannotate outgroups <arguments>
version:     1.8.14

Description: Managing the outgroups folder for funannotate compare

Arguments:
  -i, --input            Proteome multi-fasta file. Required.
  -s, --species          Species name for adding a species. Required.
  -b, --busco_db         BUSCO db to use. Default. dikarya
  -c, --cpus             Number of CPUs to use for BUSCO search.
  -d, --database         Path to funannotate database. Default: $FUNANNOTATE_DB
```

CHAPTER 11

Utilities

There are several scripts that maybe useful to users to convert between different formats, these scripts are housed in the `funannotate util` submenu.

```
$ funannotate util

    Usage:       funannotate util <arguments>
    version:     1.8.14

    Commands:
      stats              Generate assembly and annotation stats
      contrast           Compare annotations to reference (GFF3 or GBK annotations)
      tbl2gbk            Convert TBL format to GenBank format
      gbk2parts          Convert GBK file to individual components
      gff2prot           Convert GFF3 + FASTA files to protein FASTA
      gff2tbl            Convert GFF3 format to NCBI annotation table (tbl)
      bam2gff3           Convert BAM coord-sorted transcript alignments to GFF3
      prot2genome        Map proteins to genome generating GFF3 protein alignments
      stringtie2gff3     Convert GTF (stringTIE) to GFF3 format
      quarry2gff3        Convert CodingQuarry output to proper GFF3 format
gff-rename         Sort GFF3 file and rename gene models
```

11.1 Generate genome assembly stats

To generate genome assembly stats in a JSON file.

```
$ funannotate util stats

    Usage:       funannotate util stats <arguments>
    version:     1.8.14

    Description: Generate JSON file with genome assembly and annotation stats.

```

(continues on next page)

(continued from previous page)

```
    Arguments:
      -f, --fasta              Genome FASTA file (Required)
      -o, --out                Output file (JSON format)
      -g, --gff3               Genome Annotation (GFF3 format)
      -t, --tbl                Genome Annotation (NCBI TBL format)
      --transcript_alignments  Transcript alignments (GFF3 format)
      --protein_alignments     Protein alignments (GFF3 format)
```

11.2 Comparing/contrast annotations to a reference

To compare/contrast genome annotations between different GFF3 or GBK files.

```
$ funannotate util contrast

    Usage:       funannotate util contrast <arguments>
    version:     1.8.14

    Description: Compare/constrast annotations to reference. Annotations in either␣
↪GBK or GFF3 format.

    Arguments: -r, --reference            Reference Annotation. GFF3 or GBK format
               -f, --fasta                Genome FASTA. Required if GFF3 used
               -q, --query                Annotation query. GFF3 or GBK format
               -o, --output               Output basename
               -c, --calculate_pident     Measure protein percent identity between␣
↪query and reference
```

11.3 Format Conversion

```
$ funannotate util tbl2gbk

    Usage:       funannotate util tbl2gbk <arguments>
    version:     1.8.14

    Description: Convert NCBI TBL annotations + Genome FASTA to GenBank format.

    Required:    -i, --tbl          Annotation in NCBI tbl format
                             -f, --fasta        Genome FASTA file.
                             -s, --species      Species name, use quotes for binomial,
↪ e.g. "Aspergillus fumigatus"
    Optional:
                             --isolate          Isolate name
                             --strain           Strain name
                             --sbt              NCBI Submission Template file
                             -t, --tbl2asn      Assembly parameters for tbl2asn.␣
↪Example: "-l paired-ends"
                             -o, --output       Output basename
```

```
$ funannotate util gbk2parts

    Usage:       funannotate util gbk2parts <arguments>
```

(continues on next page)

(continued from previous page)

```
    version:     1.8.14

    Description: Convert GenBank file to its individual components (parts) tbl,␣
↪protein
                              FASTA, transcript FASTA, and contig/scaffold FASTA.

    Arguments:   -g, --gbk          Input Genome in GenBank format
                                   -o, --output       Output basename
```

```
$ funannotate util gff2prot

    Usage:       funannotate util gff2prot <arguments>
    version:     1.8.14

    Description: Convert GFF3 file and genome FASTA to protein sequences. FASTA␣
↪output to stdout.

    Arguments: -g, --gff3           Reference Annotation. GFF3 format
               -f, --fasta          Genome FASTA file.
               --no_stop            Dont print stop codons
```

```
$ funannotate util gff2tbl

    Usage:       funannotate util gff2tbl <arguments>
    version:     1.8.14

    Description: Convert GFF3 file into NCBI tbl format. Tbl output to stdout.

    Arguments:
      -g, --gff3           Reference Annotation. GFF3 format
      -f, --fasta          Genome FASTA file.
```

```
$ funannotate util bam2gff3

    Usage:       funannotate util bam2gff3 <arguments>
    version:     1.8.14

    Description: Convert BAM coordsorted transcript alignments to GFF3 format.

    Arguments: -i, --bam           BAM file (coord-sorted)
               -o, --output        GFF3 output file
```

```
$ funannotate util protein2genome

    Usage:       funannotate util prot2genome <arguments>
    version:     1.8.14

    Description: Map proteins to genome using exonerate. Output is EVM compatible␣
↪GFF3 file.

    Arguments:   -g, --genome       Genome FASTA format (Required)
                 -p, --proteins     Proteins FASTA format (Required)
                 -o, --out          GFF3 output file (Required)
                 -f, --filter       Pre-filtering method. Default: diamond [diamond,
↪tblastn]
```

(continues on next page)

(continued from previous page)

```
                -t, --tblastn_out  Output to save tblastn results. Default: off
                 --tblastn          Use existing tblastn results
                --ploidy           Ploidy of assembly. Default: 1
                --maxintron        Max intron length. Default: 3000
                --cpus             Number of cpus to use. Default: 2
                --EVM_HOME         Location of Evidence Modeler home directory.␣
↪Default: $EVM_HOME
                --tmpdir           Volume/location to write temporary files.␣
↪Default: /tmp
                --logfile          Logfile output file
```

```
$ funannotate util stringtie2gff3

    Usage:       funannotate util stringtie2gff3 <arguments>
    version:     1.8.14

    Description: Convert StringTIE GTF format to GFF3 funannotate compatible format.␣
↪Output
                             to stdout.

    Arguments:   -i, --input        GTF file from stringTIE
```

```
  $ funannotate util quarry2gff3

      Usage:       funannotate util quarry2gff3 <arguments>
      version:     1.8.14

      Description: Convert CodingQuarry output GFF to proper GFF3 format. Output to␣
↪stdout.

      Arguments:   -i, --input        CodingQuarry output GFF file. (PredictedPass.
↪gff3)

.. code-block:: none

  $ funannotate util gff-rename

      Usage:       funannotate util gff-rename <arguments>
      version:     1.8.14

      Description: Sort GFF3 file by contigs and rename gene models.

      Arguments:   -g, --gff3           Reference Annotation. GFF3 format
                   -f, --fasta          Genome FASTA file.
                   -o, --out            Output GFF3 file
                   -l, --locus_tag      Locus tag to use. Default: FUN
                   -n, --numbering      Start number for genes. Default: 1
```

Funannotate is a genome prediction, annotation, and comparison software package. It was originally written to annotate fungal genomes (small eukaryotes ~ 30 Mb genomes), but has evolved over time to accomodate larger genomes. The impetus for this software package was to be able to accurately and easily annotate a genome for submission to NCBI GenBank. Existing tools (such as Maker) require significant manually editing to comply with GenBank submission rules, thus funannotate is aimed at simplifying the genome submission process.

Funannotate is also a lightweight comparative genomics platform. Genomes that have had functional annotation added via the `funannotate annotate` command can be run through the `funannotate compare` script that

outputs html based whole genome comparisons. The software can run orthologous clustering, construct whole-genome phylogenies, run Gene Ontology enrichment analysis, as well as calculate dN/dS ratios for orthologous clusters under positive selection.

- *Installation*
- *Preparing your Assembly*
- *Gene Prediction*
- *Adding UTRs and refining predictions*
- *Functional annotation*
- *Comparative genomics*
- *Tutorials*
- *Utilities*