BRAKER User Guide

Katharina J. Hoff
University of Greifswald
Germany

katharina.hoff@uni-greifswald.de
+49 3834 420 4624
https://github.com/Gaius-Augustus/BRAKER

Contents

Contributors

Authors of BRAKER are:

Katharina J. Hoff[a, b], Simone Lange[a], Alexandre Lomsadze[c], Tomas Bruna[c],
Mark Borodovsky[c, d, e], Mario Stanke[a, b]

[a] University of Greifswald, Institute for Mathematics and Computer Science, Walther-Rathenau-Str. 47, 17489 Greifswald, Germany

[b] University of Greifswald, Center for Functional Genomics of Microbes, Felix-Hausdorff-Str. 8, 17489 Greifswald, Germany

[c] Joint Georgia Tech and Emory University Wallace H Coulter Department of Biomedical Engineering, 30332 Atlanta, USA

[d] School of Computational Science and Engineering, 30332 Atlanta, USA

[e] Moscow Institute of Physics and Technology, Moscow Region 141701, Dolgoprudny, Russia

What is BRAKER?

The rapidly growing number of sequenced genomes requires fully automated methods for accurate gene structure annotation. With this goal in mind, we have developed BRAKER1[R1], a combination of GeneMark-ET [R2] and AUGUSTUS [R3, R4], that uses genomic and RNA-Seq data to automatically generate full gene structure annotations in novel genome.

However, the quality of RNA-Seq data that is available for annotating a novel genome is variable, and in some cases, RNA-Seq data is not
available, at all.

BRAKER2 is an extension of BRAKER1 which allows for **fully automated training** of the gene prediction tools GeneMark-EX [R14, R15F1] and AUGUSTUS from RNA-Seq and/or protein homology information, and that integrates the extrinsic evidence from RNA-Seq and protein homology information into the **prediction**.

In contrast to other available methods that rely on protein homology information, BRAKER2 reaches high gene prediction accuracy even in the absence of the annotation of very closely related species and in the absence of RNA-Seq data.

In this user guide, we will refer to BRAKER1 and BRAKER2 simply as **BRAKER** because they are executed by the same script (`braker.pl`).

Keys to successful gene prediction

- Use a high quality genome assembly. If you have a huge number of very short scaffolds in your genome assembly, those short scaffolds will likely increase runtime dramatically but will not increase prediction accuracy.

- Use simple scaffold names in the genome file (e.g. `>contig1` will work better than
  `>contig1my custom species namesome putative function /more/information/  and lots of special characters %&!*(){}`). Make the scaffold names in all your fasta files simple before running any alignment program.

- In order to predict genes accurately in a novel genome, the genome should be masked for repeats. This will avoid the prediction of false positive gene structures in repetitive and low complexitiy regions. Repeat masking is also essential for mapping RNA-Seq data to a genome with some tools (other RNA-Seq mappers, such as HISAT2, ignore masking information). In case of GeneMark-EX and AUGUSTUS, softmasking (i.e. putting repeat regions into lower case letters and all other regions into upper case letters) leads to better results than hardmasking (i.e. replacing letters in repetitive regions by the letter `N` for unknown nucleotide). If the genome is masked, use the `--softmasking` flag of `braker.pl` .

- Many genomes have gene structures that will be predicted accurately with standard parameters of GeneMark-EX and AUGUSTUS within BRAKER. However, some genomes have clade-specific features, i.e. special branch point model in fungi, or non-standard splice-site patterns. Please read the options section [options] in order to determine whether any of the custom options may improve gene prediction accuracy in the genome of your target species.

- Always check gene prediction results before further usage! You can e.g. use a genome browser for visual inspection of gene models in context with extrinsic evidence data. BRAKER supports the generation of track data hubs for the UCSC Genome Browser with MakeHub for this purpose.

Overview of modes for running BRAKER

BRAKER mainly features semi-unsupervised, extrinsic evidence data (RNA-Seq and/or protein spliced alignment information) supported training of GeneMark-EX[F1] and subsequent training of AUGUSTUS with integration of extrinsic evidence in the final gene prediction step. However, there are now a number of additional pipelines included in BRAKER. In the following, we give an overview of possible input files and pipelines:

- Genome file, only. In this mode, GeneMark-ES is trained on the genome sequence, alone. Long genes predicted by GeneMark-ES are selected for training AUGUSTUS. Final predictions by AUGUSTUS are *ab initio*. This approach will likely yield lower prediction accuracy than all other here described pipelines. (see Figure 1),
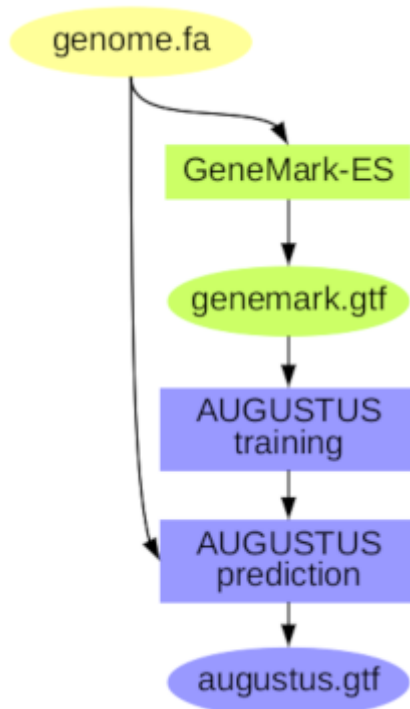
Figure 1: BRAKER pipeline A: training GeneMark-ES on genome data, only; *ab initio* gene prediction withAUGUSTUS

- Genome and RNA-Seq file from the same species (see figure 2); this approach is suitable for RNA-Seq libraries with a good coverage of the transcriptome, **important:** this approach requires that each intron is covered by many alignments, i.e. it does not work with assembled transcriptome mappings.

Figure 2: BRAKER pipeline B: training GeneMark-ET supported by RNA-Seq spliced alignment information, prediction with AUGUSTUS with that same spliced alignment information.

- Genome file and database of proteins that may be of **unknown** evolutionary distance to the target species (see Figure 3); this approach is particularly suitable if no RNA-Seq data is available, and if no protein data from a very closely related species is available. **Important:** This approach requires a database of protein families, i.e. many representatives of each protein family must be present in the database. The GaTech protein mapping database is available for download at http://exon.gatech.edu/GeneMark/Braker/protein_mapping_pipeline.tar.gz, corresponding authors for this pipeline are Alexandre Lomsadze and Tomas Bruna (Georgia Tech).
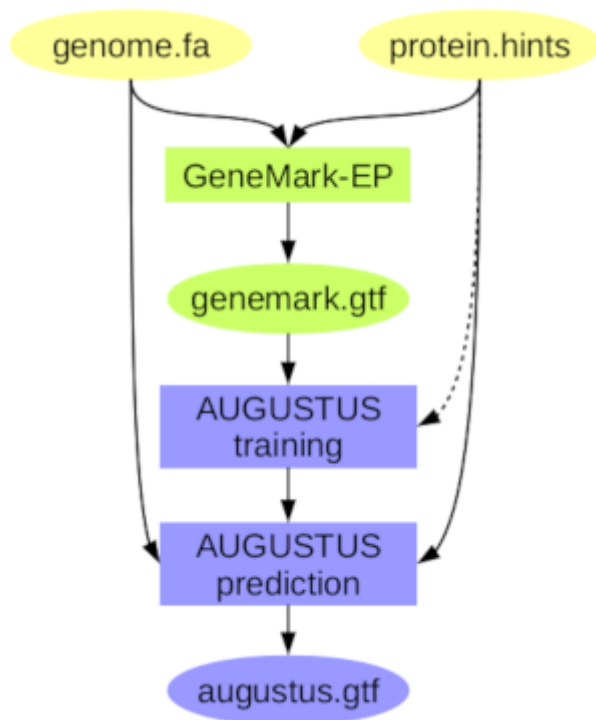
Figure 3: BRAKER pipeline C: training GeneMark-EP on protein spliced alignment information, prediction with AUGUSTUS with that same spliced alignment information. Proteins used here can be of longer evolutionary distance.

- Genome and RNA-Seq file from the same species, and proteins that may be of longer evolutionary distance to the target species (s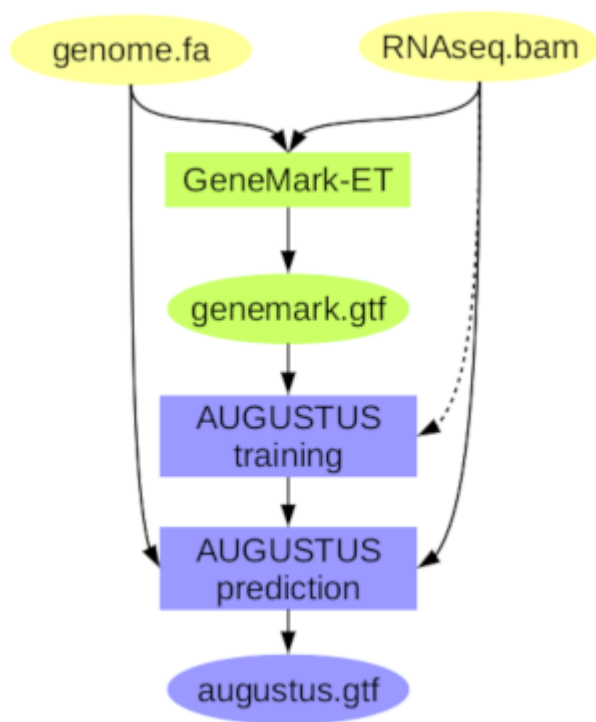ee figure 4); **important:** this approach requires a database of protein families, i.e. many representatives of each protein family must be present in the database.
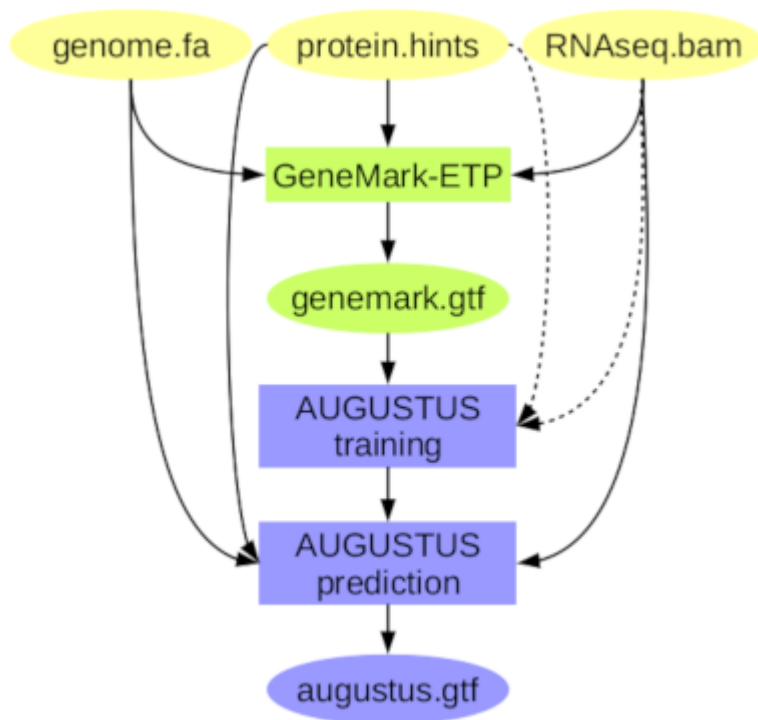
Figure 4: BRAKER pipeline D: training GeneMark-ETP supported by RNA-Seq alignment information and protein spliced alignment information (proteins can be of longer evolutionary distance), prediction with AUGUSTUS using the same alignment information. Introns supported by both RNA-Seq and protein alignment information are treated as "true positive introns", their prediction in gene structures by GeneMark-ETP and AUGUSTUS is enforced.

- Genome file and file with proteins of short evolutionary distance (see Figure 5); this approach is suitable if RNA-Seq data is not available and if the reference species is very closely related.
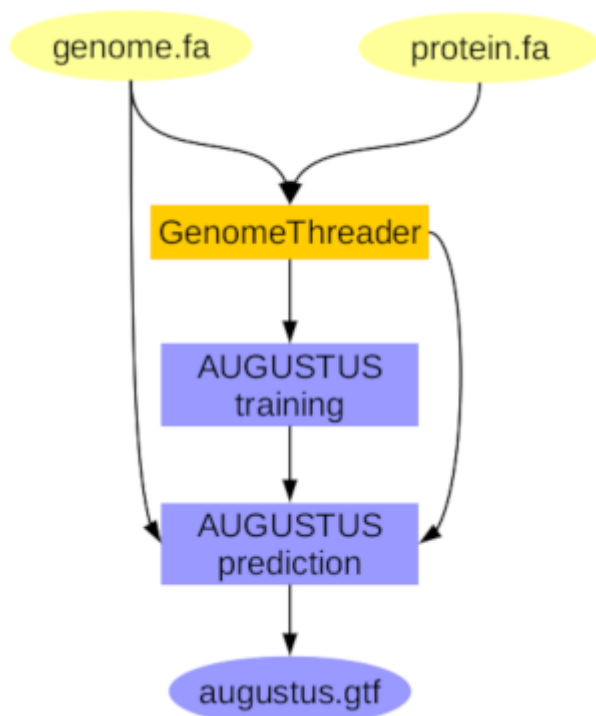
Figure 5: Additional pipeline B: training AUGUSTUS on the basis of spliced alignment information from proteins of a very closely related species against the target genome.

- Genome and RNA-Seq file and proteins of short evolutionary distance (see Figures 6 and 7). In both cases, GeneMark-ET is trained supported by RNA-Seq data, and the resulting gene predictions are used for training AUGUSTUS. In approach A), protein alignment information is used in the gene prediction step with AUGUSTUS, only. In approach C), protein spliced alignment data is used to complement the training set for AUGUSTUS. The latter approach is in particular suitable if RNA-Seq data does not produce a sufficiently high number of training gene structures for AUGUSTUS, and if a very closely related and already annotated species is available.
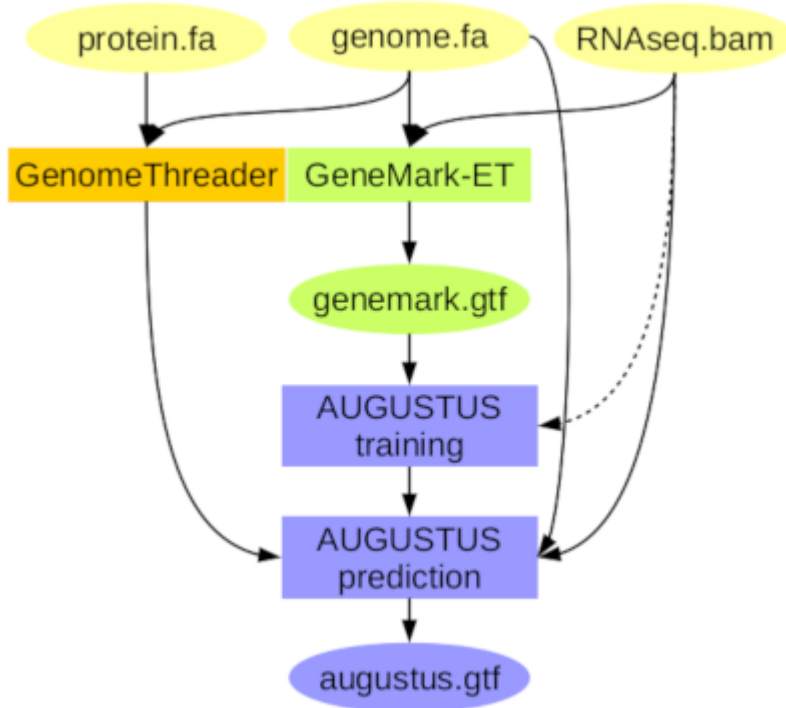
Figure 6: Additional pipeline A: training GeneMark-ET supported by RNA-Seq spliced alignment information, prediction with AUGUSTUS with spliced alignment information from RNA-Seq data and with gene features determined by alignments from proteins of a very closely related species against the target genome.
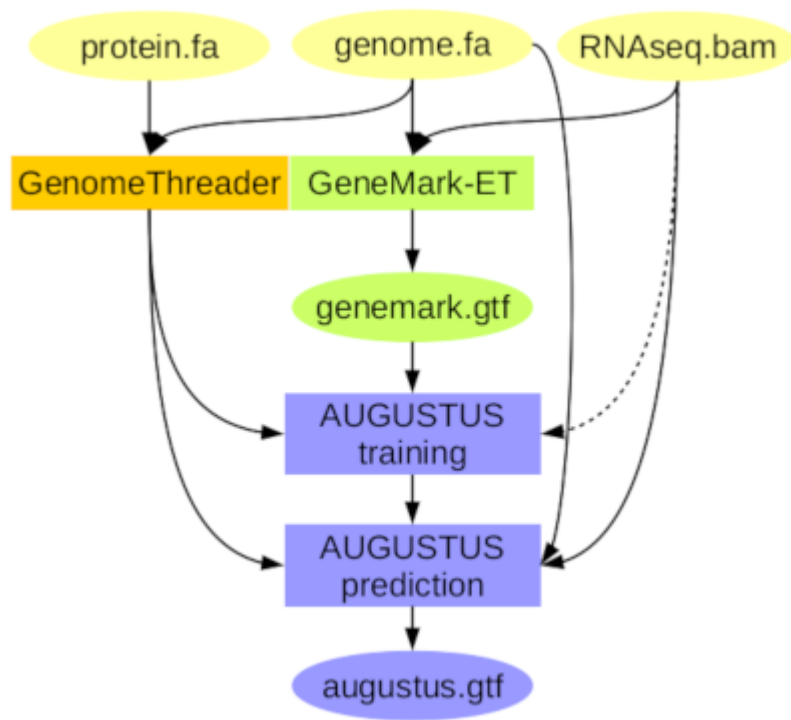
Figure 7: Additional pipeline C: training GeneMark-ET on the basis of RNA-Seq spliced alignment information, training AUGUSTUS on a set of training gene structures compiled from RNA-Seq supported gene structures predicted by GeneMark-ET and spliced alignment of proteins of a very closely related species.

Installation

Supported software versions

At the time of release, this BRAKER version was tested with:

- AUGUSTUS 3.3.1[F2]

- GeneMark-ET 4.33

- BAMTOOLS 2.5.1[R5]

- SAMTOOLS 1.7-4-g93586ed[R6]

- GenomeThreader 1.7.0[R7]

- (Spaln 2.3.1 [R8, R9, R10])[F3]

- (Exonerate 2.2.0 [R11])[F3]

- NCBI BLAST+ 2.2.31+ [R12, R13]

BRAKER

Perl pipeline dependencies

Running BRAKER requires a Linux-system with `bash` and Perl. Furthermore, BRAKER requires the following CPAN-Perl modules to be
installed:

- `File::Spec::Functions`

- `Hash::Merge`

- `List::Util`

- `Logger::Simple`

- `Module::Load::Conditional`

- `Parallel::ForkManager`

- `POSIX`

- `Scalar::Util::Numeric`

- `YAML`

On Ubuntu, for example, install the modules with CPANminus[F4]: `sudo cpanm Module::Name`, e.g. `sudo cpanm Hash::Merge`.

BRAKER also uses a Perl module `helpMod.pm` that is not available on CPAN. This module is part of the BRAKER release and does not require
separate installation.

If you do not have root permissions on the Linux machine, try setting up an **Anaconda** (https://www.anaconda.com/distribution/)
environment as follows:

```
wget https://repo.anaconda.com/archive/Anaconda3-2018.12-Linux-x86_64.sh
bash bin/Anaconda3-2018.12-Linux-x86_64.sh # do not install VS (needs root privileges)
conda install -c anaconda perl
conda install -c bioconda perl-app-cpanminus
conda install -c bioconda perl-hash-merge
conda install -c bioconda perl-parallel-forkmanager
conda install -c bioconda perl-scalar-util-numeric
conda install -c bioconda perl-yaml
conda install -c bioconda perl-class-data-inheritable
conda install -c bioconda perl-exception-class
conda install -c bioconda perl-test-pod
conda install -c anaconda biopython
conda install -c bioconda perl-file-homedir
conda install -c bioconda perl-file-which # skip if you are not comparing to reference annotation
cpanm Logger::Simple
```

BRAKER components

BRAKER is a collection of Perl scripts and a Perl module. The main script that will be called in order to run BRAKER is
`braker.pl`. Additional Perl components are:

- `align2hints.pl`

- `filterGenemark.pl`

- `filterIntronsFindStrand.pl`

- `startAlign.pl`

- `helpMod.pm`

- `findGenesInIntrons.pl`

- `downsample_traingenes.pl`

All Perl scripts (files ending with `*.pl`) that are part of BRAKER must be executable in order to run BRAKER. This should already be the case if you download BRAKER from GitHub. Executability may be overwritten if you e.g. transfer BRAKER on a USB-stick to another computer. In order to check whether required files are executable, run the following command in the directory that contains BRAKER Perl scripts:

```
ls -l *.pl
```

The output should be similar to this:

```
-rwxr-xr-x 1 katharina katharina  18191 Mai  7 10:25 align2hints.pl
-rwxr-xr-x 1 katharina katharina 408782 Aug 17 18:24 braker.pl
-rwxr-xr-x 1 katharina katharina   5024 Mai  7 10:25 downsample_traingenes.pl
-rwxr-xr-x 1 katharina katharina  30453 Mai  7 10:25 filterGenemark.pl
-rwxr-xr-x 1 katharina katharina   5754 Mai  7 10:25 filterIntronsFindStrand.pl
-rwxr-xr-x 1 katharina katharina   7765 Mai  7 10:25 findGenesInIntrons.pl
-rwxr-xr-x 1 katharina katharina  41674 Mai  7 10:25 startAlign.pl
```

It is important that the `x` in `-rwxr-xr-x` is present for each script. If that is not the case, run

```
`chmod a+x *.pl`
```

in order to change file attributes.

You may find it helpful to add the directory in which BRAKER perl scripts reside to your `$PATH` environment variable. For a single bash session, enter:

```
PATH=/your_path_to_braker/:$PATH
export PATH
```

To make this `$PATH` modification available to all bash sessions, add the above lines to a startup script (e.g. `~/.bashrc`).

Bioinformatics software dependencies

BRAKER calls upon various bioinformatics software tools that are not part of BRAKER. Some tools are obligatory, i.e. BRAKER will not run at all if these tools are not present on your system. Other tools are optional. Please install all tools that are required for running BRAKER in the mode of your choice.

Mandatory tools

GeneMark-EX

Download GeneMark-EX[F1] from http://exon.gatech.edu/GeneMark/license_download.cgi. Unpack and install GeneMark-EX as described in GeneMark-EX's `README` file.

If already contained in your `$PATH` variable, BRAKER will guess the location of `gmes_petap.pl` , automatically. Otherwise, BRAKER can find GeneMark-EX executables either by locating them in an environment variable `GENEMARK_PATH` , or by taking a command line argument (`--GENEMARK_PATH=/your_path_to_GeneMark-EX/gmes_petap/`).

In order to set the environment variable for your current Bash session, type:

```
export GENEMARK_PATH=/your_path_to_GeneMark-ET/gmes_petap/
```

Add the above lines to a startup script (e.g. `~/.bashrc`) in order to make it available to all bash sessions.[F5]

Important: GeneMark-EX will only run if a valid key file resides in your home directory. The key file will expire after 200 days, which means that you have to download a new GeneMark-EX release and a new key file after 200 days. The key file is downloaded as `gm_key.gz` . Unpack the key file and move it to a hidden file **in your home directory** as follows:

```
cd # change to your home directory
gunzip gm_key_64.gz
mv gm_key_64 .gm_key
```

If you are running GeneMark-EX in an Anaconda environment, modify the shebang of all GeneMark-EX scripts to use that perl version:

```
cd gm_et_linux_64/gmes_petap/
for f in bet_to_gff.pl bp_seq_select.pl build_mod.pl calc_introns_from_gtf.pl \
change_path_in_perl_scripts.pl gc_distr.pl get_sequence_from_GTF.pl \
gmes_petap.pl histogram.pl hmm_to_gtf.pl make_nt_freq_mat.pl \
parse_by_introns.pl parse_ET.pl parse_gibbs.pl parse_set.pl predict_genes.pl \
reformat_fasta.pl reformat_gff.pl rescale_gff.pl rnaseq_introns_to_gff.pl \
run_es.pl run_hmm_pbs.pl scan_for_bp.pl star_to_gff.pl verify_evidence_gmhmm.pl;
do
    cat $f | perl -pe 's/\/usr\/bin\/perl/\/usr\/bin\/env perl/' > $f.tmp
    mv $f.tmp $f
    chmod u+x $f
done
```

AUGUSTUS

Download AUGUSTUS from https://github.com/Gaius-Augustus/Augustus. Unpack AUGUSTUS and install AUGUSTUS according to AUGUSTUS `README.TXT` . *Do not use outdated AUGUSTUS versions from other sources! BRAKER highly depends in particular on an up-to-date Augustus/scripts directory!*

You should compile AUGUSTUS on your own system in order to avoid problems with versions of libraries used by AUGUSTUS. Compilation instructions are provided in the AUGUSTUS `README.TXT` file (`Augustus/README.txt`).

AUGUSTUS consists of `augustus` , the gene prediction tool, additional C++ tools located in `Augustus/auxprogs` and Perl scripts located in `Augustus/scripts` . Perl scripts must be executable (see instructions in section BRAKER components.

The C++ tool `bam2hints` is an essential component of BRAKER. Sources are located in `Augustus/auxprogs/bam2hints` . Make sure that you compile `bam2hints` on your system (it should be automatically compiled when AUGUSTUS is compiled, but in case of problems with `bam2hints` , please read troubleshooting instructions in `Augustus/auxprogs/bam2hints/README`).

If you would like to train UTR parameters and integrate RNA-Seq coverage information into gene prediction with BRAKER (which is possible only if an RNA-Seq bam-file is provided as extrinsic evidence) and `utrrnaseq` in the `auxprogs` directory are also required. If compilation with the default `Makefile` fails, please read troubleshooting instructions in `Augustus/auxprogs/utrrnaseq/README` .

Since BRAKER is a pipeline that trains AUGUSTUS, i.e. writes species specific parameter files, BRAKER needs writing access to the configuration directory of AUGUSTUS that contains such files (`Augustus/config/`). If you install AUGUSTUS globally on your system, the `config` folder will typically not be writable by all users. Either make the directory where `config` resides recursively writable to users of AUGUSTUS, or copy the `config/` folder (recursively) to a location where users have writing permission.

AUGUSTUS will locate the `config` folder by looking for an environment variable `$AUGUSTUS_CONFIG_PATH` . If the `$AUGUSTUS_CONFIG_PATH` environment variable is not set, then BRAKER will look in the path `../config` relative to the directory in which it finds an AUGUSTUS executable. Alternatively, you can supply the variable as a command line argument to BRAKER (`--AUGUSTUS_CONFIG_PATH=/your_path_to_AUGUSTUS/Augustus/config/`). We recommend that you export the variable e.g. for your current bash session:

```
export AUGUSTUS_CONFIG_PATH=/your_path_to_AUGUSTUS/Augustus/config/
```

In order to make the variable available to all Bash sessions, add the above line to a startup script, e.g. `~/.bashrc` .

Important:

BRAKER expects the entire `config` directory of AUGUSTUS at `$AUGUSTUS_CONFIG_PATH` , i.e. the subfolders `species` with its contents (at least `generic`) and `extrinsic` ! Providing an writable but empty folder at `$AUGUSTUS_CONFIG_PATH` will not work for BRAKER. If you need to separate augustus binary and `$AUGUSTUS_CONFIG_PATH` , we recommend that you recursively copy the un-writable config contents to a writable location.

You have a system-wide installation of AUGUSTUS at `/usr/bin/augustus` , an unwritable copy of `config` sits at `/usr/bin/augustus_config/` . The folder `/home/yours/` is writable to you. Copy with the following command (and additionally set the then required variables):

```
cp -r /usr/bin/Augustus/config/ /home/yours/
export AUGUSTUS_CONFIG_PATH=/home/yours/augustus_config
export AUGUSTUS_BIN_PATH=/usr/bin
export AUGUSTUS_SCRIPTS_PATH=/usr/bin/augustus_scripts
```

Modification of $PATH

Adding adding directories of AUGUSTUS binaries and scripts to your `$PATH` variable enables your system to locate these tools, automatically. It is not a requirement for running BRAKER to do this, because BRAKER will try to guess them from the location of another environment variable (`$AUGUSTUS_CONFIG_PATH`), or both directories can be supplied as command line arguments to `braker.pl` , but we recommend to add them to your `$PATH` variable. For your current bash session, type:

```
PATH=:/your_path_to_augustus/bin/:/your_path_to_augustus/scripts/:$PATH
export PATH
```

For all your BASH sessions, add the above lines to a startup script (e.g. `~/.bashrc`).

Bamtools

Download BAMTOOLS (e.g. `git clone https://github.com/pezmaster31/bamtools.git`). Install BAMTOOLS by typing the following in your shell:

```
        cd your-bamtools-directory mkdir build cd build cmake .. make
```

If already in your `$PATH` variable, BRAKER will find bamtools, automatically. Otherwise, BRAKER can locate the bamtools binary either
by using an environment variable `$BAMTOOLS_PATH` , or by taking a command line argument (`--BAMTOOLS_PATH=/your_path_to_bamtools/bin/` [F6]). In order to set the environment variable e.g. for your current bash session, type:

```
        export BAMTOOLS_PATH=/your_path_to_bamtools/bin/
```

Add the above line to a startup script (e.g. `~/.bashrc`) in order to set the environment variable for all bash sessions.

NCBI BLAST+

On Ubuntu, install with `sudo apt-get install ncbi-blast+` .

If already in your `$PATH` variable, BRAKER will find blastp, automatically. Otherwise, BRAKER can locate the blastp binary either by using an environment variable `$BLAST_PATH` , or by taking a command line argument (`--BLAST_PATH=/your_path_to_blast/`). In order to set the environment variable e.g. for your current bash session, type:

```
        export BLAST_PATH=/your_path_to_blast/
```

Add the above line to a startup script (e.g. `~/.bashrc`) in order to set the environment variable for all bash sessions.

Optional tools

Samtools

Samtools is not required for running BRAKER if all your files are formatted, correctly (i.e. all sequences should have short and unique
fasta names). If you are not sure whether all your files are fomatted correctly, it might be helpful to have Samtools installed because BRAKER can automatically fix certain format issues by using Samtools.

As a prerequisite for Samtools, download and install `htslib` (e.g.
`git clone https://github.com/samtools/htslib.git` , follow the `htslib` documentation for installation).

Download and install Samtools (e.g. `git clone git://github.com/samtools/samtools.git`), subsequently follow Samtools documentation for installation).

If already in your `$PATH` variable, BRAKER will find samtools, automatically. Otherwise, BRAKER can find Samtools either by taking a
command line argument (`--SAMTOOLS_PATH=/your_path_to_samtools/`), or by using an environment variable `$SAMTOOLS_PATH` .
For exporting the variable, e.g. for your current bash session, type:

```
        export SAMTOOLS_PATH=/your_path_to_samtools/
```

Add the above line to a startup script (e.g. `~/.bashrc`) in order to set the environment variable for all bash sessions.

Python3 and Biopython

If Python3 and Biopython are installed, BRAKER can generate FASTA-files with coding sequences and protein sequences predicted by AUGUSTUS and generate track data hubs for visualization of a BRAKER run with MakeHub [R16].
Both are an optional steps. The first can be disabled with the command-line flag `--skipGetAnnoFromFasta`, the latter can be activated by using the command-line options `--makehub --email=your@mail.de`; Python3 and Biopython are not required if neither of this steps shall be performed.

On Ubuntu, Python3 is installed by default. Install the Python3 package manager with:

```
`sudo apt-get install python3-pip`
```

Subsequently, install Biopython with:

```
`sudo pip3 install biopython`
```

On Ubuntu, python3 will be in your `$PATH` variable, by default, and BRAKER will automatically locate it. However, you have the option to specify the `python3` binary location in two other ways:

1. Export an environment variable `$PYTHON3_PATH`, e.g. in your `~/.bashrc` file:

    ```
    export PYTHON3_PATH=/path/to/python3/
    ```

2. Specify the command line option `--PYTHON3_PATH=/path/to/python3/` to `braker.pl`.

GenomeThreader

This tool is required, only, if you would like to run protein to genome alignments with BRAKER using GenomeThreader. This is a suitable approach if an annotated species of short evolutionary distance to your target genome is available. Download GenomeThreader from http://genomethreader.org/. Unpack and install according to `gth/README`.

BRAKER will try to locate the GenomeThreader executable by using an environment variable `$ALIGNMENT_TOOL_PATH`. Alternatively, this can be supplied as command line argument (`--ALIGNMENT_TOOL_PATH=/your/path/to/gth`).

Spaln

This tool is required, only, if you would like to run protein to genome alignments with BRAKER using Spaln. This is a suitable approach if an annotated species of short evolutionary distance to your target genome is available. (We recommend the usage of GenomeThreader instad of Spaln.) Download Spaln from https://github.com/ogotoh/spaln/releases. Unpack and install according to `spaln/doc/SpalnReadMe22.pdf`.

BRAKER will try to locate the Spaln executable by using an environment variable `$ALIGNMENT_TOOL_PATH`. Alternatively, this can be supplied as command line argument (`--ALIGNMENT_TOOL_PATH=/your/path/to/spaln`).

Exonerate

This tool is required, only, if you would like to run protein to genome alignments with BRAKER using Exonerate. This is a suitable approach if an annotated species of short evolutionary distance to your target genome is available. (We recommend the usage of GenomeThreader instad of Exonerate because Exonerate is comparably slower and has lower specificity than GenomeThreader.) Download Exonerate from https://github.com/nathanweeks/exonerate. Unpack and install according to `exonerate/README`. (On Ubuntu, download and install by typing `sudo apt-get install exonerate`.)

BRAKER will try to locate the Exonerate executable by using an environment variable `$ALIGNMENT_TOOL_PATH`. Alternatively, this can be supplied as command line argument (`--ALIGNMENT_TOOL_PATH=/your/path/to/exonerate`).

MakeHub

If you wish to automaticaly generate a track data hub of your BRAKER run, the MakeHub software, available at https://github.com/Gaius-Augustus/MakeHub is required. Download the software (either by running `git clone https://github.com/Gaius-Augustus/MakeHub.git`, or by picking a release from https://github.com/Gaius-Augustus/MakeHub/releases. Extract the release package if you downloaded a release (e.g. `unzip MakeHub.zip` or `tar -zxvf MakeHub.tar.gz`.

BRAKER will try to locate the make_hub.py script by using an environment variable `$MAKEHUB_PATH`. Alternatively, this can be supplied as command line argument (`--MAKEHUB_PATH=/your/path/to/MakeHub/`). BRAKER can also try to guess the location of MakeHub on your system.

Eval

In some (non de novo annotation) scenarios, BRAKER will be run to benchmark BRAKER predictions against an existing reference annotation (option `--annot=annot,gtf`). For doing this, BRAKER uses the Eval package [R17].

Eval is available for download from http://mblab.wustl.edu/software.html. After extraction, add the location of eval scripts to your `$PATH`:

```
PATH=/path/to/eval:$PATH
```

BRAKER will try to locate the eval scripts by searching `evaluate_gtf.pl` on your system, i.e. the eval scripts must be in your `$PATH` for usage with BRAKER.

Running BRAKER

Different BRAKER pipeline modes

In the following, we describe "typical" BRAKER calls for different input data types. In general, we recommend that you run BRAKER on genomic sequences that have been softmasked for Repeats. If your genome has been softmasked, include the `--softmasking` flag in your BRAKER call!

BRAKER with RNA-Seq data

This approach is suitable for genomes of species for which RNA-Seq libraries with a good coverage of the transcriptome are available. The pipeline is illustrated in Figure 1.

BRAKER can either extract RNA-Seq spliced alignment information from `bam` files, or it can use such extracted information, directly.

In order to run BRAKER with RNA-Seq data supplied as `bam` file(s) (in case of multiple files, separate them by comma), run:

```
braker.pl --species=yourSpecies --genome=genome.fasta \
    --bam=file1.bam,file2.bam
```

In order to run BRAKER with RNA-Seq spliced alignment information that has already been extracted, run:

```
braker.pl --species=yourSpecies --genome=genome.fasta \
    --hints=hints1.gff,hints2.gff
```

The format of such a hints file must be as follows (tabulator separated file):

```
chrName b2h intron  6591    8003    1   +   .   pri=4;src=E
chrName b2h intron  6136    9084    11  +   .   mult=11;pri=4;src=E
...
```

The source `b2h` in the second column and the source tag `src=E` in the last column are essential for BRAKER to determine whether a hint has been generated from RNA-Seq data.

Training and prediction of UTRs, integration of coverage information

If RNA-Seq (and only RNA-Seq) data is provided to BRAKER as a bam-file, and if the genome is softmasked for repeats, BRAKER can automatically train UTR parameters for AUGUSTUS. After successful training of UTR parameters, BRAKER will automatically predict genes including coverage information form RNA-Seq data. Example call:

```
braker.pl --species=yourSpecies --genome=genome.fasta \
    --bam=file.bam --softmasking --UTR=on
```

Warning: This feature is experimental!

Stranded RNA-Seq alignments

For running BRAKER without UTR paramters, it is not very important whether RNA-Seq data was generated by a *stranded* protocol (because spliced alignments are 'artificially stranded' by checking the splice site pattern). However, for UTR training and prediction, stranded libraries may provide information that is valuable for BRAKER.

After alignment of the stranded RNA-Seq libraries, separate the resulting bam file entries into two files: one for plus strand mappings, one for minus strand mappings. Call BRAKER as follows:

```
braker.pl --species=yourSpecies --genome=genome.fasta \
    --softmasking --bam=plus.bam,minus.bam --stranded=+,- \
     --UTR=on
```

You may additionally include bam files from unstranded libraries. Those files will not used for generating UTR training examples, but they will be included in the final gene prediction step as unstranded coverage information, example call:

```
braker.pl --species=yourSpecies --genome=genome.fasta \
    --softmasking --bam=plus.bam,minus.bam,unstranded.bam \
    --stranded=+,-,. --UTR=on
```

Warning: This feature is experimental!

BRAKER with proteins of unknown evolutionary distance

This approach is suitable for genomes of species for which no RNA-Seq libraries are available and if computational time for hints preparation is not an issue. A database of proteins (with possibly longer evolutionary distance to the target species) may be used in this case. The protein mapping pipeline is illustrated in figure 8.
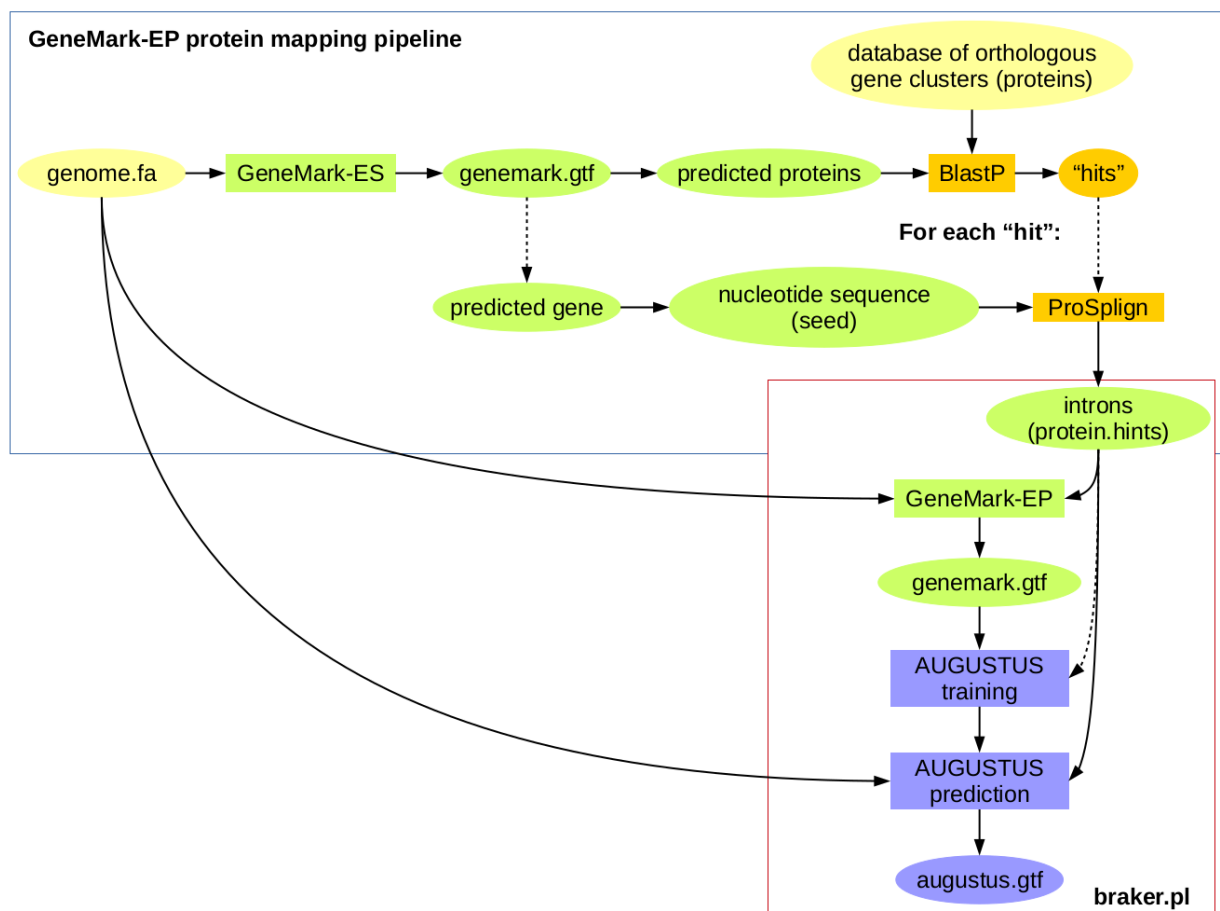
Figure 8: Protein mapping pipeline for proteins of unkown evolutionary distance (pipeline is maintained at Georgia Tech and may change at any point in time). Current version will not only produce intron hints, but also hints for CDS regions. Pipeline automatically determines which alignments are from close relatives, and which are from rather distant relatives.

Running BRAKER with proteins of longer evolutionary distance requires the preparation of "protein hints" before running BRAKER, itself. Preparing protein hints is in this case not part of BRAKER because in contrast to BRAKER, which can run on a work station with one or multiple cores, the GeneMark-EP specific protein mapping pipeline requires a cluster for execution.

For running BRAKER in this mode, type:

```
braker.pl --species=yourSpecies --genome=genome.fasta --hints=ep.hints --epmode
```

The format of such a hints file must be as follows (tabulator separated file):

```
chrName ProSplign   intron  6591    8003    5   +   .   mult=5;pri=4;src=P
chrName ProSplign   intron  6136    9084    11  +   .   mult=11;pri=4;src=P
...
```

The source `ProSplign` in the second column and the source tags `src=P` and `src=M` in the last column are essential for BRAKER.

BRAKER with proteins of short evolutionary distance

This approach is suitable if RNA-Seq data for the species of the target genome is not available and if a well annotated and very closely related reference species is available.

For running BRAKER in this mode, type:

```
braker.pl --species=yourSpecies --genome=genome.fasta \
    --prot_seq=proteins.fa --prg=gth \
    --ALIGNMENT_TOOL_PATH=/path/to/gth/binary \
    --trainFromGth
```

It is possible to generate protein alignments externally, prior running BRAKER, itself. The compatible command for running GenomeThreader prior running BRAKER, is:

```
gth -genomic genome.fa  -protein protein.fa -gff3out \
    -skipalignmentout -o gth.aln
```

In order to use such externally created alignment files, run:

```
braker.pl --species=yourSpecies --genome=genome.fasta \
    --prot_aln=proteins.aln --prg=gth --trainFromGth
```

It is also possible to run BRAKER in this mode using an already prepared hints file. In this case, run:

```
braker.pl --species=yourSpecies --genome=genome.fasta \
    --hints=hints.gff --prg=gth --trainFromGth
```

Format of the hints file should look like this:

```
chrName    gth2h   CDSpart 105984  106633  .    -    .    src=P;grp=FBpp0285205;pri=4
chrName    gth2h   start   106646  106648  .    -    .    src=P;grp=FBpp0285205;pri=4
```

Supported features in column 3 are intron, CDSpart, start, stop.

BRAKER with RNA-Seq and protein data

The native mode for running BRAKER with RNA-Seq and protein data is `--etpmode`. This will call GeneMark-ETP, which will use RNA-Seq and protein hints for training GeneMark-ETP. Hints that are supported by both sources and proteins hints of particularly high quality are enforced in gene prediction with GeneMark-ETP. Subsequently, AUGUSTUS is trained on GeneMark-ETP predictions and genes with hints are predicted by AUGUSTUS. To call the pipeline in this mode, run:

```
braker.pl --species=yourSpecies --genome=genome.fasta \
    --hints=ep_hints.gff,rnaseq.gff --etpmode
```

You can or course replace the `rnaseq.gff` hints file by a BAM-file, e.g. `--bam=ranseq.bam`.

In addition, the following pipelines can be executed by BRAKER:

- Adding protein data of short evolutionary distance to gene prediction step

- Extending training gene set with proteins of short evolutionary distance

Adding protein data of short evolutionary distance to gene prediction step

This pipeline is illustrated in Figure 6.

In general, add the options

```
        --prot_seq=proteins.fa --prg=(gth|exonerate|spaln)
```

to the BRAKER call that is described in section BRAKER with RNA-Seq data. Select one protein alignment tool from GenomeThreader (`gth` , recommended), Spaln (`spaln`) or Exonerate (`exonerate`). Of course, you may also specify the protein information as protein alignment files or hints files as described in section BRAKER with proteins of short evolutionary distance). This may result in a call similar to:

```
    braker.pl --species=yourSpecies --genome=genome.fasta \
        --bam=file1.bam,file2.bam --prot_seq=proteins.fa \
        --prg=(gth|exonerate|spaln)
```

Extending training gene set with proteins of short evolutionary distance

If the number of training gene structures identified by RNA-Seq data, only, seems to be too small, you may add training gene structures generated by protein alignments with GenomeThreader to the training gene set. This pipeline is illustrated in Figure 7.

In general, add the options

```
        --prot_seq=proteins.fa --prg=gth --gth2traingenes
```

to the BRAKER call that is described in section BRAKER with RNA-Seq data. This may result in a call similar to:

```
    braker.pl --species=yourSpecies --genome=genome.fasta \
        --bam=file1.bam,file2.bam --prot_seq=proteins.fa \
        --prg=gth --gth2traingenes
```

Description of selected BRAKER command line options

Please run `braker.pl --help` to obtain a full list of options.

--epmode

Run BRAKER in EP-mode, i.e. with protein hints from unknown evolutionary distance as produced by the GaTech protein mapping pipeline. Has to be provided with `--hints=hints.gff` .

--etpmode

Run BRAKER in ETP-mode, i.e. with protein hints from unknown evolutionary distance as produced by the GaTech protein mapping pipeline, and with RNA-Seq data. Has to be provided with `--hints=hints.gff` and additional RNA-Seq data can be provided either as additional hints file (comma separated), or as `--bam=rnaseq.bam` .

--ab_initio

Compute AUGUSTUS *ab initio* predictions in addition to AUGUSTUS predictions with hints (additional output files: `augustus.ab_initio.*` . This may be useful for estimating the quality of training gene parameters when inspecting predictions in a Browser.

--augustus_args=--some_arg=bla

One or several command line arguments to be passed to AUGUSTUS, if several arguments are given, separated by whitespace, i.e. `--first_arg=sth --second_arg=sth` . This may be be useful if you know that gene prediction in your particular species benefits from a particular AUGUSTUS argument during the prediction step.

--cores=INT

Specifies the maximum number of cores that can be used during computation. BRAKER has to run some steps on a single core, others can take advantage of multiple cores. The optimal core number of all steps is 8. If you use more than 8 cores, this will not speed up all parallelized steps, in particular, the time consuming `optimize_augustus.pl` will not use more than 8 cores. However, if you don't mind some cores being idle, using more than 8 cores will speed up other steps.

--fungus

GeneMark-EX option: run algorithm with branch point model. Use this option if you genome is a fungus.

--softmasking

Softmasking option for soft masked genome files. (Disabled by default.)

--useexisting

Use the present config and parameter files if they exist for 'species'. This step will skip training AUGUSTUS and instead use pre-trained parameters.

--crf

Execute CRF training for AUGUSTUS; resulting parameters are only kept for final predictions if they show higher accuracy than HMM parameters. This increases runtime!

--lambda=int

Change the parameter λ of the Poisson distribution that is used for downsampling training genes according to their number of introns (only genes with up to 5 introns are downsampled). The default value is $\lambda=2$. You might want to set it to 0 for organisms that mainly have single-exon genes. (Generally, single-exon genes contribute less value to increasing AUGUSTUS parameters compared to genes with many exons.)

--UTR=on

Generate UTR training examples for AUGUSTUS from RNA-Seq coverage information, train AUGUSTUS UTR parameters and predict genes with AUGUSTUS and UTRs, including coverage information for RNA-Seq as evidence. This flag only works if --softmasking is also enabled, and if the only extrinsic evidence provided are bam files.

--stranded=+,-,.,…

If `--UTR=on` is enabled, strand-separated bam-files can be provided with `--bam=plus.bam,minus.bam` . In that case, `--stranded=...` should hold the strands of the bam files (`+` for plus strand, `-` for minus strand, `.` for unstranded). Note that unstranded data will be used in the gene prediction step, only, if the parameter `--stranded=...` is set.

--makehub --email=your@mail.de

If `--makehub` and `--email=your@mail.de` (with your valid e-mail adress) are provided, a track data hub for visualizing results with the UCSC Genome Browser will be generated using MakeHub (https://github.com/Gaius-Augustus/MakeHub).

Output of BRAKER

BRAKER produces several important output files in the working directory.

- augustus.hints.gtf: Genes predicted by AUGUSTUS with intron hints from given extrinsic evidence. This file will be missing if BRAKER was run with the option `--esmode`.

- augustus.hints_utr.gtf: Genes predicted by AUGUSTUS with UTR parameters and coverage information from RNA-Seq data in GTF-format. The file will only be present if BRAKER was run with the option `--UTR=on` and a RNA-Seq BAM-file.

- augustus.ab_initio.gtf: Genes predicted by AUGUSTUS in *ab initio* mode in GTF-format. The file will always be present if AUGUSTUS has been run with the option `--esmode`. Otherwise, it will only be present if BRAKER was run with the option `--AUGUSTUS_ab_initio`.

- augustus.ab_initio_utr.gtf: Genes predicted by AUGUSTUS with UTR parameters in *ab initio* mode in GTF-format. This file will only be present if BRAKER was executed with the options `--UTR=on` and a RNA-Seq BAM-file, and with the option `--AUGUSTUS_ab_initio`.

- GeneMark-E*/genemark.gtf: Genes predicted by GeneMark-ES/ET in GTF-format. This file will be missing if BRAKER was executed with proteins of close homology and the option `--trainFromGth`.

- hintsfile.gff: The extrinsic evidence data extracted from RNAseq.bam and/or protein data. The introns are used for training GeneMark-ES/ET, all features are used for predicting genes with AUGUSTUS. The file is in GFF-format.

AUGUSTUS output files may be present with the following name endings and formats:

- GTF-format is always produced.

- GFF3-format is produced if the flat `--gff3` was specified to BRAKER.

- Coding sequences in FASTA-format are produced if the flag `--skipGetAnnoFromFasta` was not set.

- Protein sequence files in FASTA-format are produced if the flag `--skipGetAnnoFromFasta` was not set.

For details about gtf format, see http://www.sanger.ac.uk/Software/formats/GFF/. A GTF-format file contains one line per predicted exon. Example:

```
    HS04636 AUGUSTUS initial   966 1017 . + 0 transcript_id "g1.1"; gene_id "g1";
    HS04636 AUGUSTUS internal 1818 1934 . + 2 transcript_id "g1.1"; gene_id "g1";
```

The columns (fields) contain:

```
 seqname source feature start end score strand frame transcript ID and gene ID
```

If the --makehub option was used and MakeHub is available on your system, a hub directory beginning with the name `hub_` will be created. Copy this directory to a publicly accessible web server. A file `hub.txt` resides in the directory. Provide the link to that file to the UCSC Genome Browser for visualizing results.

Example data

An incomplete example data set is contained in the directory `BRAKER/example`. In order to complete the data set, please download the RNA-Seq alignment file (134 MB) with `wget`:

```
cd BRAKER/example
wget http://bioinf.uni-greifswald.de/bioinf/braker/RNAseq.bam
```

The example data set was not compiled in order to achieve optimal prediction accuracy, but in order to test pipeline components.

Data description

Data corresponds to *Drosophila melanogaster* chromosome 2R from flybase release R5, first 12000000 nucleotides.

RNA-Seq alignments were obtained by mapping Illumina paired-end librariy SRR023505 to the genome file using STAR with standard parameters (single pass mapping).

Protein sequences from Drosophila ananassae release R1.05 were aligned to the genome sequence of Drosophila melanogaster chromosome R2 using GenomeThreader with parameters `-gff3out -skipalignmentout -paralogs -prseedlength 20 -prhdist 2 -gcmincoverage 80 -prminmatchlen 20` . Protein sequence records of mapped proteins were stored in proteins.fa.

For generating protein hints from proteins of longer evolutionary distance, proteins from the eggNog database insect proportion were aligned to `Drosophila melanogaster` genome using the GaTech protein mapping pipline (excluding *Drosophila* species except for *D. grimshawi*, *D. virilis*, *D. willistoni*, *D. pseudoobscura*, *D. ananassae*).

List of files:

- `genome.fa` - genome file in fasta format

- `RNAseq.bam` - RNA-Seq alignment file in bam format (this file is not in github, it must be downloaded separately from http://bioinf.uni-greifswald.de/bioinf/braker/RNAseq.bam)

- `RNAseq.hints` - RNA-Seq hints (can be used instead of RNAseq.bam as RNA-Seq input to BRAKER)

- `prot.fa` - protein sequences of close homology in fasta format

- `ep.hints` - protein hints of remote homology in gff format

Testing BRAKER is time consuming because a full test requires the assembly of sufficient training data and subsequent training of gene predictors. Consider running BRAKER threaded (e.g. `--cores=8`) for testing. You can also select the `--skipOptimize` option for all tests that include training of AUGUSTUS in order to speed up testing.

The below given commands assume that you configured all paths to tools by exporting bash variables.

The example data set also contains scripts `tests/test*.sh` that will execute below listed commands for testing BRAKER with the example data set. You find example results of AUGUSTUS and GeneMark-EX in the folder `results/test*` . Be aware that BRAKER contains several parts where random variables are used, i.e. results that you obtain when running the tests must not be exactly identical.

We give runtime estimations derived from computing on a single core *Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz*.

Testing BRAKER with RNA-Seq data

The following command will test the pipeline according to Figure 2 (implemented in `test1.sh`):

```
braker.pl --genome=genome.fa --bam=RNAseq.bam \
    --softmasking
```

Runtime of this command is ~185 minutes.

Testing BRAKER with hints from proteins of unknown evolutionary distance

The following command will test the pipeline according to Figure 3 (implemented in `test2.sh`):

```
braker.pl --genome=genome.fa --hints=ep.hints \
    --epmode --softmasking
```

Runtime of this command is ~275 minutes.

Testing BRAKER with hints from proteins of unknown evolutionary distance and RNA-Seq

The following command will test a pipeline that first trains GeneMark-ETP with protein and RNA-Seq hints and subsequently trains AUGUSTUS on the basis of GeneMark-ETP predictions. AUGUSTUS predictions are also performed with hints from both sources, see Figure 4, implemented in `test3.sh`:

```
braker.pl --genome=genome.fa --hints=ep.hints \
    --bam=RNAseq.bam --etpmode --softmasking
```

Runtime of this command is ~380 minutes.

Testing BRAKER with proteins of close homology

The following command will test the pipeline according to Figure 5 (implemented in `test4.sh`):

```
braker.pl --genome=genome.fa --prot_seq=prot.fa \
    --prg=gth --trainFromGth --softmasking
```

Runtime of this command is ~137 minutes.

Testing BRAKER with proteins of close homology and RNA-Seq data (RNA-Seq supported training)

The following command will test the pipeline according to Figure 6 (implemented in `test5.sh`):

```
braker.pl --genome=genome.fa --prot_seq=prot.fa \
    --prg=gth --bam=RNAseq.bam --softmasking
```

Runtime of this command is ~214 minutes.

Testing BRAKER with proteins of close homoogy and RNA-Seq data (RNA-Seq and protein supported training)

The following command will test the pipeline according to Figure 7 (implemented in `test6.sh`):

```
braker.pl --genome=genome.fa --prot_seq=prot.fa \
    --prg=gth --bam=RNAseq.bam --gth2traingenes \
    --softmasking
```

Runtime of this command is ~346 minutes.

Testing BRAKER with pre-trained parameters

The training step of all pipelines can be skipped with the option `--skipAllTraining`. This means, only AUGUSTUS predictions will be performed, using pre-trained, already existing parameters. For example, you can predict genes with the command (implemented in `test7.sh`):

```
braker.pl --genome=genome.fa --bam=RNAseq.bam \
    --species=fly --skipAllTraining --softmasking
```

Runtime of this command is ~54 minutes.

Testing BRAKER with genome sequence

Implemented in `test8.sh`. Call:

```
braker.pl --genome=genome.fa --esmode --softmasking
```

Runtime of this command is ~606 minutes.

Bug reporting

Before reporting bugs, please check that you are using the most recent versions of GeneMark-EX, AUGUSTUS and BRAKER. Also, check the list of Common problems, before reporting bugs.

Reporting bugs on github

If you found a bug, please open an issue at https://github.com/Gaius-Augustus/BRAKER/issues (or contact katharina.hoff@uni-greifswald.de).

Information worth mentioning in your bug report:

Check in `braker/yourSpecies/braker.log` at which step `braker.pl` crashed.

There are a number of other files that might be of interest, depending on where in the pipeline the problem occured. Some of the following files will not be present if they did not contain any errors.

- `braker/yourSpecies/errors/bam2hints.*.stderr` - will give details on a bam2hints crash (step for converting bam file to intron gff file)

- `braker/yourSpecies/hintsfile.gff` - is this file empty? If yes, something went wrong during hints generation - does this file contain hints from source "b2h" and of type "intron"? If not: GeneMark-ET will not be able to execute properly.

- `braker/yourSpecies/startAlign.stderr` - if you provided a protein fasta file and this file is not empty, something went wrong during protein alignment

- `braker/yourSpecies/startAlign.stdout` - may give clues on at which point protein alignment went wrong

- `braker/yourSpecies/(align_gthalign_exoneratealign_spaln)/*err` - errors reported by the alignment tools gth/exonerate/spaln

- `braker/yourSpecies/errors/GeneMark-ET.stderr` - errors reported by GeneMark-ET

- `braker/yourSpecies/errors/GeneMark-ET.stdout` - may give clues about the point at which errors in GeneMark-ET occured

- `braker/yourSpecies/GeneMark-ET/genemark.gtf` - is this file empty? If yes, something went wrong during executing GeneMark-ET

- `braker/yourSpecies/GeneMark-ET/genemark.f.good.gtf` - is this file empty? If yes, something went wrong during filtering GeneMark-ET genes for training AUGUSTUS

- `braker/yourSpecies/genbank.good.gb` - try a "grep -c LOCUS genbank.good.gb" to determine the number of training genes for training AUGUSTUS, should not be low

- `braker/yourSpecies/errors/firstetraining.stderr` - contains errors from first iteration of training AUGUSTUS

- `braker/yourSpecies/errors/secondetraining.stderr` - contains errors from second iteration of training AUGUSTUS

- `braker/yourSpecies/errors/optimize_augustus.stderr` - contains errors optimize_augustus.pl (additional training set for AUGUSTUS)

- `braker/yourSpecies/errors/augustus*.stderr` - contain AUGUSTUS execution errors

Common problems

- *BRAKER complains that the RNA-Seq file does not correspond to the provided genome file, but I am sure the files correspond to each other!*

  Please check the headers of the genome FASTA file. If the headers are long and contain whitespaces, some RNA-Seq alignment tools will truncate sequence names in the BAM file. This leads to an error with BRAKER. Solution: shorten/simplify FASTA headers in the genome file before running the RNA-Seq alignment and BRAKER.

- *There are duplicate Loci in the `train.gb` file (after using GenomeThreader)!*

  This issue arises if outdated versions of AUGUSTUS and BRAKER are used. Solution: Please update AUGUSTUS and BRAKER from github (https://github.com/Gaius-Augustus/Augustus, https://github.com/Gaius-Augustus/BRAKER).

- *GeneMark fails!*

  (a) GeneMark requires a valid hidden key file in your home directory (`~/.gm_key`). The fiel expires after 200 days. Please check whether you have a valid key file before reporting an issue about this. Also, BRAKER may issue a WARNING that GeneMark is likely going to file due to limited extrinsic evidence. If you see that warning, please don't open an issue but try a different approach towards annotating your genome. For example, you can add more evidence data, you can try the protein mapping pipeline approach, you can try running `--esmode` without extrinsic evidence, …

  (b) GeneMark by default only uses contigs longer than 50k for training. If you have a highly fragmented assembly, this might lead to "no data" for training. You can override the default minimal length by setting the BRAKER argument `--min_contig=10000` .

- *BRAKER cannot find the Augustus script XYZ…*

  Update Augustus from github with `git clone https://github.com/Gaius-Augustus/Augustus.git`. Do not use Augustus from other sources. BRAKER is highly dependent on an up-to-date Augustus. Augustus releases happen rather rarely, updates to the Augustus scripts folder occur rather frequently.

- *Does BRAKER depend on Python3?*

  Partially. The options `-{}-{}make_hub` and `-{}-{}UTR` will require Python3. The general required for Python3 for generating e.g. the protein sequence output file can be disabled with `--skipGetAnnoFromFasta`. So, if you use BRAKER with `--skipGetAnnoFromFasta` and not with `-{}-{}make_hub` and `-{}-{}UTR`, BRAKER does not require Python3. The python scripts employed by BRAKER are not compatible with Python2.

Citing BRAKER and software called by BRAKER

Since BRAKER is a pipeline that calls several Bioinformatics tools, publication of results obtained by BRAKER requires that not only BRAKER is cited, but also the tools that are called by BRAKER:

- Always cite:

  - Hoff, K.J., Lomsadze, A., Borodovsky, M. and Stanke, M. (2019). Whole-Genome Annotation with BRAKER. Methods Mol Biol. 1962:65-95, doi: 10.1007/978-1-4939-9173-0_5.

  - Hoff, K.J., Lange, S., Lomsadze, A., Borodovsky, M. and Stanke, M. (2015). BRAKER1: unsupervised RNA-Seq-based genome annotation with GeneMark-ET and AUGUSTUS. Bioinformatics, 32(5):767-769.

  - Stanke, M., Diekhans, M., Baertsch, R. and Haussler, D. (2008). Using native and syntenically mapped cDNA alignments to improve de novo gene finding. Bioinformatics, doi: 10.1093/bioinformatics/btn013.

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

- If any kind of AUGUSTUS training was performed by BRAKER, NCBI BLAST will have been used, therefore cite:

  - Altschul, A.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. (1990). A basic local alignment search tool. J Mol Biol 215:403--410.

  - Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., and Madden, T.L. (2009). Blast+: architecture and applications. BMC bioinformatics, 10(1):421.

- If BRAKER was executed with a genome file and no extrinsic evidence, cite, then GeneMark-ES was used, cite:

  - Lomsadze, A., Ter-Hovhannisyan, V., Chernoff, Y.O. and Borodovsky, M. (2005). Gene identification in novel eukaryotic genomes by self-training algorithm. Nucleic Acids Research, 33(20):6494--6506.

  - Ter-Hovhannisyan, V., Lomsadze, A., Chernoff, Y.O. and Borodovsky, M. (2008). Gene prediction in novel fungal genomes using an ab initio algorithm with unsupervised training. Genome research, pages gr--081612, 2008.

- If BRAKER was executed with RNA-Seq information or with information from proteins of remote homology, then GeneMark-EX was used; cite:

  - Lomsadze, A., Burns, P.D. and Borodovsky, M. (2014). Integration of mapped RNA-Seq reads into automatic training of eukaryotic gene finding algorithm. Nucleic Acids Research, 42(15):e119.

- If BRAKER was executed with RNA-Seq alignments in bam-format, then SAMtools was used, cite:

- Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R.; 1000 Genome Project Data Processing Subgroup (2009). The Sequence Alignment/Map format and SAMtools. Bioinformatics, 25(16):2078-9.

- Barnett, D.W., Garrison, E.K., Quinlan, A.R., Strömberg, M.P. and Marth G.T. (2011). BamTools: a C++ API and toolkit for analyzing and managing BAM files. Bioinformatics, 27(12):1691-2

- If BRAKER was executed with proteins of closely related species, cite GenomeThreader:

  - Gremme, G. (2013). Computational Gene Structure Prediction. PhD thesis, Universität Hamburg.

- If BRAKER called MakeHub for creating a track data hub for visualization of BRAKER results with the UCSC Genome Browser, cite:

  - Hoff, K.J. (2019) MakeHub: Fully automated generation of UCSC Genome Browser Assembly Hubs. Preprint on bioarXive, doi: https://doi.org/10.1101/550145.

License

All source code, i.e. `scripts/*.pl` or `scripts/*.py` are under the Artistic License (see http://www.opensource.org/licenses/artistic-license.php).

Footnotes

[F1] EX = ES/ET/EP/ETP, all available for download under the name *GeneMark-ES/ET* ↩

[F2] Please use the latest version of AUGUSTUS distributed by the original developers, it is available from github at https://github.com/Gaius-Augustus/Augustus. Problems have been reported from users that tried to run BRAKER with AUGUSTUS releases maintained by third parties, i.e. Bioconda. ↩

[F3] Not tested in this release, we recommend using GenomeThreader, instead ↩

[F4] install with `sudo apt-get install cpanminus` ↩

[F5] GeneMark-EX is not a mandatory tool if AUGUSTUS is to be trained from GenomeThreader aligments with the option `--trainFromGth`. ↩

[F6] The binary may e.g. reside in bamtools/build/src/toolkit ↩

References

[R1] Hoff, Katharina J, Simone Lange, Alexandre Lomsadze, Mark Borodovsky, and Mario Stanke. 2015. "BRAKER1: Unsupervised Rna-Seq-Based Genome Annotation with Genemark-et and Augustus." *Bioinformatics* 32 (5). Oxford University Press: 767--69.↩

[R2] Lomsadze, Alexandre, Paul D Burns, and Mark Borodovsky. 2014. "Integration of Mapped Rna-Seq Reads into Automatic Training of Eukaryotic Gene Finding Algorithm." *Nucleic Acids Research* 42 (15). Oxford University Press: e119--e119.↩

[R3] Stanke, Mario, Mark Diekhans, Robert Baertsch, and David Haussler. 2008. "Using Native and Syntenically Mapped cDNA Alignments to Improve de Novo Gene Finding." *Bioinformatics* 24 (5). Oxford University Press: 637--44.↩

[R4] Stanke, Mario, Oliver Schöffmann, Burkhard Morgenstern, and Stephan Waack. 2006. "Gene Prediction in Eukaryotes with a Generalized Hidden Markov Model That Uses Hints from External Sources." *BMC Bioinformatics* 7 (1). BioMed Central: 62.↩

[R5] Barnett, Derek W, Erik K Garrison, Aaron R Quinlan, Michael P Strömberg, and Gabor T Marth. 2011. "BamTools: A C++ Api and Toolkit for Analyzing and Managing Bam Files." *Bioinformatics* 27 (12). Oxford University Press: 1691--2.↩

[R6] Li, Heng, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. 2009. "The Sequence Alignment/Map Format and Samtools." *Bioinformatics* 25 (16). Oxford University Press: 2078--9.↩

[R7] Gremme, G. 2013. "Computational Gene Structure Prediction." PhD thesis, Universität Hamburg.↩

[R8] Gotoh, Osamu. 2008a. "A Space-Efficient and Accurate Method for Mapping and Aligning cDNA Sequences onto Genomic Sequence." *Nucleic Acids Research* 36 (8). Oxford University Press: 2630--8.↩

[R9] Iwata, Hiroaki, and Osamu Gotoh. 2012. "Benchmarking Spliced Alignment Programs Including Spaln2, an Extended Version of Spaln That Incorporates Additional Species-Specific Features." *Nucleic Acids Research* 40 (20). Oxford University Press: e161--e161.↩

[R10] Osamu Gotoh. 2008b. "Direct Mapping and Alignment of Protein Sequences onto Genomic Sequence." *Bioinformatics* 24 (21). Oxford University Press: 2438--44.↩

[R11] Slater, Guy St C, and Ewan Birney. 2005. "Automated Generation of Heuristics for Biological Sequence Comparison." *BMC Bioinformatics* 6(1). BioMed Central: 31.↩

[R12] Altschul, S.F., W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. 1990. "Basic Local Alignment Search Tool." *Journal of Molecular Biology* 215:403--10.↩

[R13] Camacho, Christiam, et al. 2009. "BLAST+: architecture and applications." *BMC Bioinformatics* 1(1): 421.↩

[R14] Lomsadze, A., V. Ter-Hovhannisyan, Y.O. Chernoff, and M. Borodovsky. 2005. "Gene identification in novel eukaryotic genomes by self-training algorithm." *Nucleic Acids Research* 33 (20): 6494--6506. doi:10.1093/nar/gki937.↩

[R15] Ter-Hovhannisyan, Vardges, Alexandre Lomsadze, Yury O Chernoff, and Mark Borodovsky. 2008. "Gene Prediction in Novel Fungal Genomes Using an Ab Initio Algorithm with Unsupervised Training." *Genome Research*. Cold Spring Harbor Lab, gr--081612.↩

[R16] Hoff, K.J. 2019. MakeHub: Fully automated generation of UCSC Genome Browser Assembly Hubs. Preprint on bioarXive, doi: https://doi.org/10.1101/550145.↩