BRAKER2 User Guide

Katharina J. Hoff

March 13, 2018

Contents

Chapter 1

Introduction

1.1 What is BRAKER2?

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 [Hoff et al., 2015], a combination of GeneMark-ET [Lomsadze et al., 2014] and AUGUSTUS [Stanke et al., 2008], that uses genomic and RNA-Seq data to automatically generate full gene structure annotations in novel genomes.

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 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.

BRAKER2 can also combine RNA-Seq and protein homology information. This is in an early stage of implementation, though, and will improve with future releases.

1.2 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 `>contig1|my custom species name|some 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. 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-ET and AUGUSTUS within BRAKER2. 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 3.2 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.

1.3 Overview of modes for running BRAKER2

BRAKER2 mainly features semi-unsupervised, extrinsic evidence data (RNA-Seq and/or protein spliced alignment information) supported training of GeneMark-EX 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 BRAKER2. In the following, we give an overview of possible input files and pipelines:

- genome and RNA-Seq file from the same species (see figure 1.1A); 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,
- genome file and database of proteins that may be of longer evolutionary distance to the target species (see figure 1.1B); this approach is 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,
- genome and RNA-Seq file from the same species, and proteins that may be of longer evolutionary distance to the target species (see figure 1.1C); **important:** this approach requires a database of protein families, i.e. many representatives of each protein family must be present in the database,
- genome file and file with proteins of short evolutionary distance (see figure 1.2B); this approach is suitable if RNA-Seq data is not available and if the reference species is very closely related,
- genome and RNA-Seq file and proteins of short evolutionary distance (see figures 1.2A and 1.2C). 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.

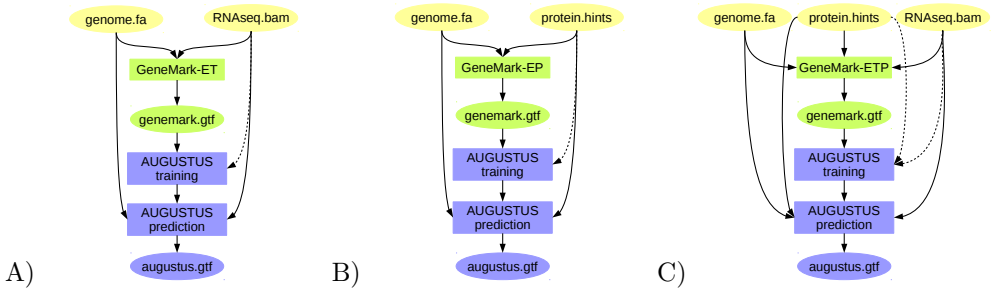

Figure 1.1: BRAKER2 pipeline A) training GeneMark-ET supported by RNA-Seq spliced alignment information, prediction with AUGUSTUS with that same spliced alignment information, B) training GeneMark-EP on protein spliced alignment information, prediction with AUGUSTUS with that same spliced alignment information. Proteins used for B) can be of longer evolutionary distance, C) 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.

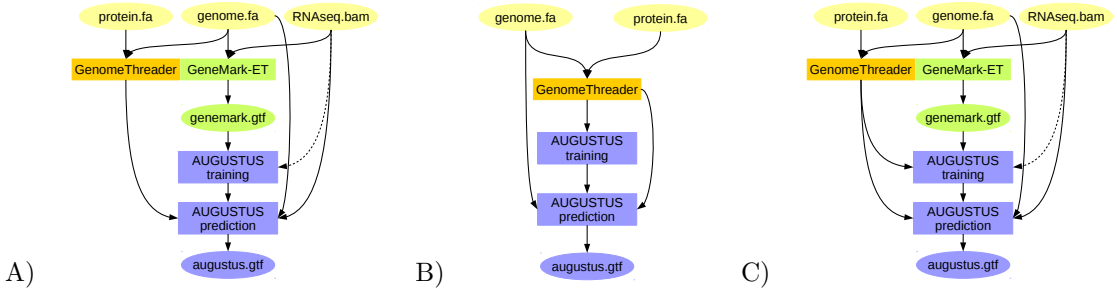

Figure 1.2: BRAKER2 pipeline A) training GeneMark-ET supported by RNA-Seq spliced alignment information, prediction with AUGUSTUS with spliced alignment information from RNA-Seq data and with gene features determined by alignments from proteins of a very closely related species against the target genome, B) training AUGUSTUS on the basis of spliced alignment information from proteins of a very closely related species against the target genome, 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.

Chapter 2

Installation

2.1 Supported software versions

At the time of release, this BRAKER2 version was tested with:

- AUGUSTUS 3.3[1]
- GeneMark-EX 4.33
- BAMTOOLS 2.5.1
- SAMTOOLS 1.7-4-g93586ed
- GenomeThreader 1.7.0
- (Spaln 2.3.1)[2]
- (Exonerate 2.2.0)[3]
- NCBI BLAST+ 2.2.31+

2.2 BRAKER2

2.2.1 Perl pipeline dependencies

BRAKER2 is implemented in Perl. Running BRAKER2 requires a Linux-system with `bash` and Perl. Furthermore, BRAKER2 requires the following CPAN-Perl modules to be installed:

- `File::Spec::Functions`
- `Hash::Merge`
- `List::Util`
- `Logger::Simple`

[1] Please use AUGUSTUS distributed by the original developers, it is available at `http://bioinf.uni-greifswald.de/augustus/binaries/augustus.current.tar.gz`. This BRAKER version has been tested with the future release, i.e. 3.4, which is not officially avilable, yet, at the date of BRAKER2 release. AUGUSTUS release maintained by third parties, i.e. on Bioconda, have been reported to cause problems with BRAKER2.

[2] Not tested in this version, we recommend using GenomeThreader, instead

[3] Not tested in this version, we recommend using GenomeThreader, instead

- Module::Load::Conditional
- Parallel::ForkManager
- POSIX
- Scalar::Util::Numeric
- YAML

On Ubuntu, for example, install the modules with CPANminus[4]: `sudo cpanm [Module::Name]`, e.g. `sudo cpanm Hash::Merge`.

BRAKER2 also uses a Perl module `helpMod.pm` that is not available on CPAN. This module is part of the BRAKER2 release and does not require separate installation.

2.2.2 BRAKER2 components

BRAKER2 is a collection of Perl scripts and a Perl module. The main script that will be called in order to run BRAKER2 is `braker.pl`. Additional Perl components are:

- align2hints.pl
- filterGenemark.pl
- filterIntronsFindStrand.pl
- startAlign.pl
- helpMod.pm

All Perl scripts (files ending with `*.pl`) that are part of BRAKER2 must be executable in order to run BRAKER2. This should already be the case if you download BRAKER2 from our website. Executability may be overwritten if you e.g. transfer BRAKER2 on a USB-stick to anothre computer. In order to check whether required files are executable, run the following command in the directory that contains BRAKER2 Perl scripts:

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

The output should be similar to this:

```
-rwxr-xr-x 1 braker braker  13802 Jan 12 14:23 align2hints.pl
-rwxr-xr-x 1 braker braker 159944 Jan 16 16:42 braker.pl
-rwxr-xr-x 1 braker braker  21217 Jul  3  2017 filterGenemark.pl
-rwxr-xr-x 1 braker braker   5716 Jul 12  2017 filterIntronsFindStrand.pl
-rwxr-xr-x 1 braker braker  32123 Jan 12 14:23 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 BRAKER2 perl scripts reside to your `$PATH` environment variable. For a single bash session, enter:

[4]install with `sudo apt-get install cpanminus`

```
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.*sim*`/.bashrc`).

2.3 Bioinformatics software dependencies

BRAKER2 calls upon various bioinformatics software tools that are not part of BRAKER2. Some tools are obligatory, i.e. BRAKER2 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 BRAKER2 in the mode of your choice.

2.3.1 Mandatory tools

2.3.1.1 GeneMark-EX

Download GeneMark-EX 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, BRAKER2 will guess the location of `gmes_petap.pl`, automatically. Otherwise, BRAKER2 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.[5]

2.3.1.2 AUGUSTUS

Download AUGUSTUS from `http://bioinf.uni-greifswald.de/augustus/downloads/index.php`. Unpack AUGUSTUS and install AUGUSTUS according to AUGUSTUS `README.TXT`.

AUGUSTUS comes with pre-compiled binaries (located in the `augustus-x.x.x/bin` folder). However, 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-x.x.x/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 2.2.2 on page 7).

The C++ tool `bam2hints` is an essential component of BRAKER2. Sources are located in `augustus-x.x.x/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-x.x.x/auxprogs/bam2hints/README`).

Since BRAKER2 is a pipeline that trains AUGUSTUS, i.e. writes species specific parameter files, BRAKER2 needs writing access to the configuration directory of AUGUSTUS that contains such files

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

(augustus-x.x.x/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 BRAKER2 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 BRAKER2 (--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: BRAKER2 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.

Example:

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 \texttt{/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 $PATH variable enables your system to locate these tools, automatically. It is not a requirement for running BRAKER2 to do this, because BRAKER2 will try to guess them from the location of another environment variable ($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.*sim*/.bashrc).

2.3.1.3 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 `$PATH` variable, BRAKER2 will find bamtools, automatically. Otherwise, BRAKER2 can locate the bamtools binary either by using an environment variable `$BAMTOOLS_PATH`, or by taking a command line argument (`--BAMTOOLS_PATH=/your_path_to_bamtools/bin/`[6]). 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.

2.3.1.4 NCBI BLAST+

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

If already in your `$PATH` variable, BRAKER2 will find blastp, automatically. Otherwise, BRAKER2 can locate the blastp binary either by using an environment variable `$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.

2.3.2 Optional tools

2.3.2.1 Samtools

Samtools is not required for running BRAKER2 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 BRAKER2 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 `$PATH` variable, BRAKER2 will find samtools, automatically. Otherwise, BRAKER2 can find Samtools either by taking a command line argument (`--SAMTOOLS_PATH=/your_path_to_samtools/`), or by using an environment variable `$SAMTOOLS_PATH`. For exporting the variable, e.g. for your current bash session, type:

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

```
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.

2.3.2.2 GenomeThreader

This tool is required, only, if you would like to run protein to genome alignments with BRAKER2 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`.

BRAKER2 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`).

2.3.2.3 Spaln

This tool is required, only, if you would like to run protein to genome alignments with BRAKER2 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 `http://www.genome.ist.i.kyoto-u.ac.jp/~aln_user`. Unpack and install according to `spaln/doc/SpalnReadMe22.pdf`.

BRAKER2 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`).

2.3.2.4 Exonerate

This tool is required, only, if you would like to run protein to genome alignments with BRAKER2 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`.)

BRAKER2 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`).

Chapter 3

Running BRAKER2

3.1 Different BRAKER2 pipeline modes

In the following, we describe "typical" BRAKER2 calls for different input data types.

3.1.1 BRAKER2 with RNA-Seq data (only)

.

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 1.1A) on page 5.

BRAKER2 can either extract RNA-Seq spliced alignment information from `bam` files, or it can use such extracted information, directly.

In order to run BRAKER2 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 BRAKER2 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 BRAKER2 to determine whether a hint has been generated from RNA-Seq data.

3.1.2 BRAKER2 with proteins of longer evolutionary distance

This approach is suitable for genomes of species for which no RNA-Seq libraries are available and for which no closely related and well annotated genome is available. A database of proteins with longer

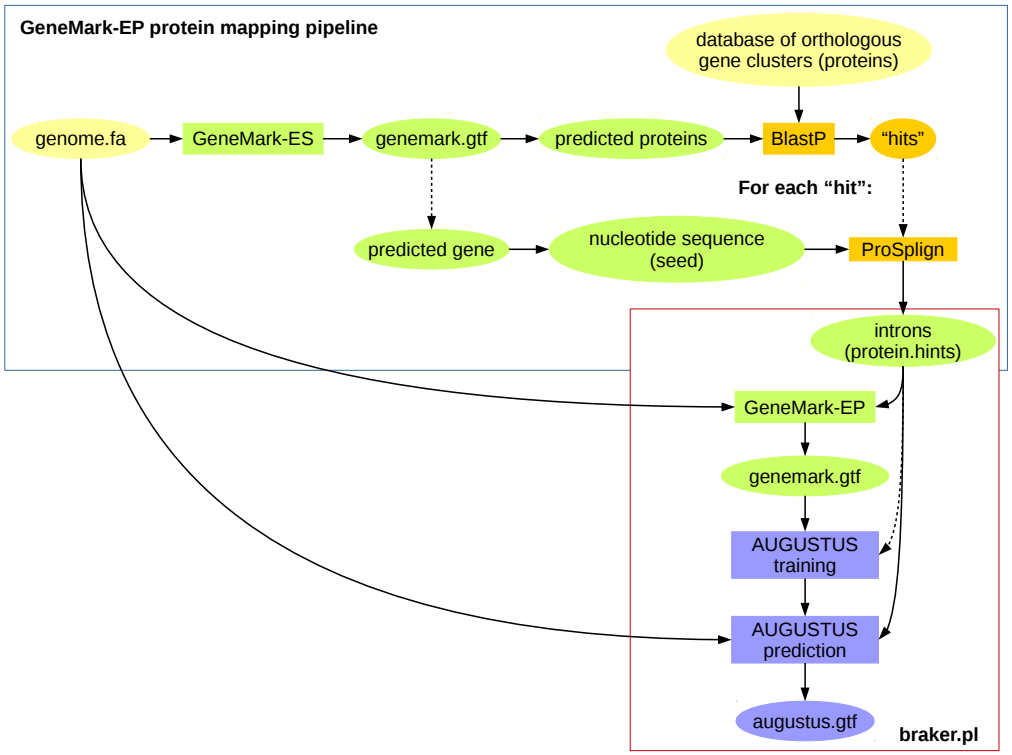

Figure 3.1: BRAKER2 and GeneMark-EP protein mapping pipeline.

evolutionary distance to the target species may be used in this case. The pipeline is illustrated in figure 3.1 on page 13.

Running BRAKER2 with proteins of longer evolutionary distance requires the preparation of "protein hints" before running BRAKER2, itself. Preparing protein hints is in this case not part of BRAKER2 because in contrast to BRAKER2, 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 BRAKER2 in this mode, type:

```
braker.pl --species=yourSpecies --genome=genome.fasta \
   --hints=hints.gff --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 tag `src=P` in the last column are essential for BRAKER2 to determine whether a hint has been generated from remote homology protein data.

3.1.3 BRAKER2 with proteins of shorter 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. The pipeline is illustrated in figure 1.2B) on page 5.

For running BRAKER2 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 BRAKER2, itself. The compatible command for running GenomeThreader prior running BRAKER2, 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 BRAKER2 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 are intron, CDSpart, start, stop.

3.1.4 BRAKER2 with RNA-Seq and protein data

BRAKER2 with RNA-Seq and protein data is currently still under development. BRAKER2 currently does not train GeneMark-EX from protein and RNA-Seq data, yet. However, if RNA-Seq data of the target species and protein data of a very closely related reference species are available, BRAKER2 already supports the following to modes.

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

This pipeline is illustrated in figure 1.2A) on page 5.

In general, add the options

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

to the BRAKER2 call that is described in section 3.1.1. 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 3.1.3 on page 13). 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)
```

3.1.4.2 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 1.2C) on page 5.

In general, add the options

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

to the BRAKER2 call that is described in section 3.1.1. 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
```

3.2 Description of selected BRAKER2 command line options

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

3.2.1 `--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.

3.2.2 `--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.

3.2.3 `--cores`

Specifies the maximum number of cores that can be used during computation. BRAKER2 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.

3.2.4 `--fungus`

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

3.2.5 `--softmasking`

Softmasking option for soft masked genome files. Set to 'on' or '1'. Recommended to use this option!

3.2.6 `--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.

3.2.7 `--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!

Chapter 4

Output of BRAKER2

BRAKER2 produces three important output files in the working directory:

```
hintsfile.gff             - The extrinsic evidence data extraced from RNAseq.bam and/or protein
                            data. The introns are used for training GeneMark-EX, all features
                            are used for predicting genes with AUGUSTUS. The file
                            is in gff format.
GeneMark-ET/genemark.gtf  - Genes predicted by GeneMark-EX in gtf format (RNA-Seq trained)
GeneMark-EP/genemark.gtf  - Genes predicted by GeneMark-EX in gtf format (Protein trained)
augustus.hints.gff        - Genes predicted by AUGUSTUS with hints from given extrinsic evidence
                            in gtf format
```

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
```

Chapter 5

Example data

Due to file size, example data for testing BRAKER2 is separately available for download at `http://bioinf.uni-greifswald.de/augustus/downloads/index.php` and `http://exon.gatech.edu/` as archive `BRAKER2examples.tar.gz`. The example data set was not compiled in order to achieve optimal prediction accuracy, but in order to test all pipeline components.

5.1 Data description

Data corresponds to *Drosophila melanogaster* chromosome 2R from flybase release R5.

RNA-Seq alignments were obtained by mapping Illumina paired-end libraries SRR023505, SRR023546, SRR023608, SRR026433, SRR027108 against the complete genome using Tophat2 with the options `--microexon-search --mate-std-dev 26 --mate-inner-dist 38 --min-intron-length 30 --min-coverage-intron 30 --min-segment-intron 30`. The example data set contains only those RNA-Seq alignments that correspond to sequence 2R.

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.

List of files:

- `genome.fa` - genome file in fasta format
- `RNAseq.bam` - RNA-Seq alignment file in bam format
- `RNAseq.hints` - RNA-Seq hints (can be used instead of RNAseq.bam as RNA-Seq input to BRAKER2)
- `prot.fa` - protein sequences of close homology in fasta format
- `ep.hints` - protein hints of remote homology in gff format

Testing BRAKER2 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 `test*.sh` that will execute below listed commands for testing BRAKER2 with the example data set. You find example results of AUGUSTUS and GeneMark-EX in the folder `results/test*`. Be aware that BRAKER2 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*.

5.2 Testing BRAKER2 with RNA-Seq (only) data (`test1.sh`)

The following command will test the pipeline according to figure 1.1A:

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

Runtime of this command is $\sim$ 581 minutes.

5.3 Testing BRAKER2 with hints from proteins of remote homology (only) (`test2.sh`)

The following command will test the pipeline according to figure 1.1B:

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

Runtime of this command is $\sim$ 492 minutes.

5.4 Testing BRAKER2 with hints from proteins of remote homology and RNA-Seq (`test3.sh`)

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.

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

Runtime of this command is $\sim$ 526 minutes.

5.5 Testing BRAKER2 with proteins of close homology (only) (`test4.sh`)

The following command will test the pipeline according to figure 1.2B:

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

Runtime of this command is $\sim$ 216 minutes.

5.6 Testing BRAKER2 with proteins of close homology and RNA-Seq data (RNA-Seq supported training) (`test5.sh`)

The following command will test the pipeline according to figure 1.2A:

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

Runtime of this command is $\sim$ 576 minutes.

5.7 Testing BRAKER2 with proteins of close homoogy and RNA-Seq data (RNA-Seq and protein supported training) (`test6.sh`

The following command will test the pipeline according to figure 1.2C:

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

Runtime of this command is $\sim$ 496 minutes.

5.8 Testing BRAKER2 with pre-trained parameters (prediction only) (`test7.sh`)

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:

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

Runtime of this command is $\sim$ 102 minutes.

Chapter 6

Bug reporting

If you found a bug, please 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_gth|align_exonerate|align_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

Bibliography

[Hoff et al., 2015] 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.

[Lomsadze et al., 2014] 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–e119.

[Stanke et al., 2008] Stanke, M., Diekhans, M., Baertsch, R., and Haussler, D. (2008). Using native and syntenically mapped cdna alignments to improve de novo gene finding. *Bioinformatics*, 24(5):637–644.