GeMoMa manual

April 2, 2019

Abstract

Gene Model Mapper (GeMoMa) is a homology-based gene prediction program. GeMoMa uses the annotation of protein-coding genes in a reference genome to infer the annotation of protein-coding genes in a target genome. Thereby, GeMoMa utilizes amino acid and intron position conservation. In addition, GeMoMa allows to incorporate RNA-seq evidence for splice site prediction.

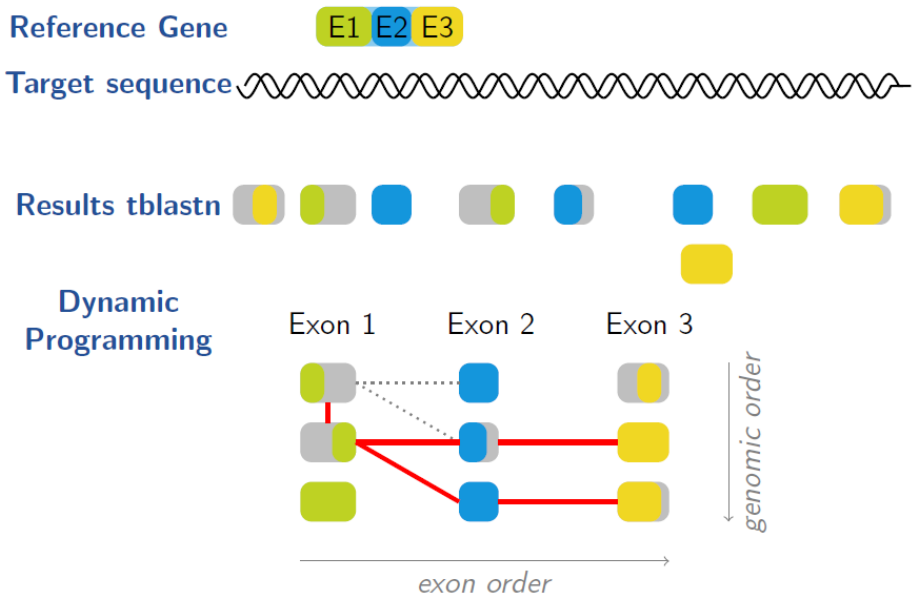

Figure 1: Illustration of the GeMoMa algorithm. GeMoMa uses `tblastn` to search for homologs of all (partially) coding exons of the reference transcript. Subsequently, a dynamic programming algorithm is used to determine the best combination of the hits.

1 GeMoMa in a nutshell

There are 7 steps that are divided into 3 phases. However, if you do not have any RNA-seq data you can skip the first phase. This guide only provides a rough overview. If your are interested in individual parameters you can call:

```
java -jar GeMoMa-<version>.jar CLI <toolname>
```

which will provide descriptions of all available parameters.

Keep in mind that homology-based gene prediction only allows us to infer genes with known peers in at least one reference organism. Hence, homology based gene prediction might miss genes with no known peer in the reference organism(s), even if these are expressed in the RNA-seq data. On the other hand, homology-based gene prediction allows for inferring genes that are not or rarely expressed in the RNA-seq samples.

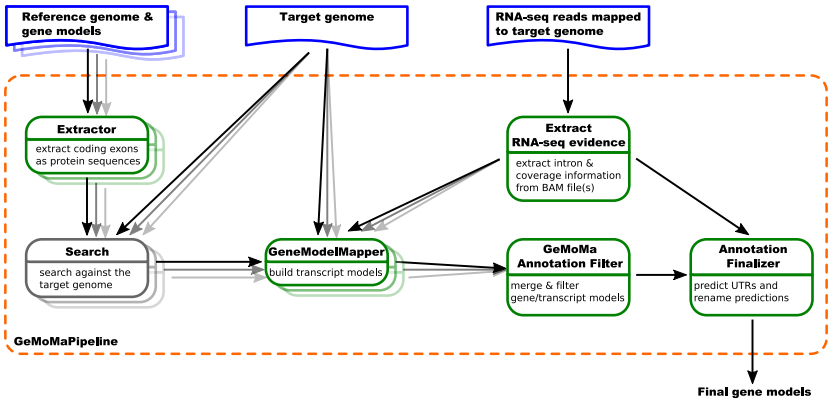

Figure 2: GeMoMa workflow. Blue items represent input data sets, green boxes represent GeMoMa modules, while grey boxes represent external modules. The GeMoMa Annotation Filter allows to combine predictions from different reference species. RNA-seq data is optional.

1.1 Quick start

We provide three scripts:

1. A test script checking whether GeMoMa runs on your system on some tiny toy data: `tests.sh`. Example data can be found in directory `test_data`.
2. An application script allowing to start the complete GeMoMa workflow with minimal parameter input using the individual modules: `run.sh`. If you like to run the workflow with standard parameters and without any tricks for speed up you can just use this script.
   - Without RNA-seq data:

     ```
     ./run.sh <ref-anno> <ref-genome> <target-genome> <out-dir>
     ```

   - With RNA-seq data:

     ```
     ./run.sh <ref-anno> <ref-genome> <target-genome> <out-dir>
              <lib-type> <mapped-reads>
     ```

3. An application script allowing to start the complete GeMoMa workflow with minimal parameter input using the GeMoMaPipeline module: `pipeline.sh`. If you like to run the workflow with standard parameters and without any tricks for speed up you can just use this script.
   - Without RNA-seq data:

     ```
     ./pipeline.sh <threads> <target-genome> <ref-anno>
     <ref-genome> <out-dir>
     ```

   - With RNA-seq data:

     ```
     ./pipeline.sh <threads> <target-genome> <ref-anno>
     <ref-genome> <out-dir>
     <lib-type> <mapped-reads>
     ```

where

1. `ref-anno` is the annotation of the reference organism (GFF/GTF)
2. `ref-genome` is the genome of the reference organism (FastA)
3. `target-genome` is the genome of the target organism (FastA)
4. `out-dir` is the output directory
5. `lib-type` is the RNA-seq library type (`{FR_UNSTRANDED, FR_FIRST_STRAND, FR_SECOND_STRAND}`)
6. `mapped-reads` are the mapped RNA-seq reads (SAM/BAM)

1.2 Phase 1: Preprocessing RNA-seq data

For this phase, you need RNA-seq data and a reference sequence or assembly of your target organism. This allows for inferring experimental verified introns and coverage. In the following command lines " . . . " should be replaced by your specific parameters.

1. Run your favorite read mapper (e.g. TopHat, . . .) on your RNA-seq data
2. Extract introns (and coverage) running:

   ```
   java -jar GeMoMa-<version>.jar CLI ERE ...
   ```

1.3 Phase 2: Prediction candidate transcripts

For this phase, you need at least one reference organism with reference sequence/assembly and annotation as well as a reference sequence/assembly for the target organism. Keep in mind that the outcome of GeMoMa highly depends on the quality of the reference annotation. It is possible to run step 3 to 5 with multiple reference organisms.

3. Extract CDS-parts (and proteins) of the reference organism running:

   ```
   java -jar GeMoMa-<version>.jar CLI Extractor...
   ```

4. Find homologous parts in the target organism running:

   ```
   tblastn -outfmt "6 std sallseqid score nident positive gaps
            ppos qframe sframe qseq sseq qlen slen salltitles" ...
   ```

   The output format of tblastn is essential for GeMoMa. Since version 1.6, we also allow to use mmseqs as search algorithm. Mmseqs is typically faster than tblastn.

5. Find homologous candidate transcripts in the target organism running:

   ```
   java -jar GeMoMa-<version>.jar CLI GeMoMa ...
   ```

   Since version 1.4, p=10 and ct=0.4 are new default values.

1.4 Phase 3: Aggregate and filter predictions

For this phase, you need only the output of step 5. If you ran the second phase for multiple reference organisms, the individual predictions can now be combined in this step.

6. Aggregate predictions running:

   ```
   java -jar GeMoMa-<version>.jar CLI GAF ...
   ```

7. Predict UTRs and rename predictions running:

   ```
   java -jar GeMoMa-<version>.jar CLI AnnotationFinalizer ...
   ```

2 Miscellaneous

2.1 Help section and parameter description

If you like to receive more information about the available tools and parameters please enter

```
java -jar GeMoMa-<version>.jar CLI
```

and follow the instructions.

2.2 General speed-up

If you

1. like to speed-up the computation and
2. have lots of compute cores,

you might accelerate GeMoMa using the following tricks:

2.2.1 RNA-seq data

If you have a lot of RNA-seq samples you can map them independently. Afterwards, you can run ERE on each of the mapped read data sets (SAM/BAM). However, if you like to use the results of these runs in GeMoMa it is a bit annoying to set a lot of intron and coverage files as parameters. For this reason, we implemented two helper classes allowing to combine introns and coverage form independent runs:

```
java -cp GeMoMa-<version>.jar projects.gemoma.CombineIntronFiles
     <output name> <input0> <input1> ...
java -cp GeMoMa-<version>.jar projects.gemoma.CombineCoverageFiles
     <output name> <input0> <input1> ...
```

Instead of listing all files, you can also use wildcards, e.g., ERE/*/*.bam, to easily combine individuals files.

2.2.2 Search and GeMoMa

If you have large data sets, i.e., a large number of lines in the CDS-parts file, you can split the job in several independent jobs. Splitting the job can be done using

```
java -cp GeMoMa-<version>.jar projects.FastaSplitter <CDS-parts>
     <numberOfSplits> "_"
```

This will result in several independent FastA files containing some CDS-parts. You can run steps 4 and 5 on these individual parts. Finally, you have to concatenate the resulting predicted annotations, protein, ... before starting step 6.

2.3 GeMoMaPipeline

If you only have a compute server and no compute cluster, the general speed-up described in the last section can directly be addressed without any manual interplay using the module GeMoMaPipeline. The module runs the complete pipeline as one job and allows to exploit the complete compute power of a server using multi-threading. However, this module can **only** exploit the compute power of a single server and does not distribute the partial jobs to a compute cluster.

If you like to receive more information about GeMoMaPipeline and its parameters please enter

```
java -jar GeMoMa-<version>.jar GeMoMaPipeline
```

and follow the instructions.

2.4 Genome-wide vs. specific genes

By default, GeMoMa tries to make predictions for all CDS of the reference organism in the target organism. Hence, we call this a genome-wide approach. Additionally, Extractor and GeMoMa have an option for filtering based on a list of some reference CDS (cf. parameter `selected`). This approach allows to further speed up the predictions especially if one is interested only in a certain part of the annotation.

2.5 Galaxy integration

GeMoMa provides a command line interface as well as everything to be integrated in your local Galaxy instance. For creating the XML files that are needed for integration into Galaxy, just run:

```
createGalaxyIntegration.sh <version>
```

Alternatively, you can run the individual commands on your own using the command line

```
java -jar GeMoMa-<version>.jar --create
```

If you like to modify some VM arguments for the java calls made by Galaxy you can alter them for each sub-tool individually. If you like to allow GeMoMa to allocate at most 40GB of RAM, you may do so by running

```
java -jar GeMoMa-<version>.jar --create GeMoMa -Xmx40g
```

to create the corresponding XML integration for GeMoMa.

3 Reference, questions and comments

If you use GeMoMa please cite:

Using intron position conservation for homology-based gene prediction.
J. Keilwagen, M. Wenk, J. L. Erickson, M. H. Schattat, J. Grau, and F. Hartung.
Nucleic Acids Research, 2016. doi: 10.1093/nar/gkw092

Combining RNA-seq data and homology-based gene prediction for plants, animals and fungi.
J. Keilwagen, F. Hartung, M. Paulini, S. O. Twardziok, and J. Grau.
BMC Bioinformatics, 2018. doi: 10.1186/s12859-018-2203-5

FAQs, complete parameter lists, release notes, ... can be found on the homepage: `http://www.jstacs.de/index.php/GeMoMa`

For further questions or comments, please contact:
jens.keilwagen@julius-kuehn.de