# Table of Contents

1. Overview .................................................. 2
1.2 Features .................................................. 2
1.3 Cite AGOUTI .................................................. 2
1.4 Obtain AGOUTI .................................................. 2
1.5 Get Started .................................................. 3
1.6 Prepare Inputs .................................................. 3
1.7 Understand Outputs .............................................. 5
1.8 Example .................................................. 5
1.9 Example Data .................................................. 6
1.10 Scaffolding on Shredded Assembly ...................... 6
1.11 Break-and-Continue ........................................... 9
1.12 Graph Visualization .......................................... 9
1.13 Support .................................................. 9
CHAPTER 1

Table of Contents

• Overview
• Features
• Cite AGOUTI
• Obtain AGOUTI
• Get Started
• Prepare Inputs
  – Genome Assembly
  – SAM/BAM File
  – Gene Models
• Understand Outputs
• Example
• Example Data
• Scaffoldding on Shredded Assembly
  – Why shredding original assemblies
  – When shredding original assemblies
  – Shredding Practices
  – Shred Assembly
  – Shred Annotation
  – Recover Original Paths
  – Report Inconsistencies
• Break-and-Continue
• Graph Visualization
1.1 Overview

AGOUTI uses paired-end RNA-seq reads to guide genome scaffolding and improve gene annotation. It works in the following steps:

1. Extracting uniquely mapped joining-read pairs
2. Denoise the set of joining-pairs using gene models
3. Traverse the graph built from the noise-free data to identify scaffolding paths
4. Reconcile each scaffolding path by applying rules defined in denoise step
5. Updating assembly and gene annotation given the scaffolds

1.2 Features

1. Scaffold hundreds to thousands of contigs, yielding more contiguous assemblies;
2. Reduce the number of gene models and update them simultaneously;
3. Shred genome assembly and corresponding annotation simultaneously;
4. Record any inconsistencies with the original (input) scaffolding results;
5. Recover original scaffolds from contigs not scaffolded by AGOUTI;
6. Support break-and-continue feature such that some time-consuming steps can be skipped if the previous run is successful;
7. Output graph in dot format for visualization

1.3 Cite AGOUTI

AGOUTI is published here:

Please cite AGOUTI using the following format:

Important: please submit your problem through GitHub issue tracker.

1.4 Obtain AGOUTI

AGOUTI is designed to be light-weight by only requiring SAMtools, python 2.7 or above, and git 1.8.5 or later.
To download AGOUTI, please use:
git clone https://github.com/svm-zhang/AGOUTI.git
To get the current version of the download, simply go to AGOUTI folder and do

```
python agouti.py --version
```

You should see, for example, something like this:

```
AGOUTI v0.2.4-dirty
```

Since v0.2.4, you can check if there is a new version by running:

```
python agouti.py update
```

It is RECOMMENDED to run update before you run AGOUTI.

If you wish to use a specific version of AGOUTI, you can first fetch all the versions available:

```
git fetch --all
```

Then show all available versions using:

```
git tag
```

Next you simply need to specify a version. For example, if you’d like to use v0.2.3:

```
git checkout v0.2.3 -b v0.2.3
```

You can also click on releases from AGOUTI github page to see all available versions.

## 1.5 Get Started

In its simplest usage, AGOUTI takes three inputs: an initial genome assembly in FASTA format, paired-end RNA-seq reads mapped against the assembly in BAM format, and gene predictions from the initial assembly in GFF3 format. For instance:

```
python agouti.py scaffold \
 -assembly example.fasta \
 -bam example.bam \
 -gff example.gff \
 -outdir ./example
```

This will produce a scaffolded assembly in FASTA format, and a updated gene models in GFF3 format. All files (including the intermediate files) will be stored under a directory specified by `--outdir`, “example” in this case.

## 1.6 Prepare Inputs

### 1.6.1 Genome Assembly

AGOUTI accepts assemblies as both contigs and scaffolds. In its scaffold form, AGOUTI breaks assemblies at gaps of a minimum length, essentially producing a shredded/split assembly (see [Shred Assembly](#)). AGOUTI scaffolds on the split assembly, and report any inconsistencies between the RNA-based scaffolding and the original scaffolding.

To shred a given assembly at gaps of at least 25 bp:
This produces a shredded assembly: example.ctg.fasta, and a file of a format similar to Fasta: example.shred.info.txt. Each header gives IDs of sequences in the original assembly. Under each header is a list of pairs of shredded contigs and the length of gaps between them. A sequence without any gaps will be by itself, and NA are used for such cases.

It is very important to use this split assembly in the following reads-mapping and gene prediction.

1.6.2 SAM/BAM File

Assuming you have a dataset of paired-end RNA-seq reads, example.1.fq and example.2.fq, and an assembly generated from either an assembler of your favorite or shredded by AGOUTI, example.fasta or example.ctg.fasta. You will first need to map the RNA-seq data against the assembly using a short-reads mapper, such as Bowtie2 or BWA. For example,

```
bwa index example.fasta
bwa mem -M example.fasta example.1.fq example.2.fq | samtools view -Sb - > example.bam
```

This produces a mapping results in BAM format. AGOUTI uses this BAM file for scaffolding. More specifically, it reads the file and extracts joining-pairs. A joining-pair is defined as one with both ends mapped to different contigs. AGOUTI uses only uniquely mapped ones by checking mapping quality. Short-reads mappers such as BWA, Bowtie2 uses a non-zero mapping quality to define unique mapping. If the mapper you are using does not use quality to mark ambiguous mapping, then you must first process your SAM/BAM file before running AGOUTI.

Several more things worth of noting:

1. Please run samtools flagstat to get stats of the mapping, and looks particular for number of pairs mapped to different chromosomes. If none, then AGOUTI will not be able to do any scaffolding.
2. Please make sure the BAM is sorted by reads name, not coordinates.

1.6.3 Gene Models

To run AGOUTI, you will also need a set of gene models predicted from the assembly. For instance,

```
augustus --AUGUSTUS_CONFIG_PATH=[path to augustus config file] -gff3=on --species=[your species] example.fasta > example.gff
```

At the end of gene prediction, you will now have a set of gene models predicted from the assembly. You can choose any * ab initio * gene predictor as long as it spits out the models in GFF format. More specifically, AGOUTI looks for the following information:

- lines annotated as gene
  - contig ID
  - gene ID, e.g. ID=genel from the attribute column (i.e. last column)
  - start and stop positions of the gene
  - strand
- lines annotated as CDS
Important Notes

1. AGOUTI is yet to support the GTF format. It will be in the near future. I will also try to provide a converter script from GTF to GFF.
2. If your GFF file has FASTA sequences at the end (e.g. generated from MAKER pipeline), please make sure to use versions v0.2.5 or above.
3. If AGOUTI fails to find any gene models, it will stop.

1.7 Understand Outputs

AGOUTI outputs its results to a base directory specified by `-outdir`. Under the base directory, there are several sub-folders created, each corresponding to a step built in AGOUTI. A run of AGOUTI using the command-line setting demonstrated in Getting Started will generated a structured output as shown in the following screenshot:

![Fig. 1.1: Alt text](image)

Each subfolder includes three types of file:

1. general progress meter info
2. debug info
3. intermediate outputs

To get a file with debug info you will need to specify `-debug`. An intermediate file can have all the joining-pairs, the denoised set of joining-pairs, the graph in DOT format, etc. Some intermediate files are important to support the break-and-continue feature, e.g. the file with the noise-free set of joining-pairs (see below for more details).

The `agouti.main.log` is prefixed with the string specified by `-p`, so do all the other files generated by AGOUTI. The sequence ID in the final assembly will also be as this prefix. By default, `agouti` will be used.

The final assembly and the updated gene models can be found under the base directory, example, along with plain text files of useful information, such as scaffolding paths, gene paths, differences between scaffolds generated by AGOUTI and original scaffolding.

1.8 Example

Scaffolding using joining-pairs with a minimum mapping quality of 20, a maximum of 5% mismatches:

```bash
gcommit 3.0 or above)
```

Scaffolding without updating gene model (v0.3.0 or above):

```bash
gcommit 3.0 or above)
```
Scaffolding a shredded assembly and report any inconsistencies between RNA-seq based scaffolding and original scaffolding:

```
python agouti.py scaffold \
-assembly example.shred.fasta \
-bam example.bam \
-gff example.gff \
-outdir ./example \
-shredpath example.shred.info.txt
```

Shredding an assembly and annotation simultaneously (**v0.3.0 or above**):

```
python agouti.py shred -assembly example.fasta -gff example.gff -p example
```

This will generate `example.shred.info.txt` and `example.shred.ctg.gff`

### 1.9 Example Data

Here gives one example data set that we used in our paper.

### 1.10 Scaffolding on Shredded Assembly

#### 1.10.1 Why shredding original assemblies

There are two benefits you can get from shredding the original assembly (you can optionally skip this entire section if your assembly is in the contig form, and no previous scaffolding is attempted). First, in the case of a gene spanning across a gap, the prediction tends to report two gene models, one for each side of the gap. This is because, to our knowledge, many programs cannot predict across gaps, especially those longer ones. Breaking at the gap and using RNA-seq data, AGOUTI therefore can correct for it by merging the two gene models, given there were connections between the two shredded contigs.

Second, scaffolding using RNA-seq reads can produce alternative paths that are based on evidences of gene models. Any inconsistencies with ones given by DNA-based scaffolding can provide useful information for further improving genome assembly.

The downside of scaffolding this way is that sequences, especially those from regions of low gene density, lose their context with others. This makes all efforts of doing DNA-based scaffolding, if any, become futile. To avoid such loss, AGOUTI (**v.0.3.0 or above**) tries to recover the original connections between contigs as much as possible (see **Recovering Original Paths** section below).

#### 1.10.2 When shredding original assemblies

It’s always recommended that you run AGOUTI directly on your scaffolds, before trying to tear it up. AGOUTI will simply try to find additionally connections between scaffolds that were missed by original scaffolding programs. This should be the firs best practice to do, regardless of how many pieces your assembly is composed of.

If you’d like to fix gene models flanking gaps and/or identify any inconsistencies from your original DNA-based scaffolding, shredding the assembly can be helpful. We are currently working on a new module that can correct for
split gene models interrupted by gaps, without shredding the assembly. This way AGOUTI can preserve as much as possible the contiguity, and further improve genome annotation at the same time. This module will be available soon.

1.10.3 Shredding Practices

First: If you shred the assembly and predict gene model on the shredded assembly using programs like AUGUSTUS, the following command line is suggested:

```bash
python agouti.py shred -assembly scaffold.fasta -p scaffold
```

This will generate `scaffold.ctg.fasta`, `scaffold.shred.info.txt`, and two files for debugging purpose. You then run, for instance AUGUSTUS and BWA, on the shredded assembly to get `scaffold.ctg.gff` and `scaffold.ctg.bam`, respectively. To scaffold, run

```bash
python agouti.py scaffold \
-assembly scaffold.ctg.fasta \
-bam scaffold.ctg.bam \
-gff scaffold.ctg.gff \
-outdir ./example \
-shredpath scaffold.shred.info.txt
```

With the `scaffold.shred.info.txt`, AGOUTI will try to recover the original scaffolding path. To disable this feature, you can simply not specify `-shredpath` option.

Second: Many people found laborious to repeat gene prediction on the shredded assembly, especially in cases the genome is huge. AGOUTI handle such cases by simultaneously shredding the annotation company the sequence. The only difference is to specify `-gff` option in the shred command line.

```bash
python agouti.py shred -assembly scaffold.fasta -gff scaffold.gff -p scaffold
```

In addition to the files described above, this also generates `scaffold.shred.ctg.gff`. This gene annotation is then used for scaffolding.

```bash
python agouti.py scaffold \
-assembly scaffold.ctg.fasta \
-bam scaffold.ctg.bam \
-gff scaffold.shred.ctg.gff \
-outdir ./example \
-shredpath scaffold.shred.info.txt
```

In this scenario, when AGOUTI tries to recover the original path, it will also connect the shredded gene models accordingly (see below).

1.10.4 Shred Assembly

Given an assembly in its scaffold form, AGOUTI can shred scaffolds into contigs at gaps of a minimum length (5 by default, user-tunable). The following figure gives an example of how it works. Let’s say a scaffold called `scaffold 1` in the assembly. This scaffold consists of three stretches of gaps of various lengths, 5, 3, and 9, respectively. By default, AGOUTI shreds it into three contigs, `Scaffold_1_0`, `Scaffold_1_1`, and `Scaffold_1_2`. AGOUTI does not cut at the second gap because it has a length of 3. Notably, AGOUTI uses `SEQID_INDEX` to tell the order of contigs in the given original scaffold. For scaffolds without gaps, AGOUTI does not split them.)
1.10.5 Shred Annotation

Since v0.3.0, AGOUTI is also able to shred gene annotation matching the give assembly. It compares start and end positions of features with coordinates of cut sites, and updates annotation accordingly. There are five types of features AGOUTI cares: gene, exon, CDS, five_prime_UTR, three_prime_UTR. The following figure gives an example of how it works. Let’s use the same scaffold (i.e. Scaffold_1) shredded in the picture above. Assume that there is a gene span across the second cut site, and it consists of three exons (green box). AGOUTI splits the assembly such that one gene becomes two (boxes in different colors) sitting on two different contigs. AGOUTI assigns them with different ID, in a similar fashion as names of shredded contigs, GENEID_INDEX. This naming tells 1) whether two shredded genes belong to a single one; and 2) the order.

Fig. 1.3: Alt text

1.10.6 Recover Original Paths

AGOUTI will try to recover original connections for shredded contigs to preserve contiguity as much as possible. To do so, contigs that are not scaffolded by AGOUTI are first identified. For a pair of such contigs, AGOUTI then re-connects them as long as they are next to each other in the original scaffolding path. Consider an example in which a scaffold is shredded into 5 contigs: A, B, C, D, and E, and AGOUTI is able to scaffold C and D. This leaves A, B and E untouched. Given our rules, AGOUTI will re-connect A and B without appending E to B, because B and E are not consecutive in the original path. If annotation is shredded at the same time with the assembly, AGOUTI will also merge them during the process.

1.10.7 Report Inconsistencies

Any inconsistencies with the original path can provide useful information to further improve genome assembly. AGOUTI provides alternative scaffolding paths in the form of network, named [prefix].consistency.nw. This network consists of 4 columns, from, interaction, to, and type. from and to are source and target nodes/contig. If two contigs connected from the same original scaffold, a type of agouti_same will be assigned; agouti_diff given otherwise. In the former case if the two contigs are not consecutive, AGOUTI reports all the skipped ones in between. In the latter scenario on the other hand, AGOUTI additionally gives immediate neighbors around each of the two contigs according to original paths. In the network file, these connections are typed original.

Consider an example illustrated in the figure below. AGOUTI connects three pairs of shredded contigs. For contig scaf669029_3 and scaf669029_7, they come from the same original scaffold (blue solid line), which can be tell by the string before the underscore. Because they are not consecutive (index 3 and 7), scaf669029_4, scaf669029_5, and scaf669029_6 are reported to tell the contigs being skipped (pink dotted line).

In the same example, AGOUTI also connects two contigs from different original scaffolds (red zigzag line), scaf669029_3 and scaf668522_35. AGOUTI additionally reports immediate neighbors of each of the two contigs (connected by green arrowed line). Both contigs come from the ends of their corresponding original scaffolds, and a path between the two can suggest a connection between the same two original scaffolds. Connections between two contigs from the middle of their original scaffolds, on the other hand, flag inconsistencies, e.g. scaf668522_30 and scaf669547_5.

The network file is Cytoscape-ready, and we also provide a style file consistency.xml under the cytoscape folder. The example demonstrated here is only a tiny part of the network. You can get the full network by simply load the example.consistency.nw file into Cytoscape.
1.11 Break-and-Continue

AGOUTI is built with a couple of modules. The output of current module will be taken as the input as the next module. Given the same input, modules such as extracting joining-pairs from BAM file, spits out the same intermediate results. AGOUTI therefore tries to save some running time by skipping such steps if they were finished successfully from previous runs. To use this feature, simply run AGOUTI the second time with the same output directory and output prefix as the previous run. If you desire a fresh start, simply use -overwrite to overwrite all results generated previously, or gives a new prefix.

1.12 Graph Visualization

AGOUTI makes the scaffolding graph accessible to users. Under scaffolding folder, you can find a file named after [prefix].agouti_scaffolding.graph.dot. The dot file can be directly loaded in packages like Graphviz. In the graph, contigs/vertices are in black circle, while there are two color codings for edges. Ones in red are the scaffolding path in the final assembly, and others in black are simply edges that were not traversed. Edges in dotted style represent connections with a number of supporting joining-pairs lower than the minimum specified.

1.13 Support

Please feel free to submit any issues through GitHub issue tracker.

Any comments are welcome as well!