AGORA « *How To* »

Alexandra LOUIS (alouis@biologie.ens.fr), Hugues ROEST CROLLIUS (hrc@ens.fr)

DYOGEN Laboratory, Institut de Biologie de l'Ecole normale supérieure (IBENS)
46 rue d'Ulm, 75005 Paris

1. History

AGORA stands for "Algorithm for Gene Order Reconstruction in Ancestors" and was developed by Matthieu Muffato during his PhD (2007-2010) in the DYOGEN Laboratory at the Ecole normale supérieure in Paris. Since then it has been constantly used in the group, especially to generate ancestral genomes for the Genomicus online server for comparative genomics (www.genomicus.biologie.ens.fr/genomicus). Many algorithms used in AGORA are described in details in Matthieu's thesis, available only in French (Muffato 2010): https://www.biblio.univ-evry.fr/theses/2010/2010EVRY0040.pdf and the core algorithm used to build and linearise the adjacency graph is described in a separate study (Berthelot et al 2015).

2. What AGORA does and does not do

AGORA will take as input a set of extant gene list, ordered by chromosome (or scaffolds), a species tree linking the genomes, and phylogenetic gene trees reconciled with the species tree. It will produce linear ancestral gene orders (with transcriptional orientation) at all the nodes of the species tree. This may result in very long successive ancestral adjacencies or CARs (Contiguous Ancestral Regions) if the data allows it (e.g. closely related extant

genomes with contiguous sequence assemblies) or very short ones if the data does not allow it (e.g extant genes distributed in short scaffolds, or very rearranged extant genomes).

AGORA will not:

- Reconstruct ancestral nucleotide or protein sequences.
- Reconstruct circular chromosomes

AGORA can be run in two versions. The first and simplest uses all possible adjacencies found in extant genomes to reconstruct ancestral adjacencies, eventually leading to contiguous ancestral regions. In principle this should work fine if the genomes are perfectly sequenced and annotated, but they rarely are. Also, gene duplications are difficult to resolve accurately in gene phylogenies, and AGORA is sensitive to errors in gene trees. A second, more complex version first identifies "robust" gene families, on the basis of a user-defined criterion. Typically this can be a requirement that there are as many genes on a tree as there are species, thus limiting the chances that duplications have occurred. AGORA will first build a temporary ancestral genome with these genes (ignoring all other families) as a robust backbone. Then, it will use remaining gene families to fill in the space between robust genes, but without breaking a chain of robust genes.

3. File formats

To reconstruct ancestral gene orders, AGORA needs 3 kinds of files (see example/data/):

- A species tree (e.g.: species.conf)
- A set of extant gene trees reconciled with the species tree (eg: GeneTreeForest.phylTree). Extant genes that are not in a tree will not be used for gene order reconstruction.
- The order of extant genes in each extant genomes (eg: genes/genes.M1.list.bz2)

• The species tree

The species tree must be in Phyltree format. The Phyltree format is a human readable format of trees developed specifically for AGORA, based on tabulations. An example of species tree is given in the package, see:

Species.conf -> Phyltree format
Species.nwk -> Newick format
Species.pdf (to visualize the example species tree)

To convert a tree from NHX format to Phyltree format, use the script newickSpeciesTree2phylTreeSpeciesTree.py in the src directory:

../../src/newickSpeciesTree2phylTreeSpeciesTree.py Species.nwk > Species.conf

Warning: ancestral species nodes have to be labelled with a unique ID for each node (e.g. "Amniota" or "Anc659123")!

• The forest of gene trees

The forest of gene trees has to be in Phyltree format as well, in a single file. An example is given in NHX format, one line per family gene tree:

GeneTreeForest.nhx.bz2 -> nhx format.

To convert NHX trees to Phyltree, use the nhxGeneTrees2phylTreeGeneTrees.py script in src directory:

```
../../src/nhxGeneTrees2phylTreeGeneTrees.py Family1.nhx > Family1.phyltree
```

- The genes files.

The genes files used by AGORA contain the list of genes on each extant genome. The format is tab-separated values, in 5 mandatory columns (a 6th is optional). One file must be provided per extant genome.

The fields are:
1. Name of the chromosome (text).
2. Start position of the gene (integer).
3. End position of the gene (integer).
4. Gene orientation (1 or -1)
5. Gene name (text)
6. Transcript name (optional)

The names of the files have to be consistent with the names of the species in the species tree and the gene names have to be those used in the gene trees, in the format prefix.species_name.suffix.

For example, if the species in the species tree are: HUMAN, MOUSE, DOG, genes files have to be named:
prefix.HUMAN.suffix (eg:genes.HUMAN.list)
prefix.MOUSE.suffix
prefix.DOG.suffix

If the species are Homo.sapiens, Mus.musculus and Canis.familiaris, genes files have to be named:

prefix.Homo.sapiens.suffix (eg: genes.Homo.sapiens.list)
prefix.Mus.musculus.suffix
prefix.Canis.familiaris.suffix

(see Species.conf, GeneTreeForest.phyltree and the 5 files of genes)

AGORA can use compressed (bz2, gz, az) or uncompressed files.

4. Running AGORA

4.1 Extraction of ancestral gene content

The first step in AGORA is to identify all ancestral genes for all ancestral genomes, and print them in one file per target ancestral genome.

Run the script ALL.extractGeneFamilies.py in the *src* directory. It takes as input the species tree, the forest of gene trees and a template to name the output files.

```
src/ALL.extractGeneFamilies.py
 example/data/Species.conf
```

```
example/data/GeneTreeForest.phylTree.bz2
-OUT.ancGenesFiles=example/results/ancGenes/all/ancGenes.%s.list.bz2
>
example/results/GeneTreeForests.withAncGenes.phyltree
2>
example/results/log.ancGenes
```

This will create a subdirectory *ancGenes* in the *results* directory. Be careful to provide the correct path to write the ancGene files (ancGenes/all/ancGenes.%s.list.bz2), it will be important if you use AGORA on "robust" family in a second step (see article). The “%s” will be automatically replaced by the extant and ancestral species name, as indicated in the species tree.

ancGenes files are tab separated files, with the following two fields:
1. Ancestral gene names
2. A space separated list of extant copies of this ancestral gene, in the genome of extant species.

On the standard output, the script produces the forest of gene trees, rewritten with the ancestral gene names at each node:

`example/results/GeneTreeForests.withAncGenes.phyltree`

4.2 Agora with no selection of robust families

This is the simplest way of running AGORA. It will generate an adjacency graph for a given ancestral genome using all available conserved adjacencies, then linearise it to produce CARs. Run the scripts step-by-step or use the encapsulated scripts (agora.py) with the configuration file (agora.ini). In the following command lines, “A0” refers to the name of the oldest ancestor (named in the species tree) to be reconstructed. All the descendants of A0 will be reconstructed as well.

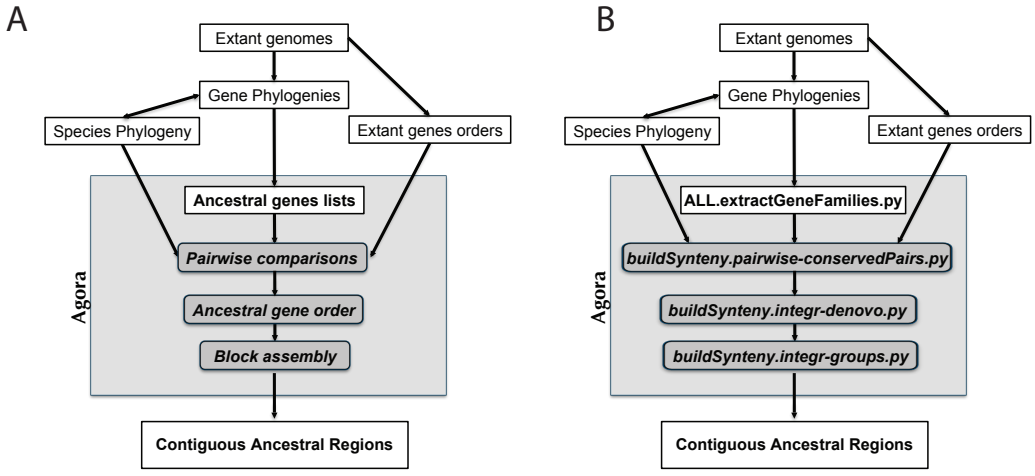

Figure 1. Schematic of the AGORA pipeline with no selection of robust families. (A) Diagram with the name of the different AGORA procedures. From the data (white boxes at the top), AGORA (grey rectangle) will first extract ancestral gene names, compare all extant genomes pairwise, build the adjacency graph and linearise it, and assemble the resulting contigs into Contiguous Ancestral Regions (CARs). (B) The same diagram with the names of the scripts required for each procedure, as described in this document.

- Step-by-step
  - Pairwise comparison

This step will compare extant genomes in all possible pairwise combinations to identify conserved adjacencies.
Run buildSynteny.pairwise-conservedPairs.py

```
mkdir example/results/diags/pairwise/pairs-all/

./src/buildSynteny.pairwise-conservedPairs.py
      ./example/data/Species.conf
      A0
      -ancGenesFiles=example/results/ancGenes/all/ancGenes.%s.list.bz2
      -genesFiles=./example/data/genes/genes.%s.list.bz2
      -OUT.pairwise=example/results/diags/pairwise/pairs-all/%s.list.bz2
      >
      example/results/abc
      2>
      example/results/diags/pairwise/pairs-all/pairs.log
```

  - Order of ancestral genes

```
mkdir -p example/results/diags/integr/denovo-all/

src/buildSynteny.integr-denovo.py
      example/data/Species.conf
      A0
      -OUT.ancDiags=example/results/diags/integr/denovo-
      all/anc/diags.%s.list.bz2
      example/results/diags/pairwise/pairs-all/%s.list.bz2
      -ancGenesFiles=example/results/ancGenes/all/ancGenes.%s.list.bz2
      +searchLoops
      >
      example/results/diags/integr/denovo-all/graph.txt.bz2
      2>
      example/results/diags/integr/denovo-all/log
```

  - Blocks assembly

```
mkdir -p example/results/diags/integr/denovo-all.groups/

src/buildSynteny.integr-groups.py example/data/Species.conf
A0 A0
-OUT.ancDiags=example/results/diags/integr/denovo-
all.groups/anc/diags.%s.list.bz2
-IN.ancDiags=example/results/diags/integr/denovo-all/anc/diags.%s.list.bz2
-ancGenesFiles=example/results/ancGenes/all/ancGenes.%s.list.bz2
-genesFiles=example/data/genes/genes.%s.list.bz2
>
example/results/diags/integr/denovo-all.groups/graph.txt.bz2
2>
example/results/diags/integr/denovo-all.groups/log
```

- All in one: Agora.py

This is a script that encapsulates the 3 previous scripts and runs them automatically. Use the *agora.ini* configuration file to set paths to scripts and parameters.

eg: see agora.ini

Just run:

```
src/agora.py agora.ini -workingDir=example/results
```

4.3 Agora with selection of robust families

This approach builds an adjacency graph using a subset of extant conserved adjacencies, which are selected by a user-defined quality criterion. The idea here is to build robust ancestral adjacency scaffolds, and to insert within these adjacencies the remaining ancestral genes.

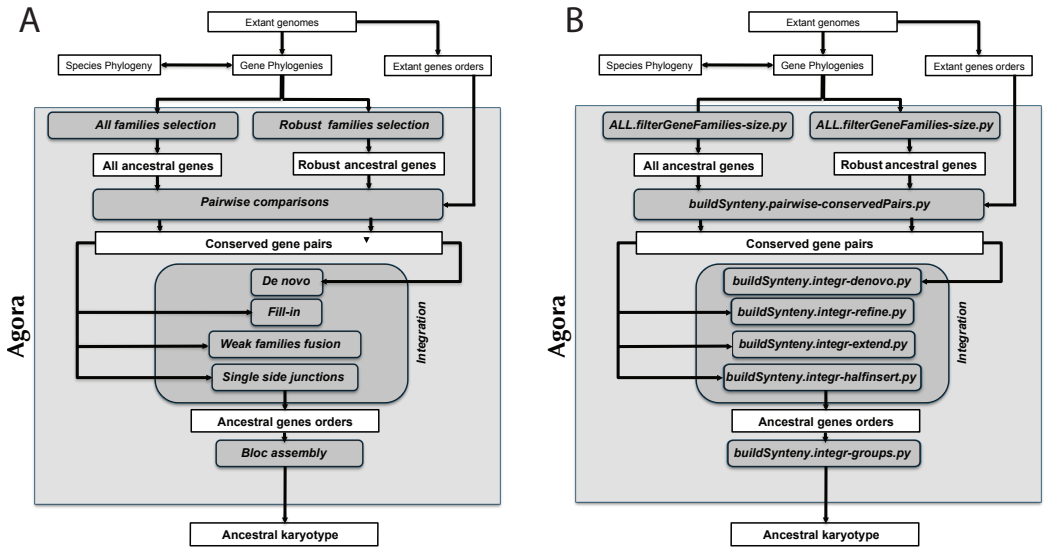

Figure 2. Schematic of the AGORA pipeline with selection of robust families. (A) Diagram with the name of the different AGORA procedures. From the data (white boxes at the top), AGORA (grey rectangle) will identify a subset of robust gene phylogenies according to a user-defined criteria, then extract all ancestral genes (all genes and robust genes). It will compare all extant genomes pairwise (considering all genes and robust genes separately), build the adjacency graphs (de novo) and linearise them to obtain robust contigs. It will then fill in the robust contigs with non-robust genes (Fill-in), build contigs of non-robust genes (weak families fusion) and insert these in the filled-in robust contigs (Single side junction). Finally it will assemble the resulting contigs (Bloc assembly) into Contiguous Ancestral Regions (CARs). (B) The same diagram with the names of the scripts required for each procedure, as described in this document.

- Step by step
  - Selection of robust genes

```
./src/ALL.filterGeneFamilies-size.py
      ./example/data/Species.conf
      A0
      -IN.ancGenesFiles=./example/results/ancGenes/all/ancGenes.%s.list.bz2
      -OUT.ancGenesFiles=./example/results/ancGenes/size-%s-
         %s/ancGenes.%s.list.bz2
```

```
      1.0,0.9,0.77
      1.0,1.1,1.33
      >
      ./example/results/ancGenes/size.txt.bz2
      2>
      ./example/results/ancGenes/size.log
```

- Pairwise comparison

This step is run once for all ancestral genes, and once per set of criteria for selecting robust families. Here, the example is given for the criteria 1.0-1.0, meaning that robust families will have exactly the same ratio between extant genes and extant species. A criteria of 0.9-1.1 will tolerate a 10% deviation from the previous ratio. In the example, robust families are built once. But users may want to build some ancestors with one condition (e.g. 1.0-1.0) and other ancestors with another condition (e.g. 0.9-1.1). See for example the agora-size.ini configuration file (section 2 and section 4) that is used when running the wrap up script agora.py. This step should therefore be run once for each set of desired criteria. AGORA will later pick what is required from each run.

For all ancestral genes:

```
./src/buildSynteny.pairwise-conservedPairs.py
      ./example/data/Species.conf
      A0
      -ancGenesFiles=example/results/ancGenes/all/ancGenes.%s.list.bz2
      -genesFiles=./example/data/genes/genes.%s.list.bz2
      -OUT.pairwise=example/results/diags/pairwise/pairs-all/%s.list.bz2
      2>
      example/results/diags/pairwise/pairs-all/pairs.log
```

For robust gene families (once per set of criteria):

```
./src/buildSynteny.pairwise-conservedPairs.py
      ./example/data/Species.conf
      A0
      -ancGenesFiles=example/results/ancGenes/size-1.0-
      1.0/ancGenes.%s.list.bz2
      -genesFiles=./example/data/genes/genes.%s.list.bz2
      -OUT.pairwise=example/results/diags/pairwise/pairs-size-1.0-
      1.0/%s.list.bz2
      >
      example/results/abc
      2>
      example/results/diags/pairwise/pairs-size-1.0-1.0/pairs.log
```

- Adjacency graphs for robust gene families

```
./src/buildSynteny.integr-denovo.py
      ./example/data/Species.conf
      A0
      -OUT.ancDiags=example/results/diags/integr/denovo-size-1.0-
         1.0/anc/diags.%s.list.bz2
      example/results/diags/pairwise/pairs-size-1.0-1.0/%s.list.bz2
      -ancGenesFiles=example/results/ancGenes/all/ancGenes.%s.list.bz2
      >
      example/results/diags/integr/denovo-size-1.0-1.0/graph.txt.bz2
      2>
      example/results/diags/integr/denovo-size-1.0-1.0/log
```

- Copy previous ancestral scaffolds based on robust genes in one single directory

```
./src/buildSynteny.integr-copy.py
      ./example/data/Species.conf
      A0
      -OUT.ancDiags=example/results/diags/integr/denovo-size-
         custom/anc/diags.%s.list.bz2
      -IN.ancDiags=example/results/diags/integr/denovo-size-1.0-
          1.0/anc/diags.%s.list.bz2
      >
      example/results/diags/integr/denovo-size-custom/graph.txt.bz2
      2>
      example/results/diags/integr/denovo-size-custom/log
```

- Refine

```
./src/buildSynteny.integr-refine.py
      ./example/data/Species.conf
      A0
      -func=0,32|100,40t|10000
      -timeout=150
      -OUT.ancDiags=example/results/diags/integr/denovo-size-custom.refine-
         all/anc/diags.%s.list.bz2
      example/results/diags/pairwise/pairs-all/%s.list.bz2
      -IN.ancDiags=example/results/diags/integr/denovo-size-
         custom/anc/diags.%s.list.bz2
      >
      example/results/diags/integr/denovo-size-custom.refine-
         all/graph.txt.bz2
      2>
      example/results/diags/integr/denovo-size-custom.refine-all/log
```

- Extend

```
./src/buildSynteny.integr-extend.py
      ./example/data/Species.conf
      A0
      +onlySingletons
      -OUT.ancDiags=example/results/diags/integr/denovo-size-custom.refine-
         all.extend-all/anc/diags.%s.list.bz2
      example/results/diags/pairwise/pairs-all/%s.list.bz2
      -IN.ancDiags=example/results/diags/integr/denovo-size-custom.refine-
          all/anc/diags.%s.list.bz2
      >
      example/results/diags/integr/denovo-size-custom.refine-all.extend-
         all/graph.txt.bz2
      2>
      example/results/diags/integr/denovo-size-custom.refine-all.extend-
         all/log
```

- Half-insert

```
./src/buildSynteny.integr-halfinsert.py
      ./example/data/Species.conf
      A0
      -OUT.ancDiags=example/results/diags/integr/denovo-size-custom.refine-
   all.extend-all.halfinsert-all/anc/diags.%s.list.bz2
   example/results/diags/pairwise/pairs-all/%s.list.bz2
   -IN.ancDiags=example/results/diags/integr/denovo-size-custom.refine-
```

```
      all.extend-all/anc/diags.%s.list.bz2
   -REF.ancDiags=example/results/diags/integr/denovo-size-custom.refine-
      all/anc/diags.%s.list.bz2
   >
   example/results/diags/integr/denovo-size-custom.refine-all.extend-
      all.halfinsert-all/graph.txt.bz2
   2>
   example/results/diags/integr/denovo-size-custom.refine-all.extend-
      all.halfinsert-all/log
```

- Scaffolding

```
./src/buildSynteny.integr-groups.py
      ./example/data/Species.conf
      A0 A0
      -OUT.ancDiags=example/results/diags/integr/denovo-size-custom.refine-
         all.extend-all.halfinsert-all.groups/anc/diags.%s.list.bz2
-IN.ancDiags=example/results/diags/integr/denovo-size-custom.refine-
all.extend-all.halfinsert-all/anc/diags.%s.list.bz2 -
ancGenesFiles=example/results/ancGenes/all/ancGenes.%s.list.bz2 -
genesFiles=./example/data/genes/genes.%s.list.bz2 >
example/results/diags/integr/denovo-size-custom.refine-all.extend-
all.halfinsert-all.groups/graph.txt.bz2 2>
example/results/diags/integr/denovo-size-custom.refine-all.extend-
all.halfinsert-all.groups/log
```

- Copy of the results in the final repository

```
      ./src/buildSynteny.integr-copy.py ./data/Species.conf A0 -
OUT.ancDiags=example/results/diags/integr/final/anc/diags.%s.list.bz2 -
IN.ancDiags=example/results/diags/integr/denovo-size-custom.refine-
all.extend-all.halfinsert-all.groups/anc/diags.%s.list.bz2 >
example/results/diags/integr/final/graph.txt.bz2 2>
example/results/diags/integr/final/log
```

- All in one: Agora.py

This script will run all the above steps automatically. Use the *agora-size.ini* configuration file to set paths to scripts and to set parameters.

Run:

```
./src/agora.py conf/agora-size.ini
      -workingDir=example/results/
      >
      agora-size.log &
```

5. Output format and post-processing scripts

- The diags files

```
example/results/diags/integr/final/anc/diags.*.list.bz2
```

This directory contains a file for each ancestral reconstructed genome (eg: diags.A0.list.bz2). There are five tab-separated fields, and values in each field are further separated by single spaces. The term ‘diag’ historically refers to the diagonal lines that appear in 2 dimensional matrices comparing 2 genomes and reflecting successive conserved adjacencies.

The fields are:
1. Name of the ancestral species.
2. Number of genes in the ancestral block.
3. List of gene IDs. Each ID corresponds to the line number in the corresponding ancGenes.Species.list (starting from 0).
4. Gene transcriptional orientation (strand) within the block.
5. A relative confidence index for each inter-block linkage. The values in parenthesis are the size of the initial blocks.

The values without parenthesis represent the number of time the two adjacent blocks are adjacent in extant species.
The sum of the lengths of the initial blocks (numbers in parenthesis) is thus equal to the size of the whole block (field number 2)

ex: A block of 8 genes in A0 made of 2 sub-blocks linked by an adjacency of score 6.

```
A0   8    4559 4179 10099 15638 1304 10998 5675 13765     -1 -1 -1 1 1 -1 -1 1    (5) 6 (3)
```

- Post processing scripts: transforming diags files to genome files

```
./src/postprocessing/misc.convertContigsToGenome.py
example/results/diags/integr/final/anc/diags.A0.list.bz2
tmp/ancGenes/all/ancGenes.A0.list.bz2 > tmp/ancGenomes/ancGenome.A0.list
```

As an example:

```
tmp/ancGenomes/ancGenomes.*.list
```

This directory contains a file for each ancestral reconstructed genome (eg: ancGenome.A0.list).

There are five tab-separated fields, and values in each field are further separated by single spaces.

The fields are:
1. Name of the ancestral block.
2. Relative start position of the ancestral gene.
3. Relative end position of the ancestral gene.
4. Ancestral gene orientation within the block.
5. Ancestral gene names. The first name corresponds to the ancestral gene, subsequent ones are the list of extant copies of this ancestral gene, in the genome of extant species.