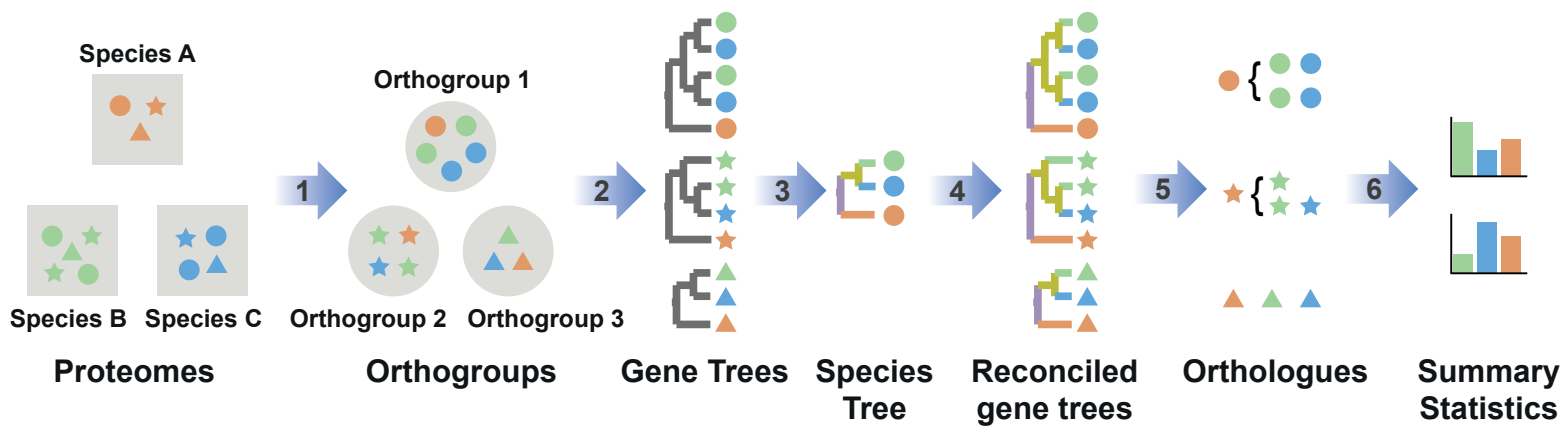


# OrthoFinder Manual:

## Accurate inference of orthologues and orthogroups made easy!



*Dr. David Emms*  
*david.emms@plants.ox.ac.uk*

*Dr. Steven Kelly*  
*steven.kelly@plants.ox.ac.uk*

October 3, 2016

# What does OrthoFinder do?

OrthoFinder is a fast, accurate and comprehensive analysis tool for comparative genomics. It finds **orthologues** and **orthogroups**, infers **gene trees** for all orthogroups and infers a **rooted species tree** for the species being analysed. OrthoFinder also provides **comprehensive statistics** for comparative genomic analyses. OrthoFinder is simple to use and all you need to run it is a set of protein sequence files (one per species) in FASTA format.

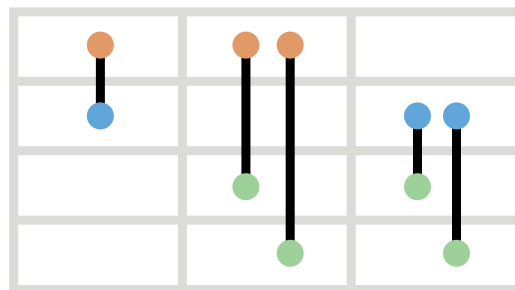
## Orthogroup



**Group of genes** descended from single gene in LCA of group of species

## Orthologues

Hu-Mo Hu-Ch Mo-Ch



**Pairs of genes** descended from single gene in LCA of pair of species

## Citation

Emms, D.M. and Kelly, S. (2015) OrthoFinder: solving fundamental biases in whole genome comparisons dramatically improves orthogroup inference accuracy, *Genome Biology* 16:157

## Links

<https://genomebiology.biomedcentral.com/articles/10.1186/s13059-015-0721-2>

<https://github.com/davidemms/OrthoFinder>

# Contents

<b>1</b>	<b>Orthogroups, Orthologues &amp; Paralogues</b>	<b>3</b>
1.1	Why Orthogroups . . . . .	3
<b>2</b>	<b>Setting Up OrthoFinder</b>	<b>4</b>
2.1	Set Up . . . . .	4
2.2	Running OrthoFinder . . . . .	4
2.3	Dependencies . . . . .	4
2.3.1	BLAST+ . . . . .	4
2.3.2	MCL . . . . .	5
2.3.3	FastME . . . . .	5
2.3.4	DLCpar . . . . .	5
2.4	Setup for advanced use . . . . .	5
2.4.1	Trees from MSA . . . . .	5
2.4.2	Python Source Code Version . . . . .	5
<b>3</b>	<b>Performing a Complete OrthoFinder Analysis</b>	<b>6</b>
<b>4</b>	<b>Results Files</b>	<b>7</b>
4.1	Results Files: Orthogroups . . . . .	7
4.2	Results Files: Orthogroup Statistics . . . . .	7
4.3	Results Files: Orthologues . . . . .	7
4.4	Results Files: Gene Trees and Species Tree . . . . .	7
<b>5</b>	<b>Advanced Usage</b>	<b>8</b>
5.1	Adding Extra Species . . . . .	8
5.2	Removing Species . . . . .	8
5.3	Adding and Removing Species Simultaneously . . . . .	8
5.4	Inferring MSA Gene Trees (to be replaced) . . . . .	8
5.5	Parallelising OrthoFinder Algorithm (-a option) . . . . .	8
5.6	Running BLAST Searches Separately . . . . .	9
5.7	Using Pre-Computed BLAST Results . . . . .	9
5.8	Using the Orthoxml Format . . . . .	9
5.9	Regression Tests . . . . .	10
<b>6</b>	<b>Appendix: File Format for Pre-Computed BLAST Results</b>	<b>11</b>
6.1	FASTA Files . . . . .	11
6.2	BLAST Results Files . . . . .	11
6.3	SequenceIDs.txt . . . . .	12
6.4	SpeciesID.txt . . . . .	12

# 1 Orthogroups, Orthologues & Paralogues

‘Orthologue’ is a term that applies to genes from two species. Orthologues are pairs of genes that descended from a single gene in the last common ancestor (LCA) of two species (Figure 1A & B). An orthogroup is the natural extension of the concept of orthology to groups of species. An orthogroup is the group of genes descended from a single gene in the LCA of a group of species (Figure 1A). When looking at the gene tree, the first divergence between the genes in an orthogroup is a speciation event and the same is true for orthologues.

As a result of gene duplication events, it is possible to have multiple genes from the same species with both orthologues and orthogroups. In the example (Figure 1A & B), the human gene HuA has two genes that are orthologues of it in chicken, ChA1 and ChA2. Looking again at the orthogroup, we see that there are two chicken genes (Figure 1A) but only one gene from mouse and human. Some authors refer to the genes ChA1 and ChA2 as co-orthologues of HuA to emphasise the fact that there are multiple orthologues. These genes are nevertheless still orthologues and so we will usually just use this broader term. In fact, gene duplication events are so common that in addition to the one-to-many relationship implied by the term ‘co-orthologues’, there are frequently many-to-many relationships between orthologues. All of these relationships are identified by an OrthoFinder analysis.

Gene duplication events give rise to paralogues. Paralogues are pairs of genes that diverged from a single gene at a gene duplication event. The two chicken genes ChA1 and ChA2 are paralogues (Figure 1A & C). Two genes from different species can also be paralogues if they diverged from one another at a gene duplication event, although there are no examples of this in Figure 1. Since all branching events in a gene tree are either speciation events (that give rise to orthologues) or duplication events (that give rise to paralogues), any genes in the same orthogroup that are not orthologues must necessarily be paralogues.

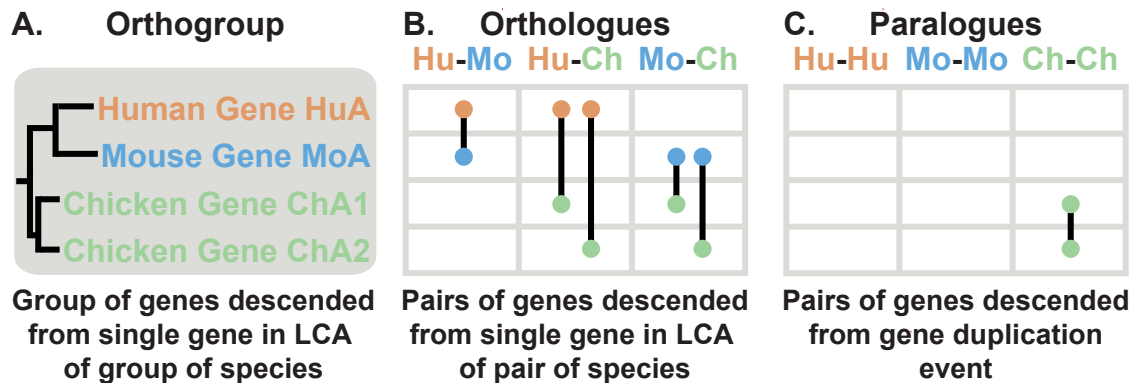


Figure 1: A hypothetical human, mouse and chicken orthogroup.

## 1.1 Why Orthogroups

If you followed the explanations above it will be clear that an orthogroup is just a gene family/clade of genes defined at a specific taxonomic level—namely, those genes descended from a single gene at the time of the LCA. Some may regard this definition of an orthogroup as unsatisfactory since an orthogroup can contain genes that are paralogues of one another (ChA1 is a paralogue of ChA2 in Figure 1). However, this definition of an orthogroup is the only logically consistent way of extending the concept of orthology to multiple species. If there have been gene duplication events it is not possible to create a group of genes containing all orthologues and only orthologues—try it with the example above!

One can still identify orthologues between the genes in each pair of species though, but the orthogroup is the correct unit of comparison when considering the group of species as a whole. In fact, one use for orthogroups is for identifying orthologues: The canonical way to identify orthologues is using a gene tree, and an orthogroup is exactly the set of genes that need to be in the gene tree in order to identify all orthologues. This is the method used by OrthoFinder.

## 2 Setting Up OrthoFinder

OrthoFinder runs on Linux and Mac, setup instructions are given below.

### 2.1 Set Up

1. Download the latest release from github: <https://github.com/davidemms/OrthoFinder/releases> (for this example we will assume it is OrthoFinder-1.0.6.tar.gz, change this as appropriate.)
2. In a terminal, `cd` to where you downloaded the package
3. Extract the files: `tar xzf OrthoFinder-1.0.6.tar.gz`
4. Test you can run OrthoFinder: `OrthoFinder-1.0.6/orthofinder -h`. OrthoFinder should print its ‘help’ text.

To perform an analysis OrthoFinder requires some dependencies to be installed and in the system path (only the first two are needed to infer orthogroups and all four are needed to infer orthologues and gene trees as well):

1. BLAST+
2. The MCL graph clustering algorithm
3. FastME (The appropriate version for your system, e.g. ‘fastme-2.1.5-linux64’, should be renamed ‘fastme’, see instructions below.)
4. DLCpar

Brief instructions are given below although users can refer to the installation notes provided with these packages for more detailed instructions.

### 2.2 Running OrthoFinder

Once the required dependencies have been installed, try running OrthoFinder on the example data:

1. `OrthoFinder-1.0.6/orthofinder -f ExampleDataset`

Assuming everything was successful OrthoFinder will end by printing the location of the results files, a short paragraph providing a statistical summary and the OrthoFinder citation. If you make use of OrthoFinder for any of your work then please cite it as this helps support future development.

If you have problems with this standalone binary version of OrthoFinder you can use the python source code version, which has a name of the form, ‘OrthoFinder-1.0.6\_source.tar.gz’ and is available from the github ‘releases tab’. See Section 2.4.2.

### 2.3 Dependencies

Each of the following packages provide their own detailed instructions for installation, here we give a concise guide.

#### 2.3.1 BLAST+

NCBI BLAST+ is available in the repositories from most Linux distributions and so can be installed in the same way as any other package. For example, on Ubuntu, Debian, Linux Mint:

- `sudo apt-get install ncbi-blast+`

Alternatively, instructions are provided for installing BLAST+ on Mac and various flavours of Linux on the “Standalone BLAST Setup for Unix” page of the BLAST+ Help manual currently at <http://www.ncbi.nlm.nih.gov/books/NBK1762/>. Follow the instructions under “Configuration” in the BLAST+ help manual to add BLAST+ to the PATH environment variable.

### 2.3.2 MCL

The mcl clustering algorithm is available in the repositories of some Linux distributions and so can be installed in the same way as any other package. For example, on Ubuntu, Debian, Linux Mint:

- `sudo apt-get install mcl`

Alternatively it can be built from source which will likely require the ‘build-essential’ or equivalent package on the Linux distribution being used. Instructions are provided on the MCL webpage, <http://micans.org/mcl/>.

### 2.3.3 FastME

FastME can be obtained from <http://www.atgc-montpellier.fr/fastme/binaries.php>. The package contains a ‘binaries/’ directory. Choose the appropriate one for your system and copy it to somewhere in the system path e.g. ‘/usr/local/bin’ and name it ‘fastme’. I.e.:

- `sudo cp fastme-2.1.5-linux64 /usr/local/bin/fastme`

### 2.3.4 DLCpar

DLCpar can be downloaded from <http://compbio.mit.edu/dlcpar/> and installed as for a standard python package:

1. Download the latest version
2. Extract the package: `tar xzf dlcpar-1.0.tar.gz`
3. `cd dlcpar-1.0/`
4. `sudo python setup.py install`

## 2.4 Setup for advanced use

The following steps are not required for the standard OrthoFinder use cases and are only needed if you want to run the ‘trees\_from\_MSA’ utility or you want to run OrthoFinder using the python source code version.

### 2.4.1 Trees from MSA

To use the trees.from.MSA utility there are two additional dependencies which should be installed and in the system path:

1. MAFFT
2. FastTree

### 2.4.2 Python Source Code Version

It is recommended that you use the standalone binaries for OrthoFinder which do not require python or scipy to be installed. However, the python source code version is available from the github ‘releases’ page (e.g. ‘OrthoFinder-1.0.6\_source.tar.gz’) and requires python 2.7 and scipy to be installed. Up-to-date and clear instructions are provided here: <http://www.scipy.org/install.html>, be sure to chose a version using python 2.7. As websites can change, an alternative is to search online for “install scipy”.

### 3 Performing a Complete OrthoFinder Analysis

Performing a complete OrthoFinder analysis is simple:

1. Download the amino acid sequences, in FASTA format, for the species you want to analyse. If you have the option, it is best to use a version containing a single representative/longest transcript-variant for each gene.
2. Optionally, you may want to rename the files to something simple since the filenames will be used as species identifiers in the results. E.g if you were using the ‘Homo\_sapiens.GRCh38.pep.all.fa’ file you could rename it to ‘Homo\_sapiens.fa’ or ‘Human.fa’.
3. Place the FASTA files all in a single directory.
4. To perform a complete OrthoFinder analysis requires just one command:  
`orthofinder -f fasta_files_directory [-t number_of_threads]`

The argument ‘`number_of_threads`’ is an optional argument to specify the number of parallel threads to use for the BLAST searches, tree inference and reconciliation. As the BLAST queries can be a time-consuming step it is best to use at least as many BLAST processes as there are CPUs on the machine.

The OrthoFinder run will finish by printing the location of the results files, a short paragraph providing a descriptive statistical summary and the OrthoFinder citation. If you make use of OrthoFinder for any of your work then please cite it as this helps justify OrthoFinder support and future development. The OrthoFinder results files are described in Section 4.

## 4 Results Files

A standard OrthoFinder run produces a set of files describing the orthogroups, orthologues and gene trees for the set of species being analysed. Their locations are given at the end of an OrthoFinder run.

### 4.1 Results Files: Orthogroups

OrthoFinder generates the main orthogroup file, **Orthogroups.csv**, and two supporting files:

1. **Orthogroups.csv** is a tab separated text file. Each row contains the genes belonging to a single orthogroup. The genes from each orthogroup are organized into columns, one per species.
2. **Orthogroups\_UnassignedGenes.csv** is a tab separated text file that is identical in format to Orthogroups.csv but contains all of the genes that were not assigned to any orthogroup.
3. **Orthogroups.txt** (legacy format) is a second file containing the orthogroups described in the Orthogroups.csv file but using the OrthoMCL output format.

### 4.2 Results Files: Orthogroup Statistics

The statistics calculated from the orthogroup analysis provide the basis for any comparative genomics analysis. They are easily plotted and can also be used for quality control.

1. **Statistics\_Overall.csv** is a tab separated text file giving useful statistics from the orthogroup analysis.
2. **Statistics\_PerSpecies.csv** is a tab separated text file giving many of the same statistics as the 'Statistics\_Overall.csv' file but on a species-by-species basis.
3. **Orthogroups\_SpeciesOverlaps.csv** is a tab separated text file containing a matrix of the number of orthogroups shared by each species-pair (i.e. the number of orthogroups which contain at least one gene from each of the species-pairs).

Most of the terms in the files **Statistics\_Overall.csv** and **Statistics\_PerSpecies.csv** are self-explanatory, the remainder are defined below.

- Species-specific orthogroup: An orthogroups that consist entirely of genes from one species.
- G50: The number of genes in the orthogroup such that 50% of genes are in orthogroups of that size or larger.
- O50: The smallest number of orthogroups such that 50% of genes are in orthogroups of that size or larger.
- Single-copy orthogroup: An orthogroup with exactly one gene (and no more) from each species. These orthogroups are ideal for inferring a species tree and many other analyses.
- Unassigned gene: A gene that has not been put into an orthogroup with any other genes.

### 4.3 Results Files: Orthologues

The orthologues spreadsheets are contained in sub-directories, one per species. Within these directories is one spreadsheet per species-pair giving all the inferred orthologues between those two species. The spreadsheets contain one column for the genes from one species and one column for genes from the other species. Orthologues can be one-to-one, one-to-many or many-to-many depending on the gene duplication events since the orthologues diverged (see Section 1 for more details). Each set of orthologues is cross-referenced to the orthogroup that contains them.

### 4.4 Results Files: Gene Trees and Species Tree

The gene trees for each orthogroup and the rooted species tree are in newick format and can be viewed using programs such as Dendroscope (<http://dendroscope.org/>) or FigTree (<http://tree.bio.ed.ac.uk/software/figtree/>).



## 5 Advanced Usage

OrthoFinder provides a number of options to allow you to incrementally add and remove species.

### 5.1 Adding Extra Species

OrthoFinder allows you to add extra species without re-running the previously computed BLAST searches:

- `orthofinder -b previous_orthofinder_directory -f new_fasta_directory`

This will add each species from the `new_fasta_directory` to existing set of species, reuse all the previous BLAST results, perform only the new BLAST searches required for the new species and recalculate the orthogroups. The `previous_orthofinder_directory` is the OrthoFinder ‘WorkingDirectory/’ containing the file ‘SpeciesIDs.txt’.

### 5.2 Removing Species

OrthoFinder allows you to remove species from a previous analysis. In the ‘WorkingDirectory/’ from a previous analysis there is a file called ‘SpeciesIDs.txt’. Comment out any species to be removed from the analysis using a ‘#’ character and then run OrthoFinder using:

- `orthofinder -b previous_orthofinder_directory`

where `previous_orthofinder_directory` is the OrthoFinder ‘WorkingDirectory/’ containing the file ‘SpeciesIDs.txt’.

### 5.3 Adding and Removing Species Simultaneously

The previous two options can be combined, comment out the species to be removed as described above and use the command:

- `orthofinder -b previous_orthofinder_directory -f new_fasta_directory`

### 5.4 Inferring MSA Gene Trees (to be replaced)

**This functionality is to be incorporated into the main orthofinder program, replacing the `trees_from_MSA` utility.**

The ‘trees\_from\_MSA’ utility will automatically generate multiple sequence alignments and gene trees for each orthogroup generated by OrthoFinder. For example, once OrthoFinder has been run on the example dataset, `trees_from_MSA` can be run using:

- `trees_from_MSA orthofinder_results_directory [-t number_of_threads]`

This will use MAFFT to generate the multiple sequence alignments and FastTree to generate the gene trees. Both of these programs need to be installed and in the system path.

### 5.5 Parallelising OrthoFinder Algorithm (-a option)

There are two separate options for controlling the parallelisation of OrthoFinder. The ‘-t’ option should always be used whereas RAM requirements may affect whether you use the ‘-a’ option or not.

1. ‘-t `number_of_threads`’: This option should always be used. It makes the BLAST searches, the tree inference and gene-tree reconciliation run in parallel. These are all highly-parallelisable and the BLAST searches in particular are by far the most time-consuming task. You should use as many threads as there are cores available.

2. ‘**-a number\_of\_orthofinder\_threads**’ The remainder of the algorithm, beyond these highly-parallelisable tasks, is relatively fast and efficient and so this option has less overall effect. It is most useful when running OrthoFinder using pre-calculated BLAST results since the time savings will be more noticeable in this case. Using this option will also increase the RAM requirements (see below).

RAM availability is an important consideration when using the ‘**-a**’ option. Each thread loads all BLAST hits between one species and all sequences in all other species. To give some very approximate numbers, each thread might require:

- 0.02 GB per species for small genomes (e.g. bacteria)
- 0.04 GB per species for larger genomes (e.g. vertebrates)
- 0.2 GB per species for even larger genomes (e.g. plants)

I.e. running an analysis on 10 vertebrate species with 5 threads for the OrthoFinder algorithm (**-a 5**) might require  $10 \times 0.04 = 0.4$  GB per thread and so  $5 \times 0.4 = 2$  GB of RAM in total. If you have the BLAST results already then the total size of all the **Blast\*\_0.txt** files gives a good approximation of the memory requirements per thread. Additionally, the speed at which files can be read is likely to be the limiting factor when using more than 5-10 threads on current architectures so you may not see any increases in speed beyond this.

## 5.6 Running BLAST Searches Separately

The ‘**-p**’ option will prepare the files in the format required by OrthoFinder and print the set of BLAST commands that need to be run.

- **orthofinder -f fasta\_files\_directory -p**

This is useful if you want to manage the BLAST searches yourself. For example, you may want to distribute them across multiple machines. Once the BLAST searches have been completed the orthogroups can be calculated using the ‘**-b**’ command as described in Section 5.7.

## 5.7 Using Pre-Computed BLAST Results

It is possible to run OrthoFinder with pre-computed BLAST results provided they are in the correct format. They can be prepared in the correct format using the ‘**-p**’ command and, equally, the files from a previous OrthoFinder run are also in the correct format to rerun using the ‘**-b**’ option. The command is simply:

- **orthofinder -b directory\_with\_processed\_fasta\_and\_blast\_results**

If you are running the BLAST searches yourself it is strongly recommended that you use the ‘**-p**’ option to prepare the files first (see Section 5.6). Should you need to prepare them manually, the required files and their formats are described in the appendix (for example, if you already have BLAST search results from another source and it will take too much computing time to redo them).

## 5.8 Using the Orthoxml Format

Orthogroups can be output in XML using the (bulky) orthoxml format. This is requested by adding ‘**-x speciesInfoFilename**’ to the command used to call orthofinder, where speciesInfoFilename should be the filename (including the path if necessary) of a user-prepared file providing the information about the species that is required by the orthoxml format. This file should contain one line per species and each line should contain the following 5 fields separated by tabs:

1. **FASTA filename**: the filename (without path) of the FASTA file for the species described on this line
2. **species name**: the name of the species

3. **NCBI Taxon ID:** the NCBI taxon ID for the species
4. **source database name:** the name of the database from which the FASTA file was obtained (e.g. Ensembl)
5. **database FASTA filename:** the name given to the FASTA file by the database (e.g. Homo\_sapiens.NCBI36.52.pep.all.fa)

As an example, a single line of the file could look like this (where each field has been separated by a tab rather than just spaces):

```
HomSap.fa Homo sapiens 36 Ensembl Homo_sapiens.NCBI36.52.pep.all.fa
```

Information on the orthoxml format can be found here: [http://orthoxml.org/0.3/orthoxml\\_doc\\_v0.3.html](http://orthoxml.org/0.3/orthoxml_doc_v0.3.html)

## 5.9 Regression Tests

A set of regression tests are included in the directory ‘**Tests**’ available from the github repository. They can be run by calling the script ‘**test\_orthofinder.py**’. They currently require version 2.2.28 of NCBI BLAST and the script will exit with an error message if this is not the case.

## 6 Appendix: File Format for Pre-Computed BLAST Results

If you want to run the BLAST searches outside of OrthoFinder and you have not already computed them then by far the best option is to use the ‘prepare’ option, ‘-p’. This will prepare the files and you can then run the BLAST searches in whatever way you wish and then run OrthoFinder on them using the ‘-b’ option. If you already have a set of BLAST search results that you want to convert into the format that OrthoFinder uses then the details are given below.

The files that must be in `directory_with_processed_fasta_and_blast_results` are:

- a FASTA file for each species
- a BLAST results file for each species pair
- SequenceIDs.txt
- SpeciesIDs.txt

Examples of the format required for the files can be seen by running OrthoFinder on the supplied ‘ExampleDataset’ and looking in the ‘WorkingDirectory/’ created. A description is given below.

### 6.1 FASTA Files

```
Species0.fa
Species1.fa
...
```

Within each FASTA file the accessions for the sequences should be of the form ‘x-y’ where x is the species ID number, matching the number in the filename and y is the sequence ID number, which starts from 0 within each species. So the first few lines of start of ‘Species0.fa’ would look like:

```
>0_0
MFAPRGK...

>0_1
MFAVYAL...

>0_2
MTTIID...
```

And the first few lines of start of ‘Species1.fa’ would look like:

```
>1_0
MFAPRGK...

>1_1
MFAVYAL...

>1_2
MTTIID...
```

### 6.2 BLAST Results Files

For each species pair ‘x’, ‘y’ there should be a BLAST results file ‘Blastx.y.txt’ where x is the index of the query FASTA file and y is the index of the species used for the database. Similarly, there should be a BLAST results file ‘Blasty.x.txt’ where y is the index of the query FASTA file and x is the index of the species used for the database. The tabular BLAST output format 6 should be used. The query and hit IDs in the BLAST results files should correspond to the IDs in the FASTA files.

**Aside, reducing BLAST computations:** Note that since the BLAST queries are by far the most computationally expensive step, considerable time could be saved by only performing  $\frac{n(n+1)}{2}$  of the

species versus species BLAST queries instead of  $n^2$ , where  $n$  is the number of species. This would be done by only searching ‘Speciesx.fa’ against the BLAST database generated from ‘Speciesy.fa’ if  $x \leq y$ . The results would give the file ‘Blastx\_y.txt’ and then this file could be used to generate the ‘Blasty\_x.txt’ file by swapping the query and hit sequence on each line in the results file. This should have only a small effect on the generated orthogroups.

### 6.3 SequenceIDs.txt

The SequenceIDs.txt give the translation from the IDs of the form x\_y to the original accessions. An example line would be:

```
0_42: gi|290752309|emb|CBH40280.1|
```

The IDs should be in order, i.e.

```
0_0: gi|290752267|emb|CBH40238.1|
0_1: gi|290752268|emb|CBH40239.1|
0_2: gi|290752269|emb|CBH40240.1|
...
...
1_0: gi|284811831|gb|AAP56351.2|
1_1: gi|284811832|gb|AAP56352.2|
...
```

### 6.4 SpeciesID.txt

The SpeciesIDs.txt file gives the translation from the IDs for the species to the original FASTA file, e.g.:

```
0: Mycoplasma_agalactiae.faa
1: Mycoplasma_gallisepticum.faa
2: Mycoplasma_genitalium.faa
3: Mycoplasma_hyopneumoniae.faa
```