



S P E C T R E

A PHYLOGENETIC TOOLSET

Spectre Documentation

Release 0.5

Sarah Bastkowski Daniel Mapleson
Monica Balvociute Andreas Spillner Taoyang Wu
Vincent Moulton

Oct 09, 2017

Contents

1	Installation	3
1.1	Platform-specific installer	3
1.2	Pre-packaged Tarball	4
1.3	From source	4
2	Running SPECTRE Tools	7
2.1	Command-line scripts	7
2.2	Changing the JVM memory limits	8
2.3	Optimisers	8
3	The Tools	9
3.1	Netmake	10
3.1.1	References	10
3.2	NetME	11
3.2.1	References	11
3.3	Flat Neighbor Joining (FlatNJ)	11
3.3.1	Walk through usage examples	13
3.3.2	File formats	14
3.3.3	References	15
3.4	SuperQ	16
3.4.1	QWeights example file	17
3.4.2	References	18
3.4.3	Credits	18
3.5	SPECTRE Viewer	18
3.5.1	File menu	19
3.5.2	Edit menu	20
3.5.3	View menu	20
3.5.4	Labelling Options	20
3.5.5	Tools	21
3.6	Miscellaneous Tools	21
3.6.1	SFilter	21
3.6.2	Distance Matrix Generator	21
3.6.3	Split Comparison Tool	22
4	Developing the codebase	23
4.1	Making your own apps using our core library	23
4.2	Source Control	23

4.3	Integrated Development Environments	24
4.4	Project Structure	24
4.5	Updating Maven Project Versions	25
4.6	Creating platform specific installers	25
5	Resources	27
6	Citing	29
7	Credits	31
8	Issues	33
9	Availability and License	35



SPECTRE a suite of tools for inferring evolutionary patterns associated with Reticulate Evolution that primarily either create or use split systems (a collection of bipartitions of the taxa) representable in two dimensions, such as SuperQ, FlatNJ, NetME and several NeighborNet variants. SPECTRE provides a graphical interface to both drive and visualise the output. The viewer allows the user to pan, zoom, rotate and modify the network, making the split networks easier to assess and interpret. All tools in SPECTRE also have command line interfaces enabling their use within pipelines, and on servers and other high performance computing environments.

SPECTRE's source code is written primarily in Java and is freely available under the [GNU GPLv3 license](#) via [github](#). In addition our core data structures, algorithms and IO routines available as a library via [Maven](#), should user's which to easily leverage our code in their own projects.

SPECTRE can be installed via three main methods either from a platform-specific install, pre-packaged tarball, or directly from github source repository via a *git clone*. The necessary steps for all methods are described in the following sections.

Some of the tools in SPECTRE use external mathematical optimizers for solving linear and quadratic problems. Should you install from a platform-specific installer or cross-platform tarball then a working version of Apache Maths and JOptimizer is included. However, some users may want to use optimizers from other vendors or sources such as Gurobi. In this case you will need to install another tool called *metaopt* first, and then install from source. Metaopt can be obtained from <https://github.com/maplesond/metaopt>. Please follow the instructions in the *metaopt* README for how to add other optimizers. Then follow the instructions for installing from source below.

Platform-specific installer

SPECTRE currently supports Debian/Ubuntu, MacOS and windows installers. Users of these platforms should find the installation experience self-explanatory. They should only need to download the appropriate file from the github repository releases page: <https://github.com/maplesond/spectre/releases> and then double click the downloaded file. There are however, some platform-specific considerations for running SPECTRE which are detailed below.

Debian/Ubuntu

Installing the debian file will put a shortcut for the GUI into either your *Science* or *Other* menu section depending on how you have your system configured. Links to the command-line versions of your apps will be added to `/usr/bin` and the program itself is installed to `/usr/share/spectre`.

MacOS

After double-clicking the DMG image file, drag the SPECTRE app into the Applications folder. You should then be able to access the GUI from the launchpad.

We have not at present installed the command-line tools onto the PATH, but they are present on your system and can be found in `/Applications/Spectre.app/Contents/MacOS`. You can either run them from here or manually link them into `/usr/local/bin` in order to have them directly available from the terminal.

Windows

The windows installer will allow you to install SPECTRE to a directory of your choosing. After which it should be available from the start menu.

Please note the command-line versions of the tools are not available on windows via this method.

Pre-packaged Tarball

Before starting the installation please ensure that the Java Runtime Environment (JRE) V1.8+ is installed and configured for your environment. You can check this by typing the following at the command line: `java -version`. Double check the version number exceeds V1.8.

The installation process from tarball is simple. The first step is acquire the tarball from <https://github.com/maplesond/spectre/releases>. Then unpacking the compressed tarball to a directory of your choice. The unpack command is: `tar -xvf spectre-<version>-<platform>.tar.gz`. This will create a sub-directory called `spectre-<version>` and in there should be the following further sub-directories:

- `bin` - contains scripts allowing the user to easily run all the tools. In general, the scripts are all command line tools except for `spectre` suffix. Scripts for all platforms are available, in general, those with no extension should work on linux and mac platforms, and those with a `.bat` extension should run on windows.
- `doc` - a html, pdf and text copy of the complete manual
- `etc` - contains configuration files and other resources for the application
- `examples` - Example files to help you get started with the SPECTRE tools
- `repo` - contains the java classes used by SPECTRE

Should you want to run the tools without referring to their paths, you should ensure the `bin` directory is on your `PATH` environment variable.

From source

SPECTRE is a java 1.8 / maven project. Before compiling the source code, please make sure the following tools are installed:

- GIT
- Maven (make sure you set the `m2_home` environment variable to point at your Maven directory) <https://maven.apache.org/>
- JDK v1.8+ (make sure you set the `JAVA_HOME` environment variable to point at your JDK directory)
- Make
- Sphinx (may require you to install python, also make sure the sphinx-build is on the path environment variable) <http://www.sphinx-doc.org/en/stable/>

You also need to make sure that the system you are compiling on has internet access, as it will try to automatically incorporate any required java dependencies via maven. Because SPECTRE is a maven project, almost all the other dependencies (not mentioned here) will be downloaded automatically as part of the Maven buildcycle. However, the one exception to this is a java library called metaopt (described at the beginning of this section), which provides a common interface to several open source and commercial optimizers. Metaopt can be obtained from: <https://github.com/maplesond/metaopt>. Please follow the instructions in the metaopt README and make sure the metaopt library has been added to your local maven repository. After this, you can proceed with the SPECTRE installation.

Now type the following:


```
git clone https://github.com/maplesond/spectre.git
cd spectre
```

Then type:

```
mvn clean install
```

or, if you wish to enable gurobi optimizer support:

```
mvn clean install -P gurobi
```

Note: If you cannot clone the git repositories using “https”, please try “ssh” instead. Consult github to obtain the specific URLs.

Assuming there were no compilation errors. The build, hopefully the same as that described in the previous section, can now be found in `./build/spectre-<version>`. There should also be a `dist` sub directory which will contain a tarball suitable for installing SPECTRE on other systems.

Running SPECTRE Tools

If you have installed SPECTRE from a platform-specific installer and wish to launch SPECTRE's graphical interface then you should just need to click the shortcut in your platforms launch menu. Details for specific tools are discussed in subsequent sections. However, first we will run through some command-line specific considerations that you should be aware of such as where to find the tools, how to change the memory limits.

Command-line scripts

We have tried to make all tools in spectre as platform independent and as simple to use as possible. For example, instead of typing `java -jar <path to executable jar>`, which is a typical line used to start a java program, we created wrapper scripts for each tool. On unix, mac and cygwin platforms these scripts have been generated without any extension, to make user experience of running the tools as close to that of running a native binary as possible. However, on windows the scripts have a `.bat` extension. Please keep this in mind when reading the tool specific documentation. All examples in this documentation are assumed to be running on a unix or mac system, so if you are running on windows please add a `.bat` suffix to the script name.

If you installed from pre-packaged tarball all these executable scripts can be found in the `bin` subdirectory of the spectre installation. If built from source then spectre from source code the scripts can be found in `<project_dir>/build/spectre-<version>/bin`.

To run spectre tools without specifying the full path add the bin directory onto your PATH environment variable.

If installed from a platform-specific installer then the location of these scripts is as follows:

debian - Installed to `/usr/share/spectre/bin` with links to `/usr/bin`, so the scripts will be directly available from a terminal without modifying the PATH.

mac - Installed to `/Applications/Spectre.app/Contents/MacOS`

windows - Will be installed to `<install_folder>/bin`

Changing the JVM memory limits

Normally 64-bit java processes are capped to 2GB of Heap Space, although you can find out the actual limit on your system by typing on a linux, mac or cygwin machine:

```
java -XX:+PrintFlagsFinal -version | grep MaxHeapSize.
```

On windows just type:

```
java -XX:+PrintFlagsFinal -version
```

And then manually find the line containing `MaxHeapSize`. Sometimes some of the tools may need more memory than this when processing large datasets. If you encounter an `OutOfMemory` error while running a spectre tool and you have more memory available on your system then you might want to consider increasing max heap size of your process.

To do this with the spectre scripts then recommended way is to modify an environment variable called `JAVA_OPTS` with the JVM options you wish to change. This environment variable should be recognised by all the spectre scripts. So, for example, if you want to set the `Xmx` value (the value associated with max heap size) permanently to 4GB you would type something like this on a bash shell:

```
export JAVA_OPTS='-Xmx4g'
```

Should you only wish to set the memory for a particular instance of a spectre program (in this case the main SPECTRE GUI spectre) your command would like this on a bash shell:

```
JAVA_OPTS='-Xmx4g'; ./spectre
```

The above line assumes the bin directory is the current working directory and has not been setup on the PATH.

Note: Any other java VM option can be set using the `JAVA_OPTS` environment variable.

Optimisers

Some tools within spectre use optimizers to maximise or minimise an objective function given a set of constraints. In order to provide flexibility to test out different optimizers with spectre tools we enable the developer to write their problems to be solved by an optimizer using our own data types. These are then translated to the data types for the optimizer selected by the user at runtime. This way the developer can keep their code optimizer agnostic.

The code for managing optimizers has been extracted from spectre and is available in a separate project called *metaopt*. This is freely available from <https://github.com/maplesond/metaopt>. Metaopt should be installed and configured with the optimisers you wish to use prior to compilation of spectre.

Currently the following optimizers are supported by metaopt, each of which has its own pros and cons:

Optimiser	Quadratic?	Configura- tion	Free	Description
Apache Math 3	No	Built in	Yes	Part of Apache Math 3, available through maven central
JOptimizer	Yes	Bundled	Yes	Pure java implementation but some dependencies not in maven
Gurobi	Yes	External	No	A popular and fast but commercial optimizer

See [Installation](#) for more details. Also please refer to the metaopt README file.

CHAPTER 3

The Tools

The main graphical interface for the software is accessed via the *SPECTRE Viewer*. From here you can visualise and modify trees and networks and launch most of the main tools within the toolkit.

The main tools and their workflows are listed here.

Netmake - Creates a circular split system from a multiple sequence alignment (fasta or Nexus) or a distance matrix (Nexus, Phylip or Emboss format) producing an outer-planar network. Allows the user to select either NeighborNet or NetMake's own method for constructing the circular ordering of taxa. The distance matrix can be presented in any of the following formats:

1. Nexus format. Must contain both “taxa” and “distances” block.
2. Phylip format.

Flat Neighbor Joining (FlatNJ) - Creates a flat planar split network where labels are not forced to the edges of the network. FlatNJ can therefore produce richer and less distorted networks than NeighborNet if supported by the data. FlatNJ however cannot be driven from a distance matrix alone, requiring quartet data instead. FlatNJ can automatically generate quartets from the following input types:

1. Multiple sequence alignment in Fasta or Nexus format. If Nexus format is used and contains a “distances” block, this information can help guide split weight estimation.
2. Geographical coordinates in Nexus format
3. Weighted split system in Nexus format

SuperQ - Creates a Supernet from a collection of partial input trees. Input trees can be derived from several different input formats:

1. Qweights files - simple, monopurpose format used by QNet and QMaker.
2. Nexus files with “st quartets” (old format) or Quartets blocks
3. Nexus files with “st splits” (old format) or Splits blocks
4. Nexus files with distance blocks
5. Treebase syntax Nexus files with TREES blocks
6. Newick trees, with or without branch lengths

The tools within spectre are divided into sub-groups based on their functionality. More detail about the specific tools are listed in these sub-groups:

Netmake

Netmake creates a compatible split system with circular ordering from a distance matrix. Netmake's default mode uses the well-known Neighbor-Net algorithm (Bryant and Moulton, 2004) for quickly ($O(n^3)$) constructing circular split systems. Neighbor-Net is an extension of the Neighbor Joining (NJ) algorithm that is used to create trees, with the difference that instead of agglomerating two neighbors into a new node immediately, Neighbor-Net pairs up another set of candidate nodes before agglomerating into a new node. This process generates a collection of splits for which it might not be possible to represent in a single tree, hence can be used to create split networks.

Neighbor-Net via netmake takes in either a distance matrix in nexus, phylip or emboss format, or an MSA in fasta or nexus format. If an MSA is provided netmake calculates the distance matrix from the MSA using the Jukes Cantor method by default, although the user can use an alternate distance calculation using the `-dc` option. Netmake will produce a circular split network in nexus format so a typical Neighbor-Net run from a nexus file (bees.nex) containing a distance matrix would produce a circular split system in an output file called "bees_out.network.nex" with the following command:

```
netmake -o bees_out bees.nex
```

Netmake's alternate mode is based partially on work by Levy and Pachter (Levy and Pachter, 2008), where we construct a circular split system by greedily optimising the minimum evolution criterion. For more information on this mode please see Thesis Chapter 4. Netmake in this mode requires that the user specify the `-alt` switch. Additional options maybe specified to select the runmode for the tool. The full list of options takes this form:

```
netmake -alt [-o <output_prefix> -w <weighting_1> -x <weighting_2> -z <tree_weighting>  
→ <input_file>
```

A simple command line to run using the travelling sales man circular ordering configuration (the default option in alt mode) looks like this:

```
netmake -alt -o bees_tsp_out bees.nex
```

And the command line run netmake in hybrid greedy minimum evolution would look like this:

```
netmake -alt -o bees.t -w GREEDY_ME -x TREE bees.nex
```

Alternatively, Neighbor-Net and the netmake alternative can be invoked through the Tools menu in SPECTRE Viewer.

References

- 4. Bryant and V. Moulton. Neighbor-net: an agglomerative method for the construction of phylogenetic networks. *Mol. Biol. Evol.*, 21:255–265, 2004.
- 4. Levy and L. Pachter. The Neighbor-Net Algorithm. *Advances in Applied Mathematics*, 47(2):240–258, 2011.
- 19. Bastkowski. From Trees to Networks and Back. *PhD Thesis*. 2013

NetME

NetME constructs a minimum evolution (ME) tree from the specified split network with an implied circular order. Such split systems can, for example, be generated by the NeighborNet algorithm or any for constructing phylogenetic networks (see [Netmake](#)). More specifically, NetME is the implementation of an $O(n^4)$ algorithm for finding an optimal minimum evolution tree in a circular set of splits where the set of species is of size n . For more information on the algorithm please see and please cite (S. Bastkowski et al, 2014).

NetME takes in a nexus file containing a distance matrix and a circular split system (such a file can be generated by running Neighbor-Net via netmake). NetME produces two output files:

- The weighted split system, in nexus format, corresponding to a restricted minimum evolution tree, where the weights are recalculated by using a Non-Negative Least Squares (NNLS) method.
- A file containing the tree length of tree weighted with Ordinary Least Squares (OLS).

In addition the user can request to output, using the '-l' option, the weighted split system, in nexus format, corresponding to a restricted minimum evolution tree, where the weights are derived from the Ordinary Least Squares (OLS) method used for constructing the tree.

NetME can be run from the command line like this:

```
netme [-o <output_prefix> -l] <nexus_file>
```

Alternatively, NetME can be invoked through the Tools menu in the SPECTRE graphical interface.

References

- 19. Bastkowski, A. Spillner, V. Moulton (2014) Fishing for Trees with NeighborNets. Information Processing Letters 114(1-2): 13-18.

Flat Neighbor Joining (FlatNJ)

The Flat Net Joining (FlatNJ) method for constructing split networks is presented in (Balvociute et al. 2014). FlatNJ first generates a system of 4-splits (quadruples) from one of the following types of data:

1. multiple sequence alignment
2. geographical coordinates
3. weighted split system

In a system of 4-splits on a set X with $|X| \geq 4$, for each 4-element subset a, b, c, d of X , all seven possible 4-splits $abcd, bacd, cabd, dlabc, abcd, acbd$ and $adbc$ are assigned a non-negative weight. FlatNJ then, from a given system of 4-splits, generates a split network that is guaranteed to be (almost) planar. FlatNJ is based on an agglomerative approach similar to the one used in methods such as NeighborJoining (Saitou and Nei, 1987) and NeighborNet (Bryant and Moulton, 2004).

The run time for generating the unweighted split system underlying the final result is $O(n^4)$. Suitable weights for the splits are estimated using a least squares fitting between the given system of 4-splits and the system of 4-splits induced by the resulting weighted split system. To solve the least squares problem the SPECTRE metaopt system is used to hook into external solvers, although we recommend that Gurobi Optimizer (www.gurobi.com) is used as the external solver if available. The resulting weighted splits are filtered using the approach described in (Grunewald et al. 2007) and, using the method described in (Spillner et al. 2011), an almost planar split network is constructed. In order to view the drawing of the network the split network viewer included in SPECTRE can be used or, alternatively, external software such as SplitsTree (Huson and Bryant, 2006).

When running FlatNJ for large datasets, it is recommended to allocate more RAM for the heap space. Otherwise an `OutOfMemoryError` may occur. Heap space can be increased using the Java VM options `-Xms` or `-Xmx`. See [Running SPECTRE Tools](#) for more information on how to adjust these options with SPECTRE scripts.

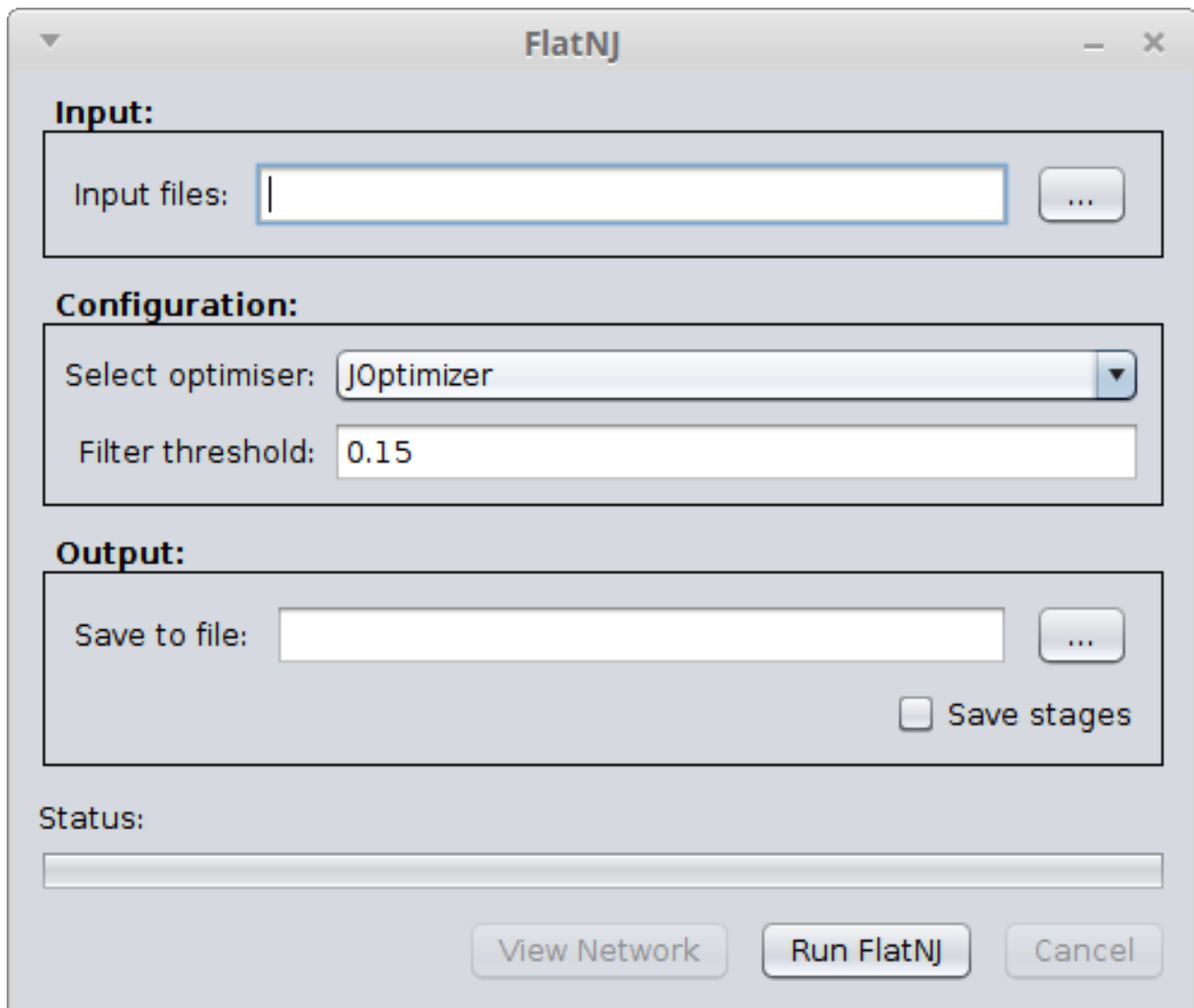
The input to FlatNJ must be provided using the positional argument `<input>` and the output is written to the file indicated by the `-o/--out <nexus file>` command line parameter.

Split with relatively small weight that are incompatible with much bigger splits in the output are filtered out from the network using the approach described in (Grunewald et al. 2007). The filtering threshold is a number between 0 and 1. The default filtering threshold is 0.15. It can be altered using the `-t/--threshold <[0.0, 1.0]>` parameter; choosing 0.0 as the threshold results in an unfiltered network whereas 1.0 yields a network that is a tree.

Usage examples:

- To compute a split network from a system of 4-splits in the file `system_of_4s.nex` using the default threshold for filtering splits: `flatnj -o network.nex system_of_4s.nex`
- To compute a split network from a system of 4-splits in the file `system_of_4s.nex` using a filtering threshold of 0.2: `flatnj -thr 0.2 -o network.nex system_of_4s.nex`

Alternatively, FlatNJ can be invoked through the Tools menu of the SPECTRE viewer.



The image shows a graphical user interface window titled "FlatNJ". It is divided into several sections: "Input:", "Configuration:", "Output:", and "Status:". The "Input:" section has a text field labeled "Input files:" followed by a button with three dots. The "Configuration:" section has a dropdown menu labeled "Select optimiser:" with "JOptimizer" selected, and a text field labeled "Filter threshold:" with "0.15" entered. The "Output:" section has a text field labeled "Save to file:" followed by a button with three dots, and a checkbox labeled "Save stages" which is currently unchecked. The "Status:" section has a long, empty text field. At the bottom of the window are three buttons: "View Network", "Run FlatNJ", and "Cancel".

Walk through usage examples

The following walk through examples illustrate the usage of FlatNJ for molecular sequence data and for geographical data. Note that the drawing of the network can be adjusted by the user in SplitsTree.

Molecular Sequence Data

To illustrate FlatNJ's usage for sequence data, we use sequences of fluorescent proteins (<spectre_dir>/examples/flatnj/colors_aln.faa file in the examples directory). For more information on this data set see the results section in (Balvociute et al. 2014). The following steps will guide you through the whole process of the network construction for the fluorescent protein data set:

1. Open a terminal window and change to the directory of FlatNJ.
2. To compute a split network from the protein sequences fasta file type: `flatnj -o <output_dir>/colors.nex <spectre_dir>/examples/flatnj/colors_aln.faa`
3. To view the network launch the SPECTRE viewer and open <output_dir>/colors.nex. The network displayed by the viewer should look similar to the one in Figure 1.

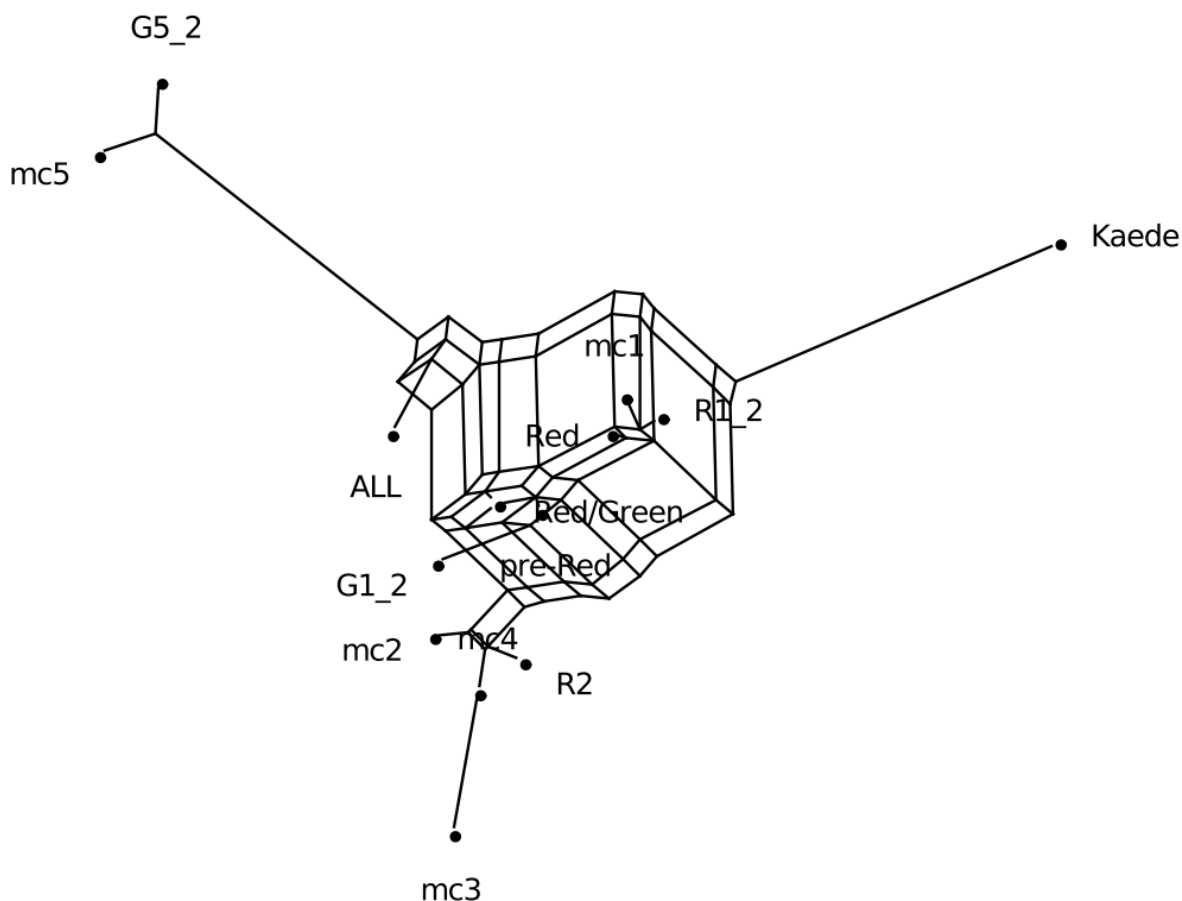


Figure 1: Split network generated from the multiple protein sequence alignment from <spectre_dir>/examples/flatnj/colors_aln.faa.

Geographical data

To illustrate FlatNJ's usage for geographical data, we use coordinates of some of the European capitals (`<spectre_dir>/examples/flatnj/europe.nex`). The following steps will guide you through the whole process of the network construction for the European capitals data set:

1. Open a terminal window and change to an empty working directory.
2. To compute a split network from the geographical data type: `flatnj -o <output_dir>/europe_net.nex <spectre_dir>/examples/flatnj/europe.nex`
3. To view the network launch the SPECTRE viewer and open `<output_dir>/europe_net.nex`. The network displayed by the viewer should look similar to the one in Figure 2.

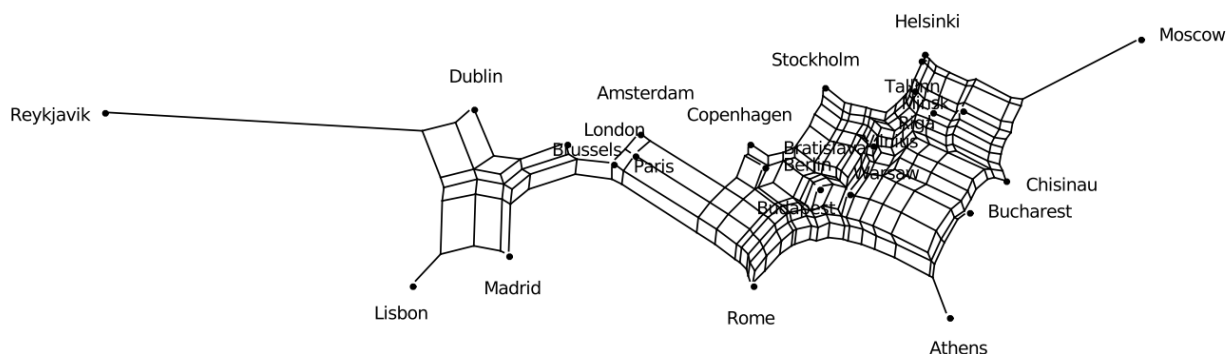


Figure 2: Split network generated from geographical coordinate data in `<spectre_dir>/examples/flatnj/europe.nex`. The network was rotated and flipped to align it with the usual representation on a map.

File formats

For input and output files the nexus format is used, with the exception of multiple sequence alignments that may also be provided as a fasta file. The various types of blocks in a nexus file used by this software package are listed in Table 1. The syntax of blocks specific to this software package is defined below. The syntax of commonly used blocks can be found e.g. in the SplitsTree manual (<http://www.splitstree.org/>).

Nexus block contents

Commonly used blocks:

- CHARACTERS multiple sequence alignment
- DATA multiple sequence alignment
- DISTANCES character distance matrix
- SPLITS split system
- NETWORK split network

Specific blocks:

- LOCATIONS geographic coordinates
- QUADRUPLES system of 4-splits

Multiple sequence alignments may be provided in either fasta or nexus files. In case the nexus format is used, sequences must be placed within the CHARACTERS or DATA blocks.

Geographical data can be processed in the form of coordinates of points in the plane using the LOCATIONS block. The syntax for this block is as follows:

```
BEGIN LOCATIONS;
  [DIMENSIONS NTAX=number-of-taxa;]
  [FORMAT LABELS={yes|left|no};]
MATRIX
  [label_1] x_1 y_1,
  [label_2] x_2 y_2,
  ...
  [label_ntax] x_ntax y_ntax,
  ;
END;
```

Distance matrices

Character distance matrices that are used by FlatNJ for the estimation of 4-split weights from multiple sequence alignments must be placed in the DISTANCES block.

Systems of 4-splits

Systems of 4-splits are stored in the block QUADRUPLES. Each quadruple contains weights of all possible 4-splits over a set of 4 taxa. The syntax of the block is as follows:

```
BEGIN QUADRUPLES
  DIMENSIONS NTAX=number-of-taxa NQUADRUPLES=number-of-quadruples;
  [FORMAT [LABELS={LEFT|NO}] [WEIGHTS={YES|NO}];]
MATRIX
  [label_1] : a1 b1 c1 d1 : [weight_a1|b1c1d1 weight_b1|a1c1d1 weight_c1|a1b1d1
    weight_c1|a1b1c1 weight_a1b1|c1d1 weight_a1c1|b1d1 weight_a1d1|b1c1],
  [label_2] : a2 b2 c2 d2 : [weight_a2|b2c2d2 weight_b2|a2c2d2 weight_c2|a2b2d2
    weight_c2|a2b2c2 weight_a2b2|c2d2 weight_a2c2|b2d2 weight_a2d2|b2c2],
  ...
  [label_n] : an bn cn dn : [weight_an|bncndn weight_bn|ancndn weight_cn|anbndn
    weight_cn|anbn cn weight_anbn|cndn weight_ancn|bndn weight_andn|bncn],
  ;
END;
```

All weights must be written in the same line.

References

- M.Balvociute, A.Spillner and V.Moulton. FlatNJ: A novel network-based approach to visualize evolutionary and biogeographical relationships. *Systematic Biology*, 63(3):383–396, 2014.
- D.Bryant and V.Moulton. Neighbor-net: an agglomerative method for the construction of phylogenetic networks. *Mol. Biol. Evol.*, 21:255–265, 2004.
- S.Grunewald, K.Forslund, A.Dress and V.Moulton. Qnet: An agglomerative method for the construction of phylogenetic networks from weighted quartets. *Mol. Biol. Evol.*, 24(2):532–538, 2007.
- D.H.Huson and D.Bryant. Application of phylogenetic networks in evolutionary studies. *Mol. Biol. Evol.*, 23(2):254–267, 2006.

- N.Saitou and M.Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406–425, 1987.
- A.Spillner, B.Nguyen, and V.Moulton. Constructing and drawing regular planar split networks. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 9:395–407, 2011.

SuperQ

SuperQ constructs a phylogenetic supernetwork from a set of weighted or unweighted partial trees by using quartets. SuperQ can directly take system of quartets as input via nexus format or through a qweights format file (see [QWeights example file](#) for more information). Alternatively if quartets are not readily available, SuperQ can automatically construct them from one of the following formats:

- Nexus files with st splits (old format) or Splits blocks ([6], [5])
- Nexus files with distance blocks ([6], [5])
- Treebase syntax Nexus files with TREES blocks ([6])
- Newick trees, with or without branch lengths [8]

Example usage:

```
superq -o <outfile> [-s <scaling_optimiser> -x <primary_optimiser>
    -y <secondary_optimiser> -b <objective> -f <filter_threshold>] <input_file> [
    ↪<input_file>]...
```

Alternatively, SuperQ has a graphical interface that can be accessed via the Tools menu in the SPECTRE viewer:

Input:

Input files:

Optimiser Setup:

Select scaling optimiser:

Select primary optimiser:

Select secondary optimiser:

Select secondary objective:

Output:

Save to file:

☐ Filter:

Status:

When running SuperQ for large datasets, it is recommended to allocate more RAM for the heap space. Otherwise an `OutOfMemoryError` may occur. Heap space can be increased using the Java VM options `-Xms` or `-Xmx`. See [Running SPECTRE Tools](#) for more information on how to adjust these options.

The supernetwork constructed from SuperQ can be visualised in the [SPECTRE Viewer](#).

QWeights example file

The following is an example QWeights file:

```
taxanumber: 6;
description: artificial data;
sense: max;
taxon: 001 name: a;
taxon: 002 name: b;
taxon: 003 name: c;
taxon: 004 name: d;
taxon: 005 name: e;
taxon: 006 name: f;
quartet: 001 002 003 004 weights: 200 0 200;
quartet: 001 002 003 005 weights: 200 0 200;
```

```
quartet: 001 002 003 006 weights: 200 0 200;  
quartet: 001 002 004 005 weights: 210 0 210;  
quartet: 001 002 004 006 weights: 210 0 210;  
quartet: 001 002 005 006 weights: 410 0 410;  
quartet: 001 003 004 005 weights: 10 0 10;  
quartet: 001 003 004 006 weights: 10 0 10;  
quartet: 001 003 005 006 weights: 210 0 210;  
quartet: 001 004 005 006 weights: 200 0 200;  
quartet: 002 003 004 005 weights: 10 0 10;  
quartet: 002 003 004 006 weights: 10 0 10;  
quartet: 002 003 005 006 weights: 210 0 210;  
quartet: 002 004 005 006 weights: 200 0 200;  
quartet: 003 004 005 006 weights: 200 0 200;
```

References

- 19. Grünwald, A. Spillner, S. Bastkowski, A. Boegershausen, V. Moulton. SuperQ: Computing Supernet-works from Quartets. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(1): 151–160, 2013.
- 19. Grünwald, K. Forslund, A. Dress, V. Moulton. QNet: An Agglomerative Method for the Construction of Phylogenetic Networks from Weighted Quartets, *Molecular Biology and Evolution*, 24(2): 532–538, 2006.

Credits

The original version QNet and the original set of quartet tools were developed by:

- Stephan Grunewald
- Kristoffer Forslund

The original version of SuperQ was developed by:

- Sarah Bastkowski

The tools have been reengineered, optimised and integrated into SPECTRE by:

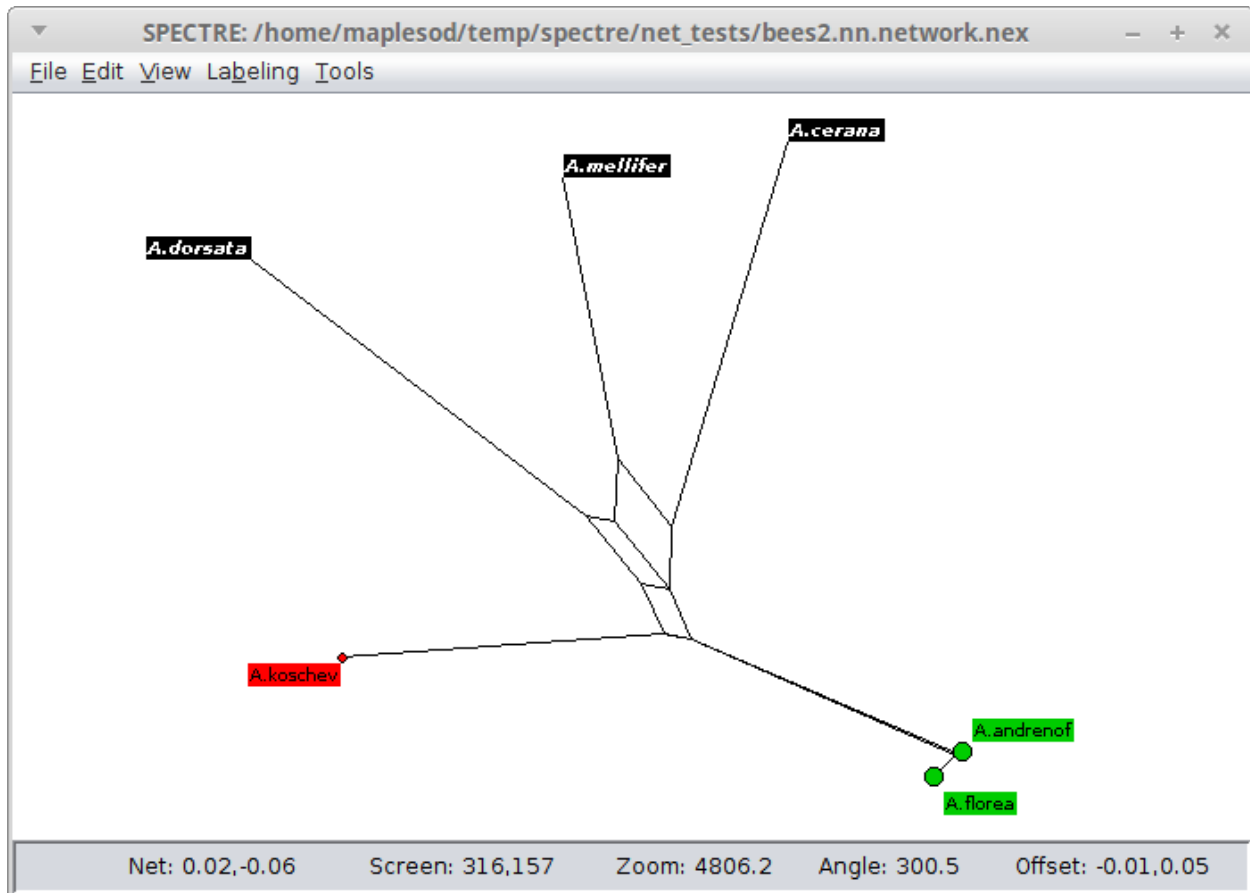
- Daniel Mapleson

All the tools have been developed, since inception, under the supervision of:

- Andreas Spillner
- Vincent Moulton

SPECTRE Viewer

The SPECTRE Viewing tool allows users to visualise trees and networks stored in a nexus file. An example screenshot of a loaded networking is shown below. The window is broken up from top to bottom into the menu bar, viewing canvas and status bar.



The viewer allows the user to navigate the network through pan, zoom and rotate controls. Additionally, properties of nodes can be altered, such as label positions, size and styles.

We will describe the functions the viewer offers by describing the menu bar items.

File menu

Open - Opens a nexus file (Ctrl-O). Files can also be opened dragging and dropping a suitable nexus file into window. Additionally, spectre can automatically load a file at startup by passing the file an argument from the command line. Suitable nexus files must either contain a “network” block describing the locations of nodes and edges of the network, or they must contain a split system. The viewer will automatically generate the nodes and edges from the split system to create the network.

Save - Saves the current network (Ctrl-S), or saves the network to an alternative nexus file name via “Save as”. The saved nexus file will contain the original split system, network nodes and edges and viewer configuration settings used at time of save. This allows the user to reload a network later without having to reapply all modifications.

Save image - Saves the image shown on the canvas. Multiple images formats are supported:

1. PDF - Contains vector graphics, suitable for publications.
2. PNG - Rasterised image, lossless compression
3. GIF - Rasterised image, lossless compression
4. JPG - Rasterised image, lossy compression

When typing a file name please ensure that it has one of the extensions listed above in order for netview to save to the correct format.

Edit menu

Copy - Copies the selected labels to the clipboard (Ctrl-C)

Select all - Selects all labels in the network (Ctrl-A)

Find - Selects labels that match a particular name or regular expression (Ctrl-F). This option launches a separate dialog window.

View menu

Users can orientate the image via mouse and key controls and through items in the View menu.

Pan - Move the image around the screen (Arrow keys). Can be controlled by holding the left-mouse button whilst moving the mouse over the canvas where the initial click occurs away from labels.

Rotate - Rotate the image around the centre of the canvas (Ctrl-Left and Ctrl-Right). Can be controlled through holding the right-mouse button and moving the mouse relative to the centre point.

Zoom - Zoom the image in or out (Ctrl-Up and Ctrl-Down). Can be controlled via the mouse scroll wheel.

Flip - The user can flip the image either horizontally (Shift-Ctrl-H) or vertically (Shift-Ctrl-V).

Optimise layout - This function rescale and translate the image so that the content fits the current canvas window best (Ctrl-P).

Show trivial splits - Toggles whether to show or remove splits that separate one taxon from the rest of the taxa in the network (Ctrl-T).

Show range - Toggles the ruler in the top left (Ctrl-R)

Labelling Options

The viewer provides multiple options for controlling the look, feel and positioning of labelled nodes in the network. These options are controlled via the Labelling menu.

Show labels - Turns on or off whether node labels should be visible (Ctrl-L)

Color labels - Turns on or off whether the labels color should be inverted (i.e. Black text on white background, or white text in a black box). This option is only relevant if show labels is switched on.

Format selected nodes - Clicking on this option brings up a dialog box that allows users to modify the shape, size and color of the currently selected nodes. In addition, they can modify the size and style of the current selected nodes' label text.

Fix all label positions - This forces the labels to stay in the same proximity to the node as currently viewed, even after the image is rotated or flipped.

Leaders - Sometimes taxa labels would be placed so close to each other that the text would overlap on the screen. This often occurs with internal nodes. To avoid this problem labels towards the edge of the window are joined to the network node via a "leader" line in order to aid readability. This sub-menu allows the user to control the look and feel of the leaders.

Tools

The SPECTRE graphical interface can also launch other tools in the package. These are launched via the Tools menu and will create a separate window where the user can specify input and output files as well as parameters to be passed to the algorithms. Details regarding the graphical interface for each tool are described along with the associated documentation.

Miscellaneous Tools

These tools were developed for particular tasks during the development of the SPECTRE tools and are included here in case they are of use to others.

SFilter

Sometimes it is useful to filter out splits in a split system that are weakly supported, i.e. have a low weight. Sometimes these splits may be the result from numerical error or noise. The sfilter tool can filter splits from a nexus file that have a weight below a certain threshold. Example usage:

```
sfilter [--output <outfile> --min_threshold <threshold>] <infile>
```

<infile> and <outfile> should be Nexus files with st splits blocks, whereas <threshold> is a real number. Only those splits whose weight is higher than the threshold number times the weight of the most highly weighted conflicting split are retained from <infile> to <outfile>.

Distance Matrix Generator

This tool can be run in one of three ways:

1. Random distance matrix generation
2. Convert MSA to distance matrix
3. Convert distance matrix to alternate file format

The mode is determined by the positional argument passed in. If that argument is an integer then this tool creates one or more phylip or nexus files with a randomly generated distance matrix using the integer value representing the number of taxa. Example usage:

```
distmatgen -s 10 -t nexus -o randomdm 20
```

So this will create 10 nexus files containing a “distances” block containing 20 taxa each, with distances between each taxa randomly generated separately for each file. Output filenames will be of the format “randomdm-*<sample_index>*.nex”.

Alternatively, if the argument is an MSA file in Fasta or Nexus format then we generate a distance matrix based on the specified calculator. By default this is Jukes Cantor. Example usage:

```
distmatgen -t phylip -o bees bees.fa
```

This command line would generate a distance matrix output to a phylip file called “bees.phy” from an MSA stored in fasta format.

Finally, if the input file contains a distance matrix already, then the distance matrix is simply converted to the specified file format. For example, to convert from an emboss style distance matrix to nexus format:

```
distmatgen -t nexus -o places.nex places.distmat
```

IMPORTANT NOTE: If a nexus file is passed in as input, then the tool will automatically check to determine whether it contains an MSA or a distance matrix and will run in the logical mode. If the nexus file contains both MSA and distance matrix, then it will generate a new distance matrix from the MSA.

Split Comparison Tool

This tool can take in one or more nexus files containing split systems. The first file provided is considered the reference and additional files are compared against the reference. This tool produces tab delimited output displaying the fields:

- filename - The file name of the nexus file containing a split system. This will have a (ref) suffix if it is the first file and therefore treated as the reference.
- nb_taxa - The number of taxa used in the split system
- nb_splits - The number of splits in the split system
- circular - Whether or not the split system is circular
- compatible - Whether or not the split system is compatible (tree-like)
- full - Whether or not the split system contains the full complement of splits
- match - Whether or not this split system matches the reference

A command line for comparing 4 different nexus files is as follows:

```
splitcompare ref.nex alt_method.nex nn.nex other.nex
```

Developing the codebase

SPECTRE is designed to be open source and easy to extend, maintain and develop by the community. This section of the manual will help you work with the codebase, whether you just plan to use our core library in your own java application or want to modify or extend our codebase.

Making your own apps using our core library

If you wish to use our core library within your own applications then we recommend that you use maven for our own project. You can then include our library as a maven dependency by adding the following snippet into the dependencies section in your pom.xml:

```
<dependency>
  <groupId>uk.ac.uea.cmp.spectre</groupId>
  <artifactId>core</artifactId>
  <version>[PUT-VERSION-HERE]</version>
</dependency>
```

And that's it! You should now automatically download the library when you import maven changes or run your maven build cycle. The published versions can be found at: <http://mvnrepository.com/artifact/uk.ac.uea.cmp.spectre/core/>

Should you prefer not to use maven in your project, you can download the pre-compiled jar from maven central directly, or alternatively, build SPECTRE and copy the core jar file from *build/spectre-
<version>/repo/uk/ac/uea/cmp/spectre/core/</version>*

The rest of this section assumes you want to modify or extend the spectre codebase.

Source Control

The source code for spectre is version controlled using GIT. The public repository is hosted on github at <https://github.com/maplesond/spectre.git>

If you plan to make contributions directly to the spectre codebase and want to work closely with us on a new tool or feature then please email daniel.mapleson@earlham.ac.uk about your planned changes. He can grant you write access to the codebase. We use the [Git-Flow](#) branching model in order to make it easier to work on the codebase as a team. The main takeaway message here is do NOT commit changes directly to the master branch as this might effect the stability of the suite for everyone! To make managing the branches easier we recommend a gitflow aware client tool, such as [SmartGIT](#).

However, for most external developers we recommend you [fork](#) our github repository. You are then free to use whichever branching model you like in your own fork. If you want to merge back changes to the original codebase then do so using the [pull request](#) mechanism.

Integrated Development Environments

SPECTRE was developed in the Java programming language and has become a relatively large project. Because Java is a relatively verbose language (as compared to a language like python), we strongly recommend using an Integrated Development Environment (IDE). This will enable you to easily visualise the project structure, navigate around the code, refactor code and generally be productive.

SPECTRE was developed using the [IntelliJ](#) IDE, but, while this is our preferred IDE, we do not store the IntelliJ project files in the repository. Instead we use maven to manage the project structure, making the spectre codebase IDE agnostic, so you should be able to use whichever Java IDE you are most familiar with. Most modern Java IDEs, and all those that have a wide user base, will also support maven project object models (POMs). The exact details of how to load spectre will vary from IDE to IDE but it should be as simple as opening an existing project and selecting the pom.xml in the root directory of spectre. The IDE should then load the project structure. You should now be ready to view, modify or extend the codebase.

Project Structure

Within the parent maven project for spectre there is a single “pom.xml” which describes common properties for all child modules. This file contains details such as project details, developer list, compiler settings, unit test configuration, common dependencies and some common jar packaging settings. Beyond the pom.xml there are the child modules themselves, which each have their own pom.xml describing their specific configuration. Broadly speaking the project is structured into two main areas: *core* and *apps*.

Core Contains classes that are used by other modules, that contain some kind of general functionality which means they can be used in different situations. These classes were broken down into sub groups based on their specific kind of functionality as follows:

Class Group	Package name	Description
Data structures	ds	Phylogenetic data structures relating to concepts such as Splits, Trees, Networks, Distances and Quartets
File Handling	io	Loading and saving common phylogenetic file formats. Specifically, Nexus and Phylip format.
Mathematics	math	Math related functionality such as basic statistics, matrix algebra, and storing of tuples.
User Interface	ui	Functionality to help with Command Line Interfaces and Graphical interfaces
Misc Utils	util	Miscellaneous functionality

Apps Contains all the applications managed by spectre. Most of these apps rely heavily on the *core* library.

Updating Maven Project Versions

We have included the version-maven-plugin to control versions. To update the version number for all sub modules and dependencies, from the parent project directory type: *mvn versions:set -DnewVersion=<VERSION_HERE>*.

Creating platform specific installers

We use the javapackager program that comes with the JDK for this. Currently we support .deb (debian/ubuntu), .dmg (mac) and .exe (windows) installers. To create the installer first ensure no maven module version numbers contain the “-SNAPSHOT” suffix (see above section for modifying version numbers), then build spectre on the required platform using the following command: *mvn clean install -Drelease*. You will find the installer within *./build/dist* (possibly *./build/dist/installer/bundles* depending on platform).

IMPORTANT NOTE: We strongly recommend you use JDK9 for this task. Results with JDK8 may vary. Please ensure the JAVA_HOME variable is set to ensure this correctly setup.

CHAPTER 5

Resources

SPECTRE is largely built upon Sarah Bastkowski's PhD thesis

SPECTRE was also presented as a poster at the Mathematical and Computational Evolutionary Biology (MCEB) 2014 conference: poster

CHAPTER 6

Citing

Should you use our software for your research please cite our preprint manuscript available on BioRxiv: <http://www.biorxiv.org/content/early/2017/07/27/169177>.

The paper for spectre is currently being prepared. If you use spectre in your work and wish to publish in the meantime please refer the project's web page: <https://github.com/maplesond/spectre.git>

Additionally, spectre consists on a number of tools that have already been published, if your work uses any of these pre-published tools please also cite the associated publication(s). A reverse chronological list of all publications related to tools now built into spectre are in the table below.

FlatNJ Monika Balvociute, Andreas Spillner and Vincent Moulton, 2014. FlatNJ: A novel network-based approach to visualize evolutionary and biogeographical relationships. *Systematic Biology*.

SuperQ Stefan Grunewald, Andreas Spillner, Sarah Bastkowski, Anja Bogershausen and Vincent Moulton, 2013. SuperQ: computing supernetworks from quartets. *EE/ACM Transactions on Computational Biology and Bioinformatics*.

NetME Sarah Bastkowski, Andreas Spillner and Vincent Moulton, 2013. Fishing for minimum evolution trees with Neighbor-Nets. *Information Processing Letters*.

NetMake Dan Levy and Lior Pachter, 2010. The Neighbor-Net Algorithm. *Advances in Applied Mathematics*.

QNet Stefan Grünwald, Kristoffer Forslund, Andreas Dress and Vincent Moulton, 2006 QNet: An Agglomerative Method for the Construction of Phylogenetic Networks from Weighted Quartets *Molecular Biology and Evolution* Volume 24, Issue 2

SPECTRE is an international collaboration. Active contributors to spectre are listed in the following table (in publication order):

Name	Email	Current Institute	Contribution
Daniel Mapleson	daniel.mapleson@earlham.ac.uk	Earlham Institute (EI)	General software development, testing, optimisation and codebase management
Sarah Bastkowski	sarah.bastkowski@earlham.ac.uk	Earlham Institute (EI)	SuperQ, NetMake, NetME, NeighborNet Implementation, Algorithms, Optimisers
Monica Balvociute	???	University of Otago	FlatNJ, NetView
Andreas Spilner	anspillner@gmail.com	NA	Mathematics, Algorithms, Optimisers, SPECTRE development
Taoyang Wu	taoyang.wu@gmail.com	University of East Anglia (UEA)	Mathematics, Testing
Vincent Moulton	vincent.moulton@cmp.uea.ac.uk	University of East Anglia (UEA)	Mathematics, Algorithms

In addition, the people that contributed to the original tools which are now contained within spectre are listed in the following table (in alphabetical order):

Name	Contribution
Anja Boegershausen	SuperQ
Kristoffer Forslund	QNet
Stephan Gruenewald	QNet, SuperQ

CHAPTER 8

Issues

Should you discover any issues with spectre, or wish to request a new feature please raise a ticket at <https://github.com/maplesond/spectre/issues>. Alternatively, contact Sarah Bastkowski at sarah.bastkowski@earlham.ac.uk; or Daniel Mapleson at: daniel.mapleson@earlham.ac.uk

CHAPTER 9

Availability and License

Open source code available on github: <https://github.com/maplesond/spectre.git>

SPECTRE is available under GNU GLP V3: <http://www.gnu.org/licenses/gpl.txt>