RAWR v1.0

Local Galaxy Instance, GUI desktop client, and Web Server

Plus, API

Supplementary Document

Table of Contents

Overview

RAWR is a non-parametric resampling method written in Python. We provide two resampling algorithms here, RAndom Walk Resampling (RAWR) and SEquential RESampling (SERES), for performing support estimation for multiple sequence alignments and phylogenetic trees. RAWR and SERES conceptually work by random walking along biological molecular sequences and retaining the sequence dependence during the resampling process. We present this tutorial for people who prefer to use a GUI application or a website interface. The algorithms are freely available under the GNU General Public License.

Installation

Software Install

RAWR has a GUI interface tested to work on Apple macOS Sequoia 15.0, Linux Ubuntu 22.04 LTS, and Windows 11. Please make sure you have set **python** to link to Python3. We have last checked on Python 3.10.15 that the following python dependencies are compatible.

Open a terminal and navigate to the **rawr-web-software** folder. We provide code for Anaconda and Virtualenv to manage your python packages.

Virtualenv v20.26.6 commands:

```
cd rawr-software
python -m venv .rawr
. .rawr/bin/activate

pip install "ete3>=3.1.3"
pip install "PyQt5>=5.15.10"
pip install "numpy>=1.26.4"
pip install "scipy>=1.14.1"
pip install "sklearn>=1.5.2"
pip install "Bio>=1.84"
pip install "flask>=3.0.3"
pip install "pathlib>=1.0.1"
pip install "pytz>=2024.1"
pip install "python-dateutil>=2.9.0post0"
pip install "pandas>=2.2.2"
pip install "joblib>=1.4.2"
```

If you prefer Anaconda package manager, use the command below to install dependencies. We used Anaconda v22.9.0.

```
conda create -n python3.10 python=3.10
conda activate python3.10

conda install -c conda-forge -c etetoolkit ete3 pyqt numpy scipy scikit-learn biopython flask pathlib
pytz python-dateutil pandas
```

To run RAWR GUI software, run the following command in rawr-software folder:

```
python main.py
```

Website Install

We provide the codebase to set up a RAWR web server to run on your local host. Users can directly use the web server for small dataset analysis. The limitation of the web server is 50Mb of the input alignment with no more than 50 taxa.

First, please make sure your local host's operating system is one of the following: Linux or macOS. Please make sure you have set **python** to link to Python3.

If you are using macOS, please change your terminal shell to Bash.

If you're using Windows, please install Windows Subsystem for Linux and reboot prior to setting up the RAWR web service. If you have access to Hyper-V, you can alternatively create a Linux virtual machine.

The web server has been tested on Linux macOS Sequoia 15.0 and Ubuntu 22.04 LTS.

Open a terminal and navigate to the **rawr-web-software** folder. We provide code for Anaconda and Virtualenv to manage your python packages.

Virtualenv v20.26.6 commands:

```
cd rawr-web
python -m venv .rawr
. .rawr/bin/activate

pip install "ete3>=3.1.3"
pip install "PyQt5>=5.15.10"
pip install "numpy>=1.26.4"
pip install "scipy>=1.14.1"
pip install "sklearn>=1.5.2"
pip install "Bio>=1.84"
pip install "flask>=3.0.3"
pip install "pathlib>=1.0.1"
pip install "pytz>=2024.1"
pip install "python-dateutil>=2.9.0post0"
pip install "pandas>=2.2.2"
pip install "joblib>=1.4.2"
```

Conda v22.9.0 commands:

```
conda create -n python3.10 python=3.10
conda activate python3.10

conda install -c conda-forge -c etetoolkit ete3 pyqt numpy scipy scikit-learn biopython flask flask-
mail pathlib pytz python-dateutil pandas
```

Second, set up your server e-mail.

Its purpose is to notify the user when and where the results can be found, in case they time out or exit from the homepage, they will still get the result safely delivered via email when it's ready.

We have included instructions below to set up a Gmail account as a server sender. Setting up Flask to use an existing emailing service is the easiest way to use the asynchronous emailing option, but feel free to adapt the code to your needs.

1. Turn on 2FA for your specific gmail account.
2. Generate an "App password" with Gmail.
3. Edit the file **app.py** in the following app.config.update() section, updating **MAIL_USERNAME** and **MAIL_PASSWORD**.

```
app.config.update(
    MAIL_SERVER='smtp.gmail.com',
    MAIL_PORT=465,
    MAIL_USE_SSL=True,
    MAIL_USERNAME='username@gmail.com',
    MAIL_PASSWORD='app password'
)
```

When

To run RAWR web server, run the following command in **rawr-web** folder and open the website link (http://10.0.2.15:5000/) with any browser.

```
python app.py

#You should see this in the terminal:
* Running on http://10.0.2.15:5000/
# Click to open the link with a browser
```

Local Galaxy Install

Galaxy is available for Linux and macOS. This tutorial has been tested on Linux macOS Sequoia 15.0 and Ubuntu 22.04 LTS.

Please make sure you have set **python** to link to Python3.

If you are using macOS, please change your terminal shell to Bash.

If you're using Windows, please install Windows Subsystem for Linux and reboot prior to setting up Galaxy. If you have access to Hyper-V, you can alternatively create a Linux virtual machine.

Navigate to **rawr-galaxy** folder and download galaxy following instructions from official website: https://galaxyproject.org/admin/get-galaxy/

Folder structure after Galaxy is set up properly:

```
rawr-galaxy/
├── galaxy/         # download from https://galaxyproject.org/admin/get-galaxy/
├── conda_stuff/
├── dist/
├── rawr_src/
│   ├── mafft/
│   └── raxmlHPC/
```

Three steps to set up Galaxy:

1) compile RAWR
2) install Planemo and set up Galaxy
3) use Planemo to serve Galaxy locally (after initial setup, only use this step)

Compile RAWR

This tutorial used Python v3.10.15, but feel free to use a newer version.

1. navigate to **rawr-galaxy** folder, create rawr environment, and compile RAWR binary executable.

   if you would prefer, we provide precompiled RAWR tarballs at https://gitlab.msu.edu/liulab/rawr-web-software/-/tree/master/rawr-galaxy.

   Virtualenv v20.26.6 commands:

```
cd rawr-galaxy
python -m venv .rawr
. .rawr/bin/activate

pip install Bio>=1.84
pip install "python-dateutil>=2.9.0"
pip install "ete3>=3.1.3"
pip install "numpy>=2.1.2"
pip install "scipy>=1.14.1"
pip install "sklearn>=1.5.2"
pip install "pandas>=2.2.3"
pip install "pyinstaller>=6.10.0"

pyinstaller --hidden-import="sklearn.utils._cython_blas" --hidden-
import="sklearn.neighbors.typedefs" rawr.py

deactivate
```

   Conda v22.9.0 commands:

```
cd rawr-galaxy

conda create -n rawr python=3.10
conda activate rawr

conda config --show channels
conda config --add channels conda-forge
conda config --add channels bioconda
conda config --remove channels defaults

conda install -c conda-forge -c etetoolkit ete3 numpy scipy scikit-learn biopython pandas
pyinstaller

pyinstaller --hidden-import="sklearn.utils._cython_blas" --hidden-
import="sklearn.neighbors.typedefs" rawr.py

conda deactivate
```

2. modify the **rawr-galaxy/rawr.xml** file on line 3 with full path to your rawr executable. To find the full path in, navigate to **dist/rawr/** and type **pwd** into your terminal.

```
1 <tool id="rawr" name="RAWR random sampler" version="1.0"
python_template_version="3.5">
2   <command detect_errors="exit_code"><![CDATA[
3    /full/path/to/dist/rawr/rawr
```

3. copy over MAFFT and RAxML folders from **rawr_src**

```
cp -r rawr_src/mafft* dist/rawr/_internal/
cp -r rawr_src/raxml* dist/rawr/_internal/
```

4. double check that rawr is working

```
/full/path/to/dist/rawr/rawr
#returns:
usage: rawr [-h] [--algorithm [ALGORITHM]] [--task [TASK]] [--path [PATH]] [--n [N]] [--rate
[RATE]] [--numanchor [NUMANCHOR]]
          [--lenanchor [LENANCHOR]] [-seed SEED] [--mafft MAFFT] [--raxml RAXML] [--verbose]
          alnfile treefile
rawr: error: the following arguments are required: alnfile, treefile

# small example
/full/path/to/dist/rawr/rawr example/dataset-10taxa/alignment.fasta example/dataset-
10taxa/infer.tree --task tree --algorithm rawr --n 2 --rate 0.1
```

Setting up Galaxy Virtualenv

This Virtualenv tutorial is for installing Planemo and Galaxy. We used Python v3.10.15 in Virtualenv v20.26.6 with Galaxy v24.1.2.

1. download Galaxy from official website: https://galaxyproject.org/admin/get-galaxy/ This creates the **galaxy** folder.
2. navigate to **rawr-galaxy/galaxy** and create virtualenv

```
cd galaxy
python -m venv .venv
. .venv/bin/activate
```

2. install Galaxy dependencies. If you run into trouble, look at the debugging galaxy section.

```
PYTHONPATH= sh scripts/common_startup.sh --no-create-venv
```

Package versions used in tutorial:

```
virtualenv          20.26.6
python              3.10.15
pip                 22.0.2
planemo             0.75.26
galaxy-tool-util    24.1.2
galaxy-util         24.1.2
cwltool             3.1.20241007082533
yarn                1.22.22
```

3. locate the modified **rawr.xml** file. Make sure it's in **rawr-galaxy** folder (or wherever you call planemo from, make sure rawr.xml is in the same folder)

```
cd ..
ls      # rawr-galaxy folder
you should see the rawr.xml file you edited earlier
dist  example  galaxy  rawr.py  rawr.spec  rawr_src  rawr.xml
```

4. install planemo and serve Galaxy

```
pip install planemo
planemo serve --galaxy_root galaxy
```

If you run into trouble with installation, look at the **Debugging Galaxy** section.

Setting up Galaxy Conda

This Conda tutorial teaches you how to install Planemo, set up Galaxy, and create RAWR conda package. This was tested with Python v3.10.15 in Anaconda v24.5.0 with Galaxy v24.1.2.

Note: you can set up Galaxy Conda in one of two ways:

I. manually, as described in the instructions below. This is the recommended method.

II. allow Planemo to handle Galaxy set up, it will pull from the latest github repository. If you wish to let Planemo do it, simply install planemo, complete step 1, and skip to running galaxy conda.

Manually install Galaxy Conda (recommended)

1. download Galaxy from official website: https://galaxyproject.org/admin/get-galaxy/ This creates the **galaxy** folder.
2. create local conda package for RAWR

```
conda activate rawr
cp -r conda_stuff/* dist
cd dist
```

Note: folder **conda_stuff** contains a different rawr.xml file intended for Conda.

Modify **dist/rawr.xml** on line 6 with full path to your rawr executable. To find the full path, navigate to **dist/rawr/** and type **pwd** into your terminal.

```
<tool id="rawr" name="RAWR random sampler" version="1.0"
python_template_version="3.5">
  <requirements>
     <requirement type="package" version="1.0">rawr</requirement>
  </requirements>
  <command detect_errors="exit_code"><![CDATA[
   /full/path/to/dist/rawr/rawr
```

build conda package

```
conda install conda-build
conda build conda-recipe
conda deactivate
```

2. ensure your environment has the correct version of python you want to use with Galaxy (code has been tested on Python 3.8 as well, and you can try newer Python versions).

```
python --version
Python 3.10.15
```

3. navigate to the "galaxy" folder and start up galaxy with "sh run.sh".

```
cd galaxy
sh run.sh
```

4. The Galaxy server will then be set up for the first time, followed immediately by starting the Galaxy server. If you run into issues, help is available in the **Debugging Galaxy** section.

   If you see the following services starting, Galaxy has finished set up and is trying to start the server. It is safe to stop the process here.

```
celery                    STARTING
celery-beat               STARTING
gunicorn                  STARTING
```

   Or if you see this line, Galaxy server is already up and running, indicating both server set up and launch has completed successfully. It is safe to stop the process here.

```
Serving on http://127.0.0.1:8080
```

   You can close the Galaxy server process.

```
# Break the server process with (macOS) command+. or (Linux) ctrl+c
```

   During setting up Galaxy for the first time, a _galaxy_ conda environment will be created automatically for you.

Package versions used in tutorial for Planemo environment **_galaxy_** (this environment is created for you when setting up Galaxy with **run.sh**).

```
python              3.10.15
galaxy-tool-util    24.1.2
galaxy-util         24.1.2
planemo             0.75.26
cwltool             3.1.20240708091337
stdlib-list         0.10.0
virtualenv          20.26.6
yarn                1.22.22
```

5. install Planemo for Galaxy

```
conda deactivate
conda activate _galaxy_   # created for us by Galaxy

conda config --show channels
conda config --add channels conda-forge
conda config --add channels bioconda
conda config --remove channels defaults

conda install -c bioconda -c conda-forge cwltool stdlib-list planemo
```

6. use Planemo to serve Galaxy

```
cd ../dist       # where rawr.xml is located
planemo serve --conda_use_local --galaxy_root ../galaxy

#You should see this line:
Serving on http://127.0.0.1:9090
# Click to open the link with a browser
```

If the installation does not go smoothly, some tips are available in the **Debugging Galaxy** section.

Debugging Galaxy

1. Could not solve environment:

```
Could not solve for environment specs
The following packages are incompatible
```

During Galaxy server set up, you may run into problems where the environment cannot be solved. Check that your environment has the same or higher package versions as tutorial.

```
pip list | grep galaxy-tool-util                    #Virtualenv command
```

```
conda list | grep galaxy-tool-util                  # Conda command
```

2. **Socket error:**

```
Error: Cannot open an HTTP server: socket.error reported AF_UNIX path too long
```

This error occurs when setting up Galaxy server for the first time. It means your filepath is too long for sockets, which are limited to 104 characters. Just move the folder containing galaxy to a directory with shorter path. For example:

```
cp -r rawr-galaxy ~/
```

Then navigate to inside the galaxy folder and try setting up the server.

```
cd ~/rawr-galaxy/galaxy
sh run.sh
```

3. **Error 137:**

```
Browserslist: caniuse-lite is outdated. Please run:
  npx update-browserslist-db@latest
Killed
error Command failed with exit code 137.
```

Code 137 is usually an indication of out of memory. Building the Galaxy client is very memory intensive. Try separating the server set up from the client set up.

Navigate inside Galaxy folder and build the server with the following command:

```
GALAXY_SKIP_CLIENT_BUILD=1 sh run.sh
```

Once you see the following text, you know that the Galaxy server is being initiated. You can stop the process and proceed to build the client.

```
celery                    STARTING
celery-beat               STARTING
gunicorn                  STARTING
```

Ensure you have yarn v1 (at the time of this writing, Galaxy depends on yarn v1)

```
npm install -g yarn@1
```

Then build Galaxy client:

```
make client
```

(Optional) if you're still running into code 137, try editing the Makefile to allow Galaxy to use more memory. Increase the number after "max-old-space-size=". In tests on Windows with Linux virtual machine, setting this parameter to 10 GB gave enough resources to Galaxy to build client successfully.

```
max-old-space-size=10240 # MB. This lets Galaxy use up to 10 GB of RAM, which is definitely more than enough.
```

4. **Yarn error:**

```
Usage: yarn [options]
yarn: error: no such option: --network-timeout
```

During Galaxy set up, you may randomly run into yarn issue. Double check your yarn version is correct. Yarn v1 is what Galaxy requires at the time of this writing.

```
npm install -g yarn@1
yarn --version
sh run.sh                    # retry setting up Galaxy
```

5. Galaxy server fails to launch on missing tool id:

```
Exception: Missing tool 'id' for tool at 'XmlToolSource[/path/to/rawr-
galaxy/galaxy/packages/app/galaxy/tools/bundled/expression_tools/expression_macros.xml]'
```

If you see errors about missing id, it's likely that Planemo was launched from an unexpected folder. Planemo must be launched in the folder containing the correct version of rawr.xml.

- Conda Galaxy: navigate to **rawr-galaxy/dist**, ensure **rawr.xml** is present, and call **planemo serve --galaxy_root ../galaxy**
- Virtualenv Galaxy: navigate to **rawr-galaxy**, ensure **rawr.xml** is present, and call **planemo serve --galaxy_root ../galaxy**

6. Virtualenv Galaxy -- Unclickable RAWR:

This is for using Galaxy with Virtualenv. If everything looks fine but you can't use RAWR (and you see error messages about package dependencies or **h11._util.LocalProtocolError: Too much data for declared Content-Length**), you may be using the wrong version of rawr.xml.

Make sure the file **rawr-galaxy/rawr.xml** does **not** have the following <requirements> code on lines 2 – 4. If you see these three lines in rawr-galaxy/rawr.xml, delete them.

```
2   <requirements>
3       <requirement type="package" version="1.0">rawr</requirement>
4   </requirements>
```

7. Conda Galaxy -- Unclickable RAWR:

This is for using Galaxy with Conda. If everything looks like it's working but you can't click on RAWR inside of Galaxy (plus error messages with loading tools in the terminal), there may be an issue with automatically creating a conda environment for RAWR. Make sure of the following:

- **rawr-galaxy/dist/rawr.xml** file has the correct path to RAWR
- **rawr-galaxy/dist** is the folder you're inside when calling Planemo
- **_galaxy_** has most updated version of **conda-build**
- **/path/to/anaconda3/conda-bld/rawr-1.0-0.tar.bz2 exists** -- this is the result of building compiled RAWR into a conda package.

MAFFT and RAxML versions

RAWR source comes with standalone versions of MAFFT v7.487, RAxML v8.2.12 for macOS and Linux, and RAxML v8.2.10 for Windows.

If you want updated versions of these software, you can download or compile the standalone versions from their respective websites and update their respective folders inside **src** folder (please make sure to double check that the binary filenames remain the same).

- MAFFT: https://mafft.cbrc.jp/alignment/software/.
- RAxML: https://github.com/stamatak/standard-RAxML.

Running RAWR v1.0

GUI software

```
cd rawr-software
python main.py
```

web server

```
cd rawr-web
python app.py

#You should see this in the terminal:
* Running on http://10.0.2.15:5000/
# Click to open the link with a browser
```

Galaxy Virtualenv

In the future, you do not need to set up again. Simply do the following:

```
cd rawr-galaxy
. galaxy/.venv/bin/activate
planemo serve --galaxy_root galaxy

#You should see this line:
Serving on http://127.0.0.1:9090
# Click to open the link with a browser
```

Galaxy Conda

In the future, you do not need to set up again. Simply do the following:

```
cd rawr-galaxy/dist
conda activate _galaxy_
planemo serve --conda_use_local --galaxy_root ../galaxy

#You should see this line:
Serving on http://127.0.0.1:9090
# Click to open the link with a browser
```

Input

If you are running RAWR for MSA estimation, please prepare a multiple alignment (MSA) file in FASTA format. If you are running RAWR for phylogeny alignment estimation, please prepare an MSA file and a phylogeny in Newick tree format.

In this tutorial, we will not go over preparing the inputs. Instead, we provide a small dataset to try out RAWR. The example dataset has been prepared from sequence files and aligned with MAFFT. Then, we used the alignment file as input into RAxML and reconstructed a phylogeny under the General Time Reversal (GTR) model.

1) Find the example datasets in your RAWR directory under the examples folder. Here, you should see dataset-10taxa.

2) Select either RAWR or SERES as resampling algorithm.

3) Select either multiple sequence alignment (MSA) estimation or phylogenetic tree estimation.

4) Optionally, you can edit the default parameter values.

   By default:
   Sample Number: 10 (minimum 2. Represents total number of sampled sequences.)
   Reverse Rate: 0.1 (0 to 1 not including 0 or 1. Represents the reverse rate of the random walk.)
   Anchor Number: 20 (SERES only. Represents the number of anchors.)
   Anchor Length: 5 (SERES only. Represents the length of the anchors.)

5) Select a file input for the FASTA alignment file. In our example datasets, choose **alignment.fasta**

6) Select a file input for the Newick phylogeny tree file. In out example dataset, choose **infer.tree.**

7) Click **OK** or **Submit** and let the program run to completion.

   Please note that if you are using the GUI software version of RAWR v1.0, it is best to provide an empty results folder to avoid any problems with RAxML refusing to run due to previous runs' files present or improper software termination.

Output

Sampled sequence output

The Sequence sampler will create a tar compressed directory called **samples**. The sampled sequences and indices can be found here.

*.seq.fasta: the unaligned sampled sequence in FASTA format.
*.index: this file contains a series of column indices of the input alignment. The column indices represent sampled columns of input alignment in order.

MSA support estimation output

The MSA support estimator will calculate the support value for input alignment and save the support value to MSA.support.csv.

MSA.support.csv contains four columns. The first column locates the position of a residue pair. The second column and third column are the row index of two residues in this residue pair. Only residue pairs with no gaps are contained in this result. The last column is the support value, which ranges from 0 to 1.

MSA.support.csv_jalview_annotation.txt contains a JalView annotation file for visualization. (Note: the local Galaxy client is implemented such that running MSA support estimation with RAWR v1.0 will only output JalView annotation file.)

Tree support estimation output

The tree support estimator will generate annotated tree in Newick format, which is saved to **tree.support.txt**. It will also generate a figure called **tree.support.png** to visualize the tree structure and support values.

Visualize Output

MSA support visualizer

The MSA support estimator will calculate the support value for input alignment and save the support value to **MSA.support.csv** and provide a feature annotation file **MSA.support.csv_jalview_annotation.txt** which is a JalView annotation file for visualization.

The following graphic tutorial shows you how to view the JalView annotations. We used JalView 2.11.4.0 in the following example.

1. Click “File” > “Input Alignment” > “From File” to enter your original “alignment.fasta” input file. This will open an alignment viewer inside of JalView.

3. Inside the alignment viewer, click “File” > “Load Features/ Annotations” (see Figure 1). Figure 2 shows how JalView will visualize your alignment file. The support bar graph color can be changed with our custom script.

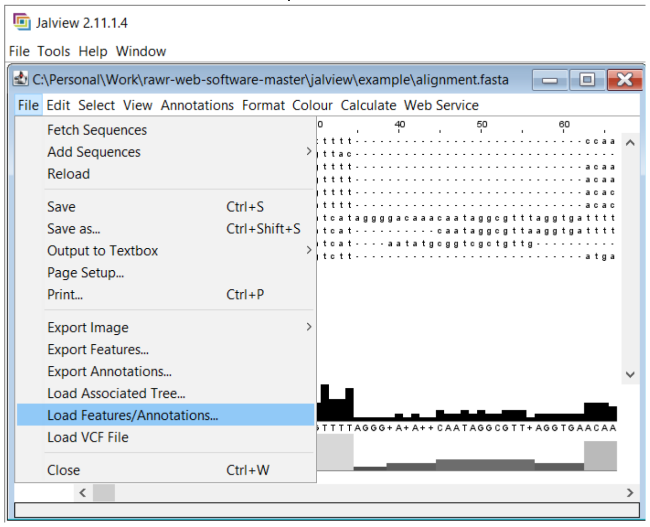
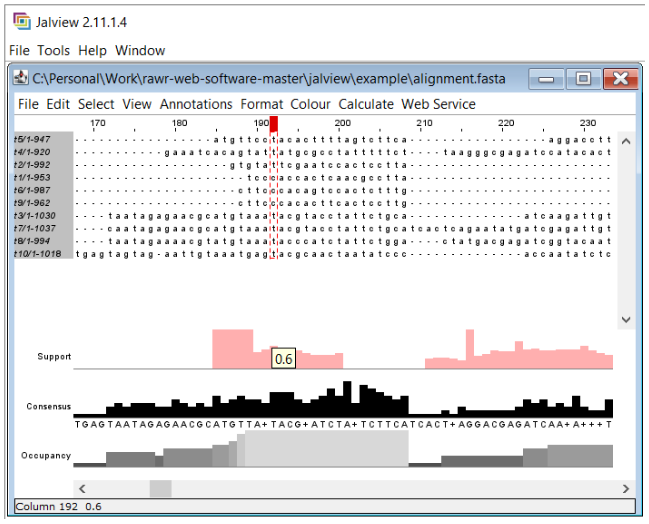


Figure 1: JalView alignment viewer. After opening the MSA file, load in the sequence features/annotations file generated from our custom script.


Jalview 2.11.1.4
File Tools Help Window
C:\Personal\Work\rawr-web-software-master\jalview\example\alignment.fasta
File Edit Select View Annotations Format Colour Calculate Web Service
Support
0.6
Consensus
Occupancy
Column 192 0.6

```
             170       180       190       200       210       220       230
t5/1-947     ----------------atgttcctacactttttagtcttca-----------------aggacctt
t4/1-920     -----------gaaatcacagtattatgcgcctatttttct----taagggcgagatccatacact
t2/1-992     ------------------gtgtattcgaatccactcctta-------------------------
t1/1-953     ---------------------tcccaccactcaacgcctta-------------------------
t6/1-987     --------------------cttccccacagtccactctttg------------------------
t9/1-962     --------------------cttccccacacttcactccttg------------------------
t3/1-1030    ----taatagagaacgcatgtaaatacgtacctattctgca--------------atcaagattgt
t7/1-1037    ----caatagagaacgcatgtaaatacgtacctattctgcatcactcagaatatgatcgagattgt
t8/1-994     ----taatagaaaacgtatgtaaatacccatctattctgga----ctatgacgagatcggtacaat
t10/1-1018   tgagtagtag-aattgtaaatgagtacgcaactaatatccc--------------accaatatctc
Consensus    TGAGTAATAGAGAACGCATGTTA+TACG+ATCTA+TCTTCATCACT+AGGACGAGATCAA+A+++T
```

Figure 2: JalView MSA support visualization.

Tree support visualizer

We provide an image of the Newick tree with confidence support (see Figure 3). Tree visualization is integrated into RAWR v1.0 GUI software, local Galaxy client, and web application. You can also choose other phylogeny visualizers such as Interactive Tree Of Life (iTOL).

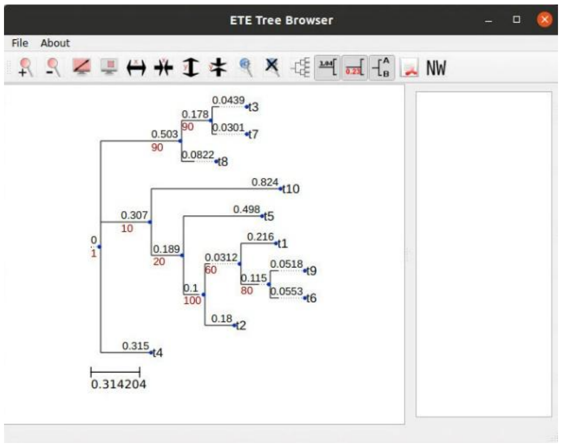


Figure 3: Newick tree with RAWR support values visualized in RAWR v1.0 GUI desktop client.

Picture Tutorials

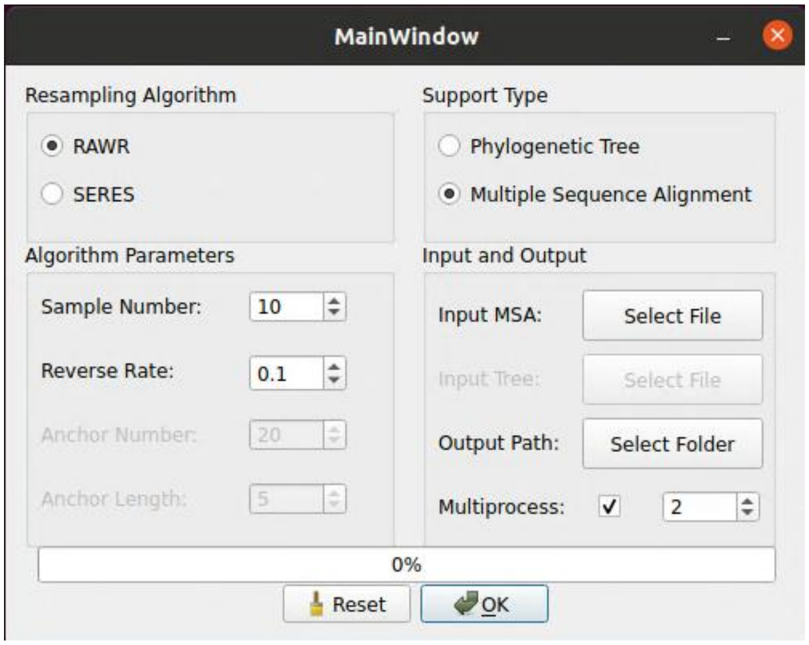
Software, RAWR algorithm, Multiple Sequence Alignment Support

Suppose that I choose to use RAWR algorithm to resample my multiple species alignment file.

To start, I enter the directory of RAWR software and launch a terminal.

Submitting command “python main.py” in this folder starts my application, seen in Figure 4.

Then, I choose the Resampling Algorithm “RAWR”, Support Type “Multiple Sequence Alignment”, select MSA file “trueAlignment.fasta”, and create a new folder called “test”. Optionally, click “Multiprocess” checkbox to run parallel version of the software. Click “OK” to start the run.



Resampling Algorithm: [✓ RAWR] SERES

Support Type: Phylogenetic Tree [✓ Multiple Sequence Alignment]

Sample Number: 10

Reverse Rate: 0.1

Anchor Number: 20

Anchor Length: 5

Input MSA:

Input Tree:

Output Path:

Multiprocess: ✓ 2

Figure 4: Software GUI when running RAWR algorithm to estimate support for a given multiple sequence alignment.

The results are in my output folder “test”, where the “samples” subfolder contains the resampling done by RAWR and the parent directory contains the “MSA.support.csv” file and the “MSA_Support.csv_jalview_annotation.txt” file. The JalView annotation file displays the support estimation for each alignment site.

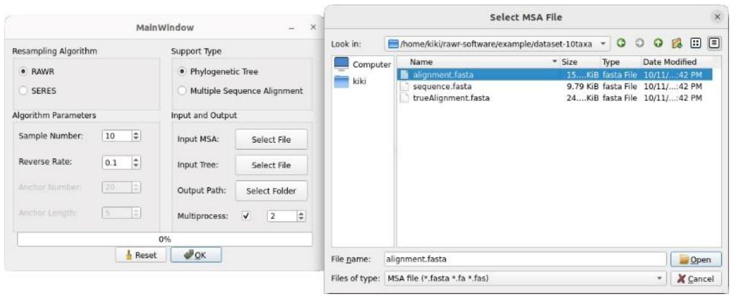
Software, RAWR algorithm, Phylogenetic Tree Support

This example supposes I want to use RAWR algorithm to resample my multiple species alignment file and produce phylogenetic tree support values.

To begin, enter the folder containing RAWR software and launch a terminal from this directory.

Type and enter “python main.py” in the terminal to start the application.

In Figures 5 and 6, I choose the Resampling Algorithm “RAWR”, Support Type “Phylogenetic Tree”, select my MSA file “trueAlignment.fasta” for Input MSA, select my tree file “infer.tree” for Input Tree, and create a new folder called “testtree”. Optionally, click the Multiprocess checkbox to run the parallel version of the software. Click “OK” to start the run. Figure 7 shows the output tree with support values.


MainWindow

Resampling Algorithm: (●) RAWR ( ) SERES

Support Type: (●) Phylogenetic Tree ( ) Multiple Sequence Alignment

Algorithm Parameters
Sample Number: 10
Reverse Rate: 0.1
Anchor Number: 20
Anchor Length: 5

Input and Output
Input MSA: Select File
Input Tree: Select File
Output Path: Select Folder
Multiprocess: ✓ 2

0%

Reset OK

Select MSA File

Look in: /home/kiki/rawr-software/example/dataset-10taxa

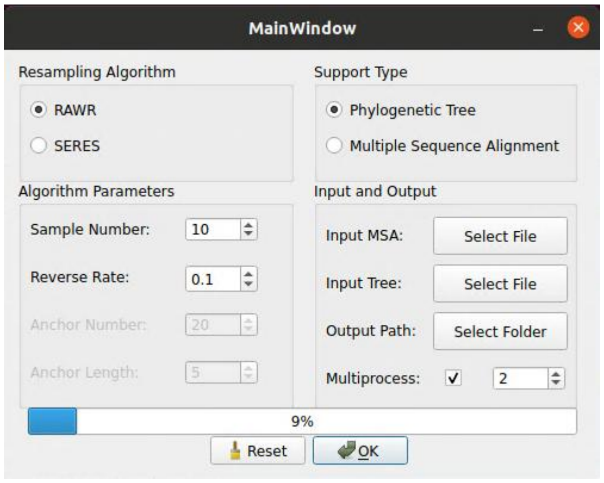
Computer
kiki

Name	Size	Type	Date Modified
alignment.fasta	15....KiB	fasta File	10/11/...:42 PM
sequence.fasta	9.79 KiB	fasta File	10/11/...:42 PM
trueAlignment.fasta	24....KiB	fasta File	10/11/...:42 PM

File name: alignment.fasta
Files of type: MSA file (*.fasta *.fa *.fas)

Open Cancel

Figure 5: Select input files and creating output folders in GUI software.


MainWindow

Resampling Algorithm: (●) RAWR ( ) SERES

Support Type: (●) Phylogenetic Tree ( ) Multiple Sequence Alignment

Algorithm Parameters
Sample Number: 10
Reverse Rate: 0.1
Anchor Number: 20
Anchor Length: 5

Input and Output
Input MSA: Select File
Input Tree: Select File
Output Path: Select Folder
Multiprocess: ✓ 2

9%

Reset OK

Figure 6: Gui software RAWR algorithm resampling for phylogenetic tree support estimation.

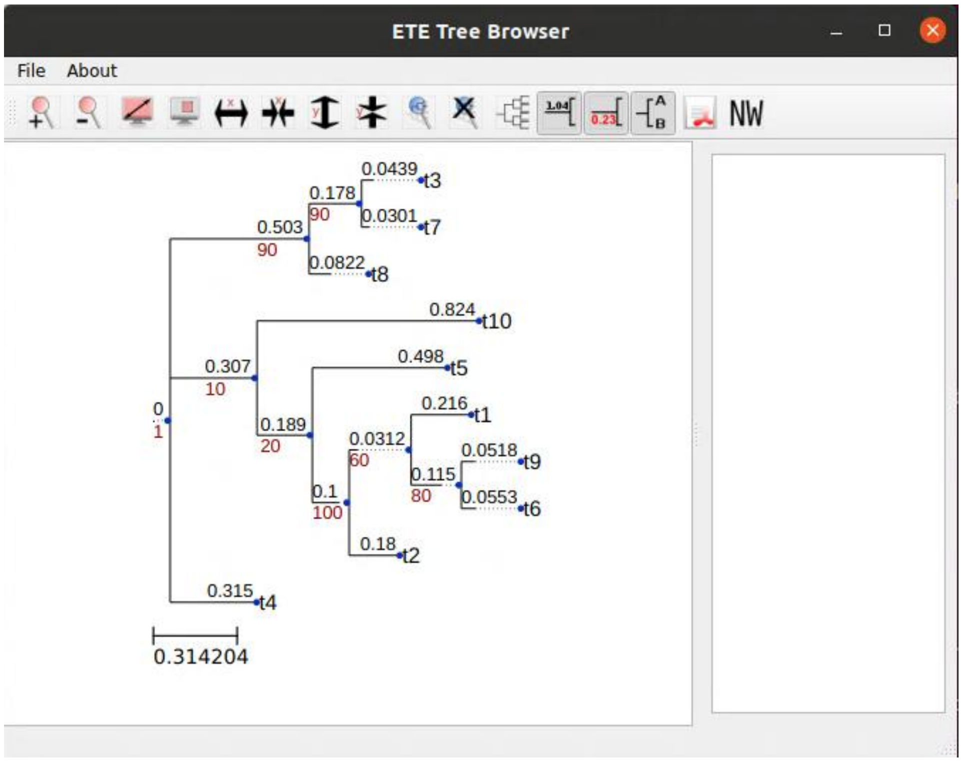


Figure 7: Software output showing a phylogeny with support values, the result of using RAWR resampling with phylogenetic tree support estimation options.

The results are in my output folder “testreet”, where the “samples” subfolder contains the resampling done by RAWR and the parent directory contains the phylogeny file in Newick format.

Website, SERES algorithm, Phylogenetic Tree Support

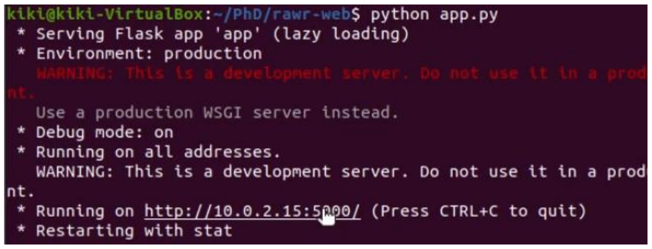
Suppose that I want to give my lab mates the ability to send jobs through a computer dedicated to hosting RAWR web server and receive their results by email. Below is an example showing how to launch the RAWR web server after proper installation and to use SERES algorithm to resample and provide support for an input phylogeny.

Go to the directory of RAWR web server and launch a terminal.

Type and enter “python app.py” and look for the correct terminal messages to pop up, as seen in Figure 8. The internal network website link can be verified with the terminal output.

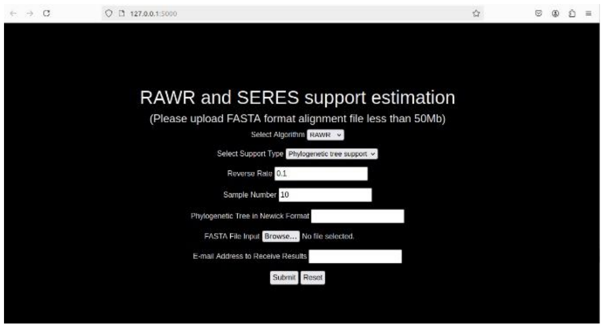
The user can open the website link on the dedicated computer through any browser and see the display in Figure 9.

In Figure 10 and 11, I choose the Resampling Algorithm “SERES”, Support Type “Phylogenetic Tree”, alter the Algorithm Parameters “Sample Number: 2”, input my MSA file “alignment.fasta” into Input MSA, copy-paste the contents of “infer.tree” into Input Tree parameter, and enter an email address to receive the alert when results are ready. When all the values are correct, I click “Submit” to confirm the job submission.



```
kiki@kiki-VirtualBox:~/PhD/rawr-web$ python app.py
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a prod
nt.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a prod
nt.
 * Running on http://10.0.2.15:5000/ (Press CTRL+C to quit)
 * Restarting with stat
```

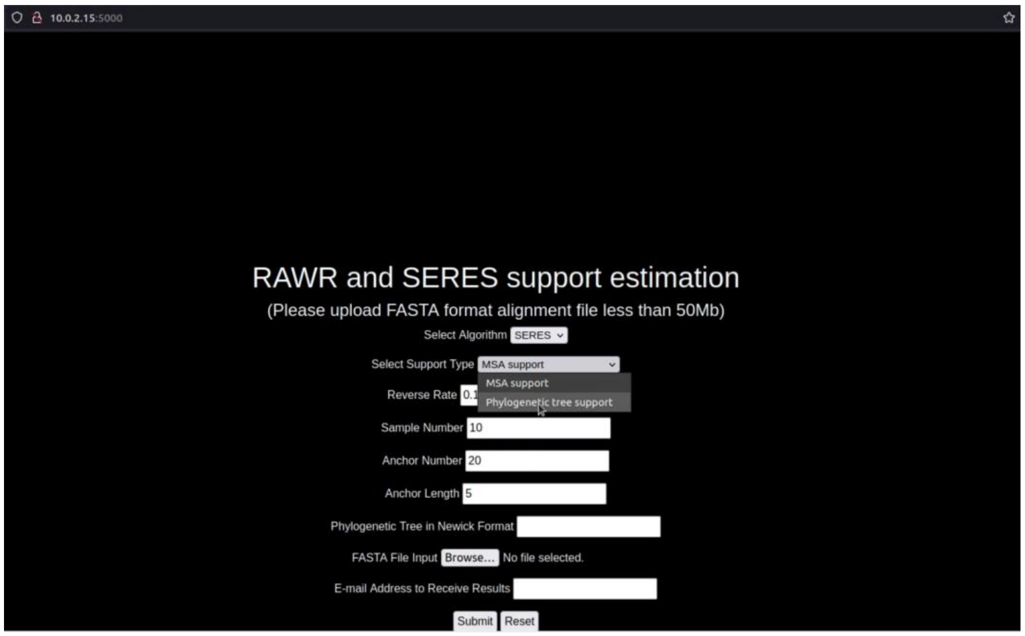
Figure 8: RAWR web server correct launch terminal output. Where the mouse is located shows the link that the intranet website can be found on.



127.0.0.1:5000

RAWR and SERES support estimation
(Please upload FASTA format alignment file less than 50Mb)
Select Algorithm: RAWR
Select Support Type: Phylogenetic tree support
Reverse Rate: 0.1
Sample Number: 10
Phylogenetic Tree in Newick Format:
FASTA File Input: Browse... No file selected.
E-mail Address to Receive Results:
Submit Reset

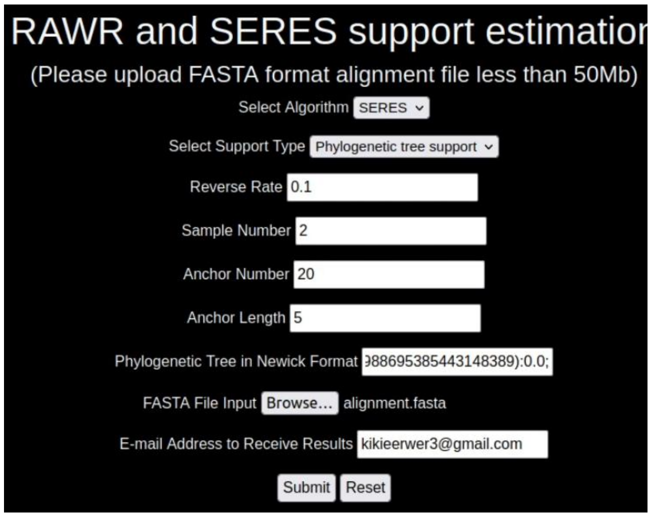
Figure 9: Default RAWR internal website running on a FireFox browser.



10.0.2.15:5000

RAWR and SERES support estimation
(Please upload FASTA format alignment file less than 50Mb)
Select Algorithm: SERES
Select Support Type: MSA support
MSA support
Phylogenetic tree support
Reverse Rate: 0.1
Sample Number: 10
Anchor Number: 20
Anchor Length: 5
Phylogenetic Tree in Newick Format:
FASTA File Input: Browse... No file selected.
E-mail Address to Receive Results:
Submit Reset

Figure 10: Website has drop-down menu to change between algorithms.



RAWR and SERES support estimatio

(Please upload FASTA format alignment file less than 50Mb)

Select Algorithm: SERES

Select Support Type: Phylogenetic tree support

Reverse Rate: 0.1

Sample Number: 2

Anchor Number: 20

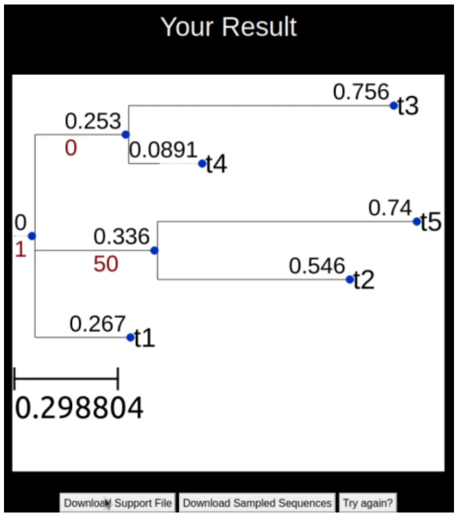
Anchor Length: 5

Phylogenetic Tree in Newick Format: 988695385443148389):0.0;

FASTA File Input: Browse... alignment.fasta

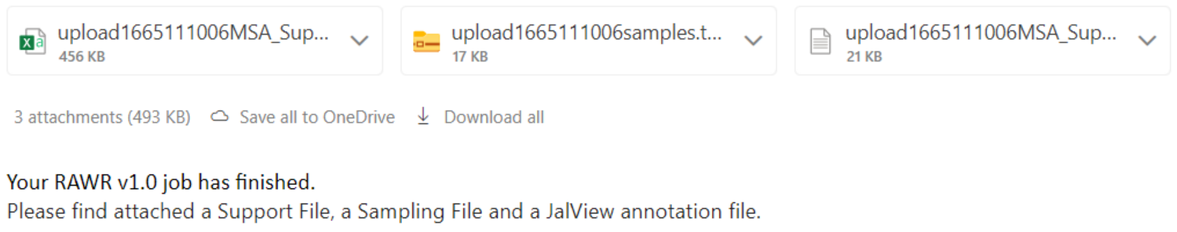
E-mail Address to Receive Results: kikieerwer3@gmail.com

Figure 11: Website running SERES resampling for phylogeny tree support estimation and all parameter values filled in.



Your Result

Figure 12: Website redirects to results page once processing has finished. The phylogeny with confidence values is presented in .png format. The resampling files and Newick formatted tree file have downloadable links that work from the same computer.



upload1665111006MSA_Sup...
456 KB

upload1665111006samples.t...
17 KB

upload1665111006MSA_Sup...
21 KB

3 attachments (493 KB)   Save all to OneDrive   Download all

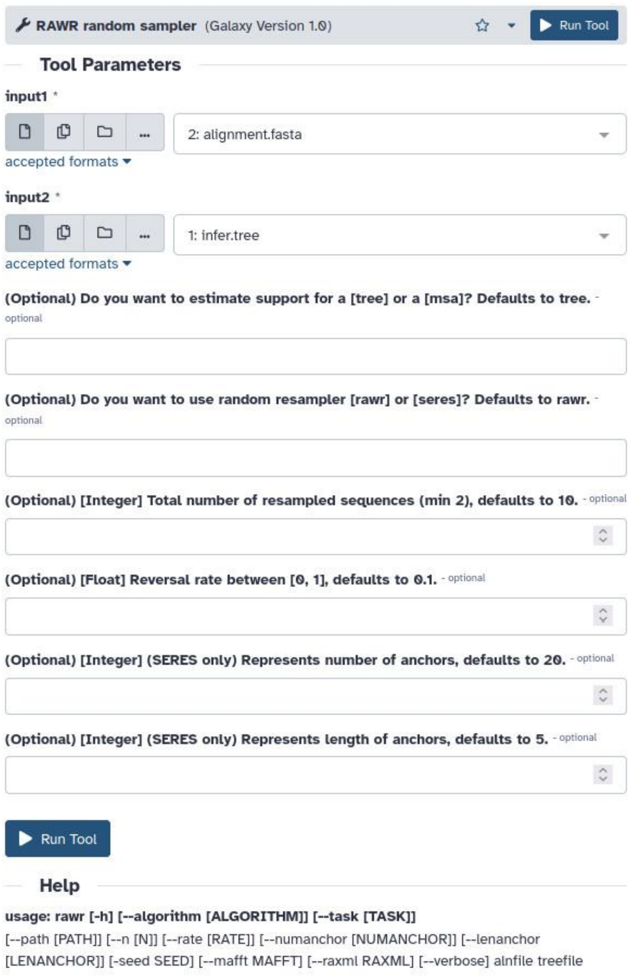
Your RAWR v1.0 job has finished.
Please find attached a Support File, a Sampling File and a JalView annotation file.

Figure 13: Emailed results to user when processing is finished with downloadable links (from the same computer).

Figures 12 and 13 show the results in the redirected webpage and the email alert sent to the user, respectively. For phylogenetic tree support estimation, the support file is in Newick tree format, the SERES resampled files is in a tarball, and the visualized phylogeny is available for download.

Local Galaxy instance, RAWR algorithm, phylogenetic tree support

After launching the local Galaxy instance (see install instructions), navigate to RAWR random sampler on the left toolbar. You should see Figure 14 appear in the center of webpage.


RAWR random sampler (Galaxy Version 1.0)

Tool Parameters

input1 *

2: alignment.fasta

input2 *

1: infer.tree

(Optional) Do you want to estimate support for a [tree] or a [msa]? Defaults to tree. - optional

(Optional) Do you want to use random resampler [rawr] or [seres]? Defaults to rawr. - optional

(Optional) [Integer] Total number of resampled sequences (min 2), defaults to 10. - optional

(Optional) [Float] Reversal rate between [0, 1], defaults to 0.1. - optional

(Optional) [Integer] (SERES only) Represents number of anchors, defaults to 20. - optional

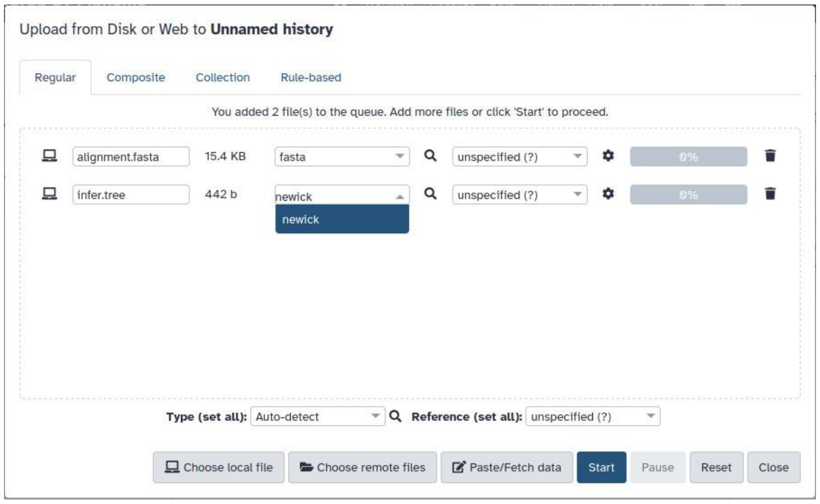
(Optional) [Integer] (SERES only) Represents length of anchors, defaults to 5. - optional

Help

```
usage: rawr [-h] [--algorithm [ALGORITHM]] [--task [TASK]]
[--path [PATH]] [--n [N]] [--rate [RATE]] [--numanchor [NUMANCHOR]] [--lenanchor
[LENANCHOR]] [-seed SEED] [--mafft MAFFT] [--raxml RAXML] [--verbose] alnfile treefile
```

Figure 14: Local Galaxy running RAWR random sampler.

Click on the **Upload File** button on the lefthand side to select input1 and input2, which are alignment file and phylogeny file respectively. Make sure the file types are correct, **fasta** and **newick,** respectively. See Figure 15.


Upload from Disk or Web to **Unnamed history**

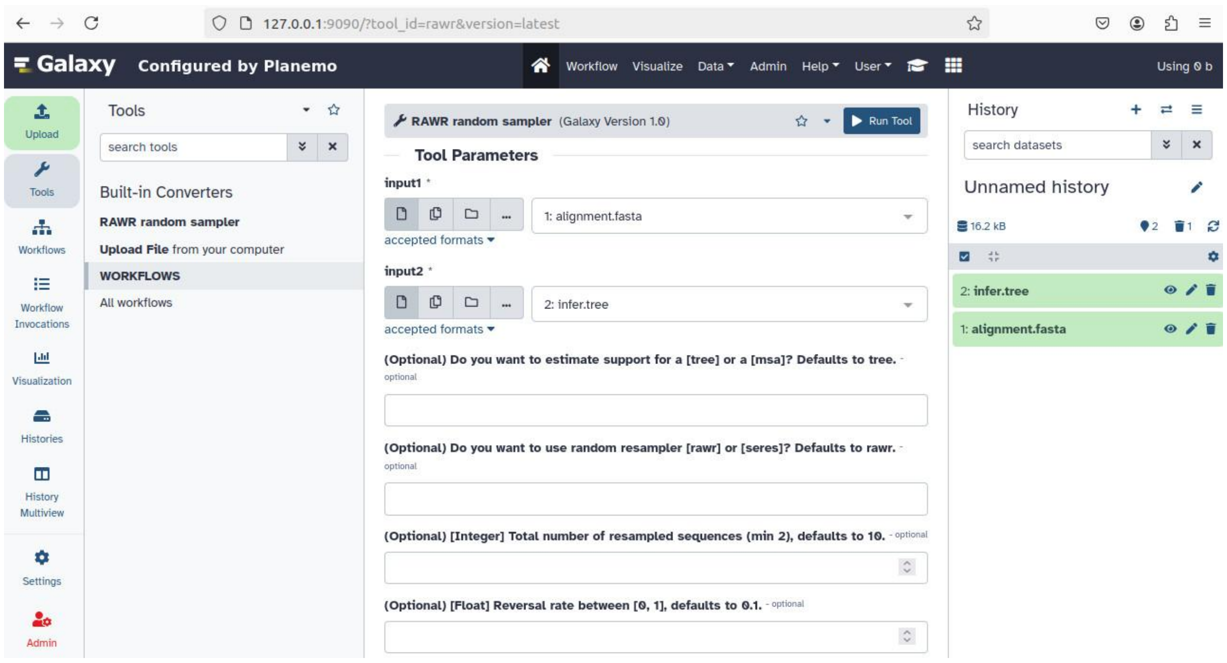
You added 2 file(s) to the queue. Add more files or click 'Start' to proceed.

alignment.fasta	15.4 KB	fasta	unspecified (?)	0%
infer.tree	442 b	newick	unspecified (?)	0%

Type (set all): Auto-detect **Reference (set all):** unspecified (?)

Figure 15: Upload files, specifying file types.

Navigate back to RAWR random sampler after uploading files. See Figure 16. Optionally, there are parameters you can adjust or leave blank. Hit **Run Tool** to execute RAWR v1.0.


127.0.0.1:9090/?tool_id=rawr&version=latest

RAWR random sampler (Galaxy Version 1.0)

Tool Parameters

input1 *: 1: alignment.fasta

accepted formats

input2 *: 2: infer.tree

accepted formats

(Optional) Do you want to estimate support for a [tree] or a [msa]? Defaults to tree. - optional

(Optional) Do you want to use random resampler [rawr] or [seres]? Defaults to rawr. - optional

(Optional) [Integer] Total number of resampled sequences (min 2), defaults to 10. - optional

(Optional) [Float] Reversal rate between [0, 1], defaults to 0.1. - optional

Unnamed history

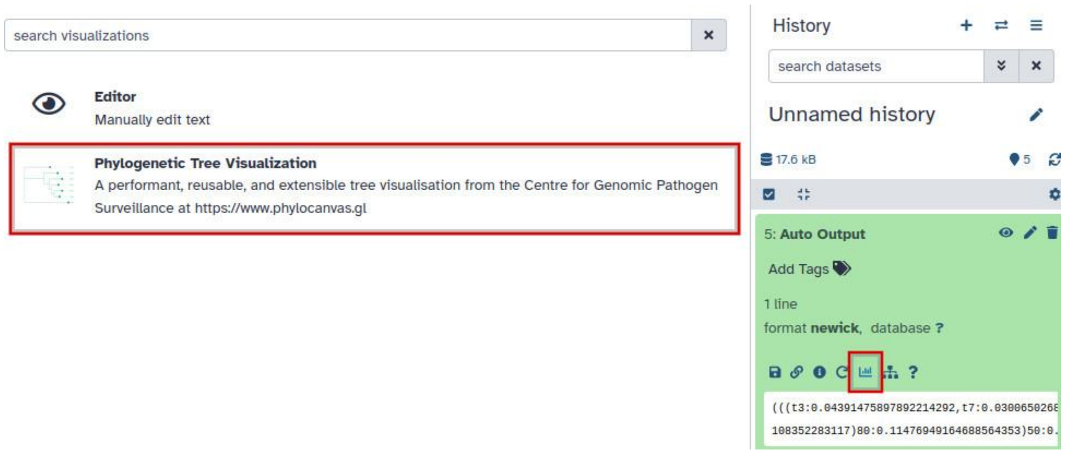
16.2 kB 2 1

2: infer.tree

1: alignment.fasta

Figure 16: After uploading files, navigate back to RAWR random sampler

Click Execute to run the algorithm. The result will be called Auto Output. Clicking visualize icon (see Figure 17) gives you a few options.

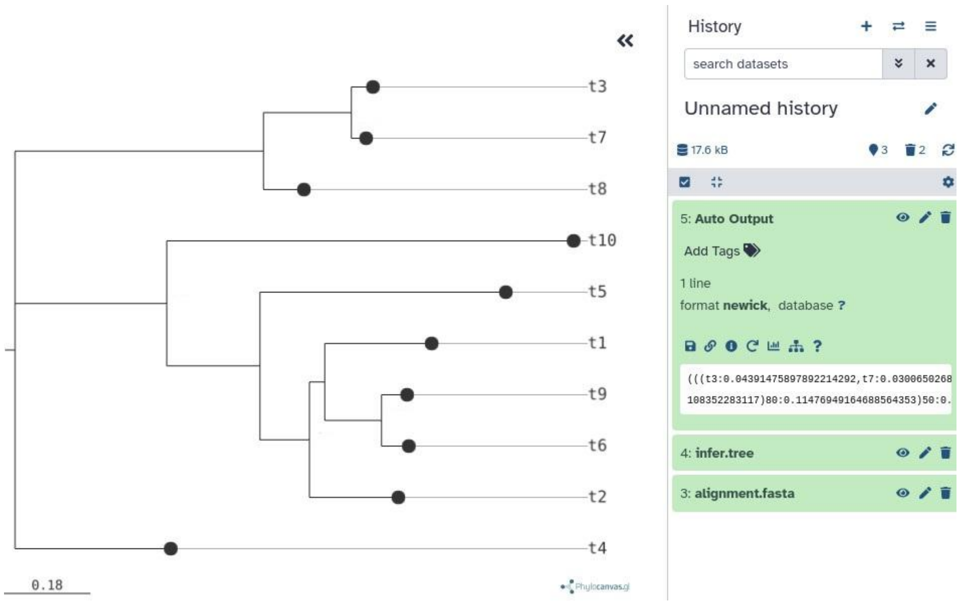

search visualizations
Editor
Manually edit text
Phylogenetic Tree Visualization
A performant, reusable, and extensible tree visualisation from the Centre for Genomic Pathogen Surveillance at https://www.phylocanvas.gl
History
search datasets
Unnamed history
17.6 kB
5
5: Auto Output
Add Tags
1 line
format newick, database ?

```
(((t3:0.0439147589789221429 2,t7:0.0300650268
108352283117)80:0.11476949164688564353)50:0.
```

Figure 17: After Galaxy completes RAWR analysis.

Hit the **visualize** button on the righthand side inside the **Auto Output** box to pull up the visualization options (Figure 17). Choosing **Phylogenetic Tree Visualization** will generate a visualization for your phylogeny tree (see Figure 18).

To generate a visualization with bootstrap support in local Galaxy, please read the next section.

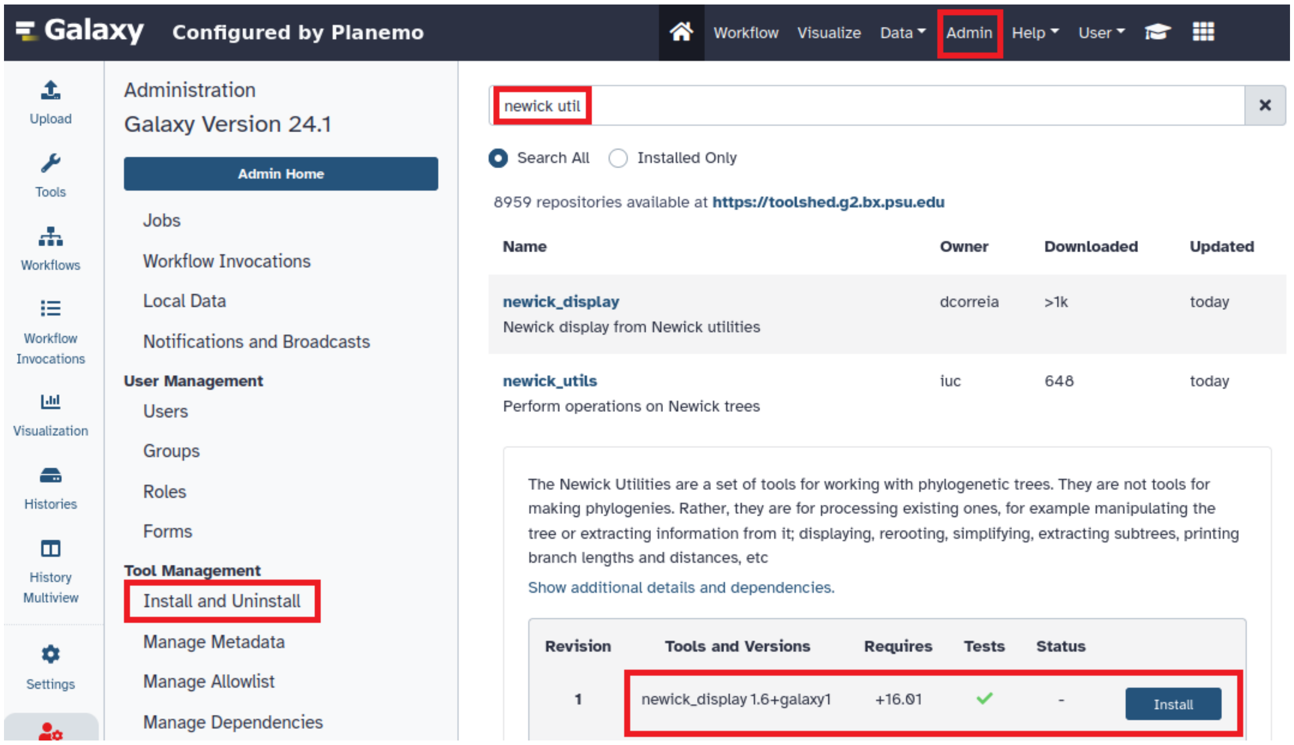


```
(((t3:0.0439147589789221429 2,t7:0.0300650268
108352283117)80:0.11476949164688564353)50:0.
```

Figure 18: Result of running rawrsampler on local Galaxy instance.

Local Galaxy instance, visualizing tree with support values through Newick Display, a tool downloadable from Galaxy ToolShed

After launching the local Galaxy instance (see install instructions), navigate to the **Admin** page and under **Tool Management > Install and Uninstall** search for keywords “newick util” (see Figure 19). Hit **Install** button next to the tool with package name “newick_display 1.6+galaxy1” (see Figure 19).



8959 repositories available at **https://toolshed.g2.bx.psu.edu**

Name	Owner	Downloaded	Updated
newick_display Newick display from Newick utilities	dcorreia	>1k	today
newick_utils Perform operations on Newick trees	iuc	648	today

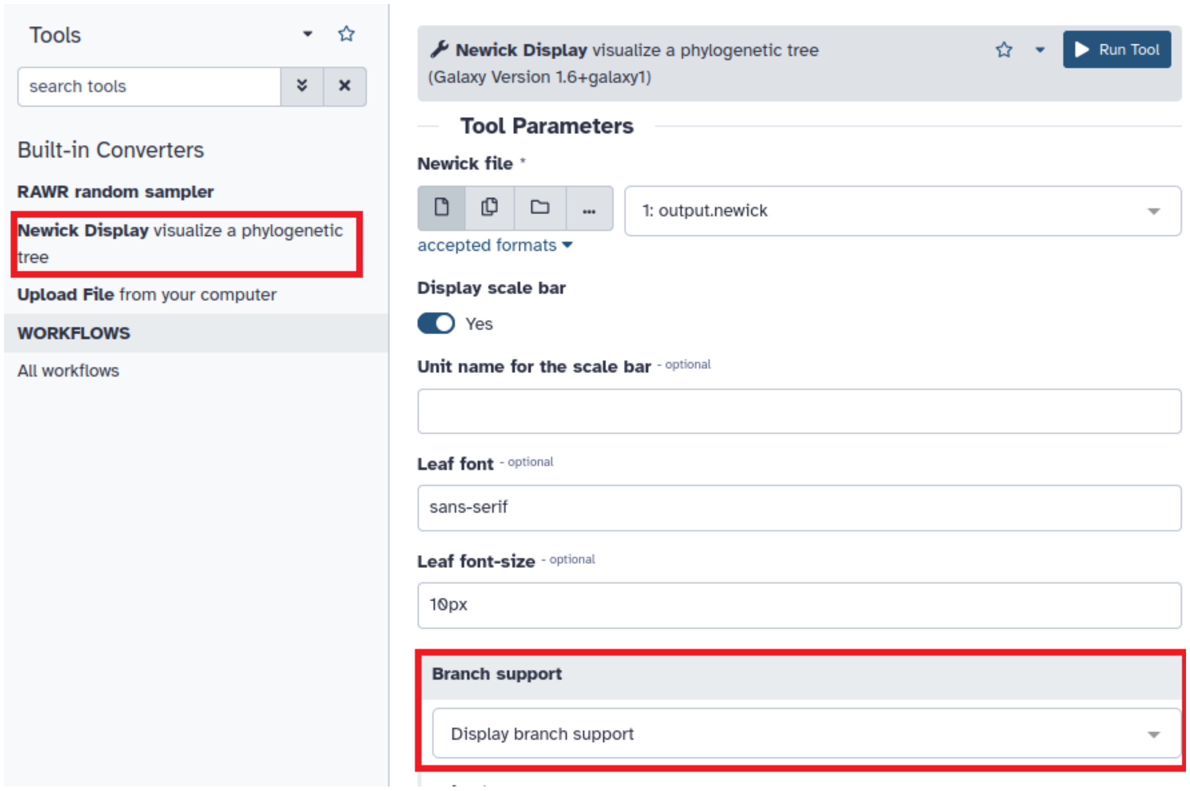
The Newick Utilities are a set of tools for working with phylogenetic trees. They are not tools for making phylogenies. Rather, they are for processing existing ones, for example manipulating the tree or extracting information from it; displaying, rerooting, simplifying, extracting subtrees, printing branch lengths and distances, etc

Show additional details and dependencies.

Revision	Tools and Versions	Requires	Tests	Status
1	newick_display 1.6+galaxy1	+16.01	✓	-

Figure 19: Installing Newick Display for local Galaxy.

Refresh the webpage and the newly installed **Newick Display** will be automatically added to the **Tools** section. You can use the RAWR output file as input and make sure to enable **Display branch support** option (see Figure 21). You can choose to save the file type as **svg** or **png**, we tested with png. Hit **Run Tool** to generate the image file.



Newick Display visualize a phylogenetic tree (Galaxy Version 1.6+galaxy1)

Tool Parameters

Newick file *: 1: output.newick

Display scale bar: Yes

Unit name for the scale bar - optional:

Leaf font - optional: sans-serif

Leaf font-size - optional: 10px

Branch support: Display branch support

Figure 20: Using Newick Display to generate tree with bootstrap support.

Once the image is generated, it can be visualized in **OpenSeadragon**, a default, preinstalled visualization tool (see Figure 21). You can then **toggle full page** to view the full image (see Figures 22 & 23). This image can be saved to file.

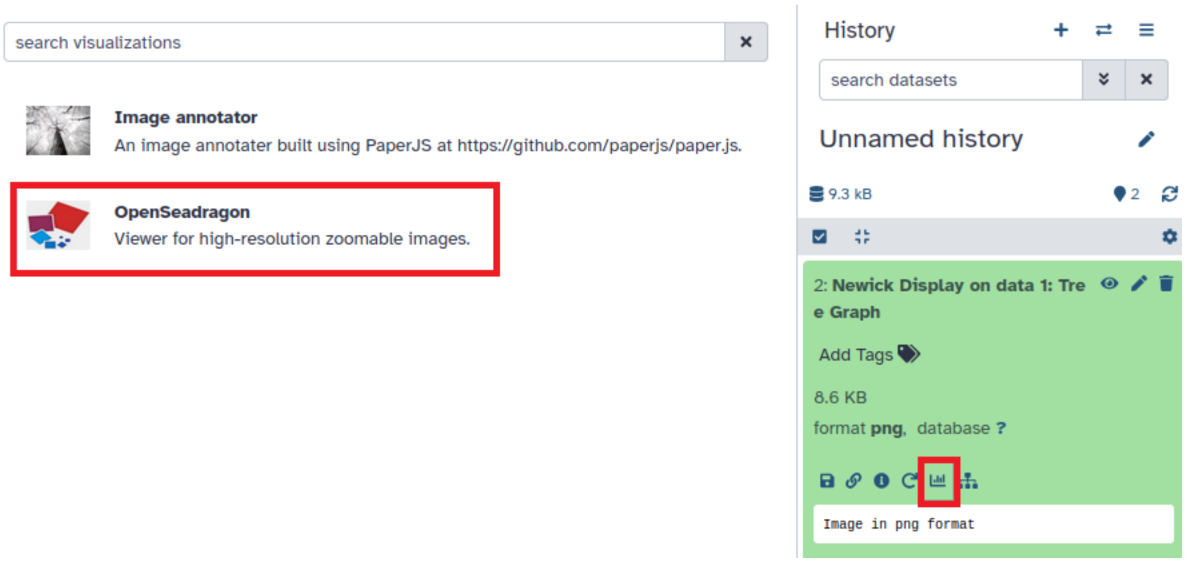

search visualizations
Image annotator
An image annotater built using PaperJS at https://github.com/paperjs/paper.js.
OpenSeadragon
Viewer for high-resolution zoomable images.
History
search datasets
Unnamed history
9.3 kB
2
2: Newick Display on data 1: Tree Graph
Add Tags
8.6 KB
format png, database ?
Image in png format

Figure 21: Open the generated image.

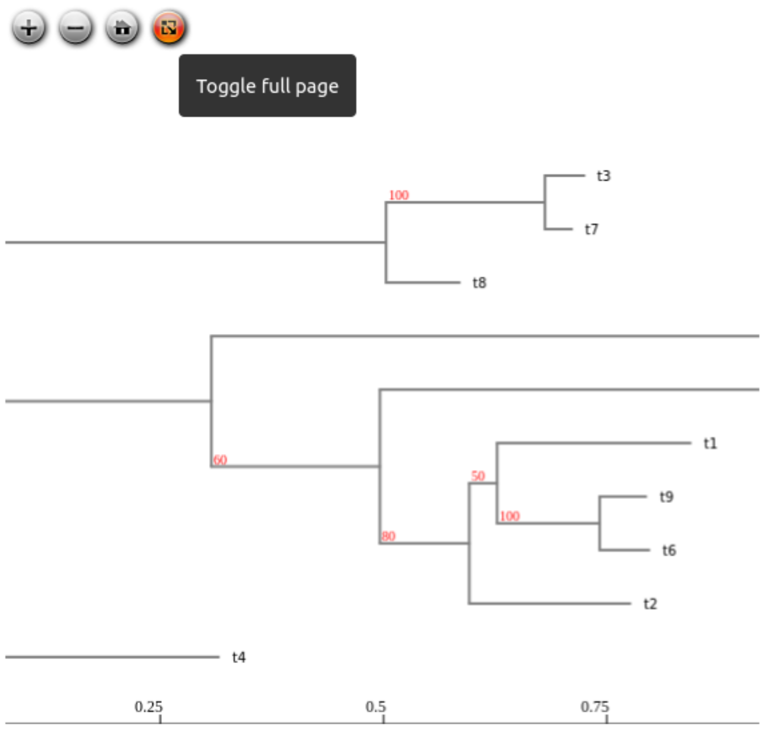


Figure 22: Toggle full page display in OpenSeadragon.

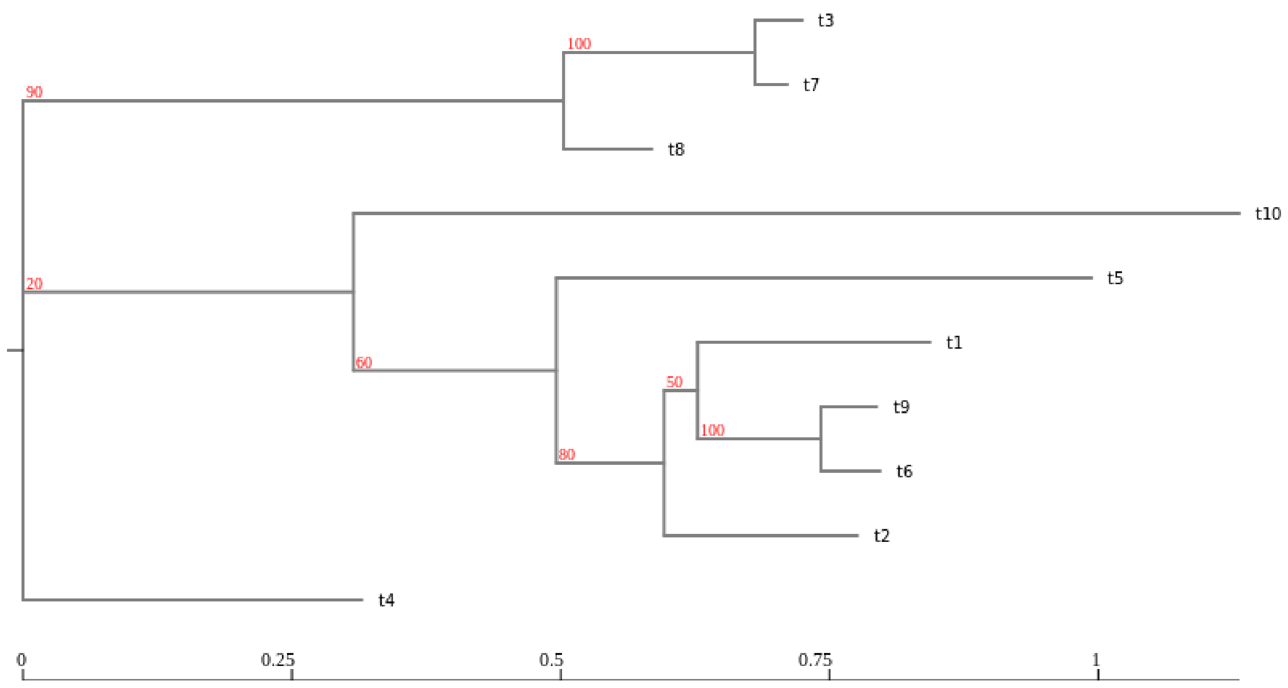
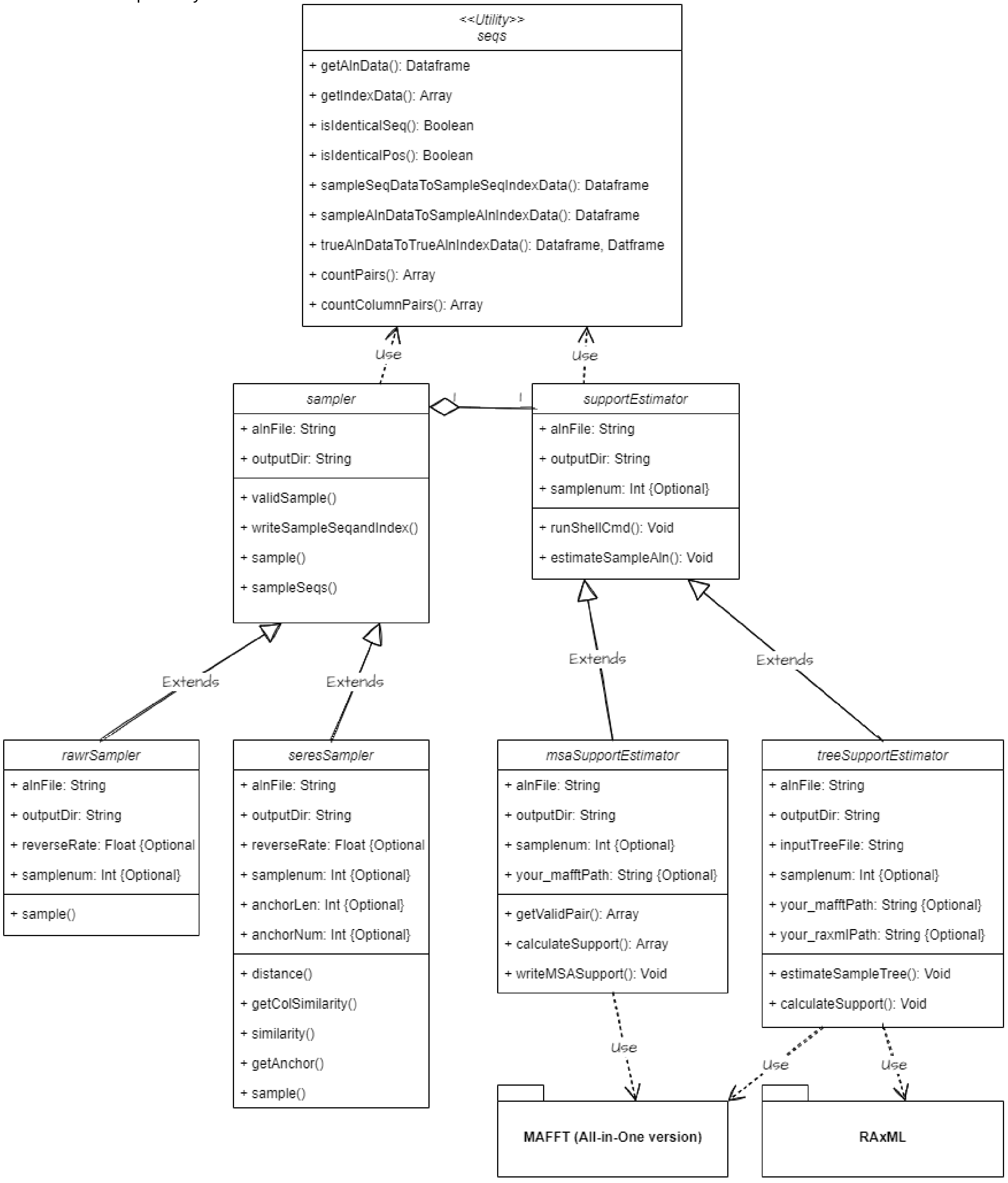


Figure 23: Phylogenetic tree with bootstrap support displayed inside local Galaxy instance.

API

API UML class diagram

RAWR’s Application Program Interface (API) is shown in Figure 24. There are three source code files, “sampler.py”, “supportEstimator.py”, and “seqs.py”. RAWR depends on MAFFT and RAxML programs to perform aligning and phylogeny reconstruction respectively.



<<Utility>>
seqs
+ getAlnData(): Dataframe
+ getIndexData(): Array
+ isIdenticalSeq(): Boolean
+ isIdenticalPos(): Boolean
+ sampleSeqDataToSampleSeqIndexData(): Dataframe
+ sampleAlnDataToSampleAlnIndexData(): Dataframe
+ trueAlnDataToTrueAlnIndexData(): Dataframe, Datframe
+ countPairs(): Array
+ countColumnPairs(): Array
Use
Use
sampler
+ alnFile: String
+ outputDir: String
+ validSample()
+ writeSampleSeqandIndex()
+ sample()
+ sampleSeqs()
1
1
supportEstimator
+ alnFile: String
+ outputDir: String
+ samplenum: Int {Optional}
+ runShellCmd(): Void
+ estimateSampleAln(): Void
Extends
Extends
Extends
Extends
rawrSampler
+ alnFile: String
+ outputDir: String
+ reverseRate: Float {Optional
+ samplenum: Int {Optional}
+ sample()
seresSampler
+ alnFile: String
+ outputDir: String
+ reverseRate: Float {Optional
+ samplenum: Int {Optional}
+ anchorLen: Int {Optional}
+ anchorNum: Int {Optional}
+ distance()
+ getColSimilarity()
+ similarity()
+ getAnchor()
+ sample()
msaSupportEstimator
+ alnFile: String
+ outputDir: String
+ samplenum: Int {Optional}
+ your_mafftPath: String {Optional}
+ getValidPair(): Array
+ calculateSupport(): Array
+ writeMSASupport(): Void
treeSupportEstimator
+ alnFile: String
+ outputDir: String
+ inputTreeFile: String
+ samplenum: Int {Optional}
+ your_mafftPath: String {Optional}
+ your_raxmlPath: String {Optional}
+ estimateSampleTree(): Void
+ calculateSupport(): Void
Use
Use
Use
MAFFT (All-in-One version)
RAxML

Figure 24: RAWR API class diagram.

RAWR API Examples

How to use RAWR algorithm to resample an alignment file:

```
rawr = sampler.rawrSampler(alnFile, outputDir)
rawr.sampleSeqs()
```

How to use SERES algorithm to resample an alignment file:

```
seres = sampler.seresSampler(alnFile, OutputDir)
seres.sampleSeqs()
```

How to adjust parameters for RAWR resampling:

```
rawr = sampler.rawrSampler(alnFile, outputDir, reverseRate, samplenum)
rawr.sampleSeqs()
```

How to adjust parameters for SERES resampling:

```
seres = sampler.seresSampler(alnFile, OutputDir, reverseRate,samplenum, anchorLen, anchorNum)
seres.sampleSeqs()
```

How to use RAWR algorithm with MSA support estimation:

```
rawr = sampler.rawrSampler(alnFile, outputDir)
rawr.sampleSeqs()
rawr_msa = supportEstimator.msaSupportEstimator(alnFile, outputDir)
rawr_msa.calculateSupport()
```

How to use SERES algorithm with phylogeny support estimation:

```
seres = sampler.seresSampler(alnFile, OutputDir)
seres.sampleSeqs()
seres_tree = supportEstimator.treeSupportEstimator(alnFile, outputDir, treeFile, samplenum)
seres_tree.calculateSupport()
```

How to use SERES algorithm with phylogeny support estimation and custom MAFFT & RAxML software:

```
seres = sampler.seresSampler(alnFile, OutputDir)
seres.sampleSeqs()
seres_tree = supportEstimator.treeSupportEstimator(alnFile, outputDir, treeFile, samplenum,
treeFile,samplenum,"/path/to/mafft","/path/to/raxml")
seres_tree.calculateSupport()
```

How to output JalView format MSA support file to import into and visualize in JalView

```
from src.MSA_support_csv_2_jalview_sequence_annotation import support_csv_2_jalview

rawr = sampler.rawrSampler(alnFile, outputDir)
rawr.sampleSeqs()
rawr_msa = supportEstimator.msaSupportEstimator(alnFile, outputDir)
rawrMSAoutput = rawr_msa.calculateSupport()
support_csv_2_jalview(rawrMSAoutput,"pink")
```

Hints to Help Debug Errors

1. When you are using the RAWR GUI software or the RAWR API, **creating a new output folder each time is important** because sometimes, program crashes can lead to remanent files that can conflict with future software runs. The reason for this behaviour is RAxML checks for the existence of RAxML_info.name files in the folder and will refuse to run if the file already exists.

2. **If speed is of the utmost importance in your work pipeline, we recommend that you try the RAWR v1.0 API and parallelize by independent samples.** Depending on the specifications of your computer, RAWR v1.0 in Galaxy, GUI, and web server will take some time to process your samples. GUI desktop client has multiprocessing in some areas of the analyses, but not all. This is because RAWR v1.0 is currently a sequentially implemented program, but we have plans to implement parallelization to speed up both the software and the web server in the future.

3. **Let's talk about a hypothetical case: your RAWR program crashes hundreds of times during testing and your computer feels slow and sluggish.** It is possible that you are experiencing the slowdown from orphaned threads. If so, you can try killing the subprocesses that RAWR spawns by opening your terminal and killing these processes by name. Examples by operating system below:

   a. Windows:
   ```
   taskkill /IM "RAxML" /F
   taskkill /IM "mafft" /F
   ```
   b. macOS:
   ```
   pkill RAxML
   pkill mafft
   ```
   c. Linux:
   ```
   pkill RAxML
   pkill mafft
   ```

Related Publications

Wang, W., Hejasebazzi, A., Zheng, J., & Liu, K. J. (2021). Build a better bootstrap and the RAWR shall beat a random path to your door: phylogenetic support estimation revisited. Bioinformatics, 37(Supplement_1), i111-i119.

Wang, W., Smith, J., Hejase, H. A., & Liu, K. J. (2020). Non-parametric and semi-parametric support estimation using SEquential RESampling random walks on biomolecular sequences. Algorithms for Molecular Biology, 15(1), 1-15.

Wang, W., Wuyun, Q., & Liu, K. J. (2019, November). An application of random walk resampling to phylogenetic HMM inference and learning. In 2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (pp. 44-51). IEEE.