2 Basic setups

Installing and testing Nemo and SLiM

Prerequisites

On all systems:

- a simple text editor (not Word!), like Textedit, Notepad, gedit, kate, etc.
- R or RStudio
- a terminal or console to run command line tools

LINUX:

- the GSL (GNU Scientific Library): `sudo apt install libgsl-dev`
- a recent version of `gcc` compiler (≥ 7) able to compile C++11 code
- `cmake` and `make` for compilation

Prerequisites: MacOS and Windows

MacOS:

- `Terminal` application (found in /Applications/Utilities)
- not required but useful to install Apple Developer Tools to compile programs from source

Windows (≥ 7):

- install Cygwin from: https://www.cygwin.com/install.html; install following packages:
  - gcc-core-11.3.0-1; gcc-g++-11.3.0-1
  - make-4.4.1-2
  - libgsl-devel-2.3-2; libgsl19-2.3-2
  - cmake-3.23.2-1
- `Notepad` or `Notepad++` or any text editor allowing you to write code in ASCII text format.

Nemo

There are two versions of Nemo:

nemo: original version, most complete set of features, no age structure

- homepage: https://nemo2.sourceforge.io
- download latest version from the download site

nemoage: version for age-structured populations, not complete set of features

- homepage: https://bitbucket.org/ecoevo/nemo-age-release
- download MacOS and Windows packages from the download page
- for Linux system, install from git repository with command (in a terminal window):

```
git clone https://bitbucket.org/ecoevo/nemo-age-release.git
```

We will use both versions of Nemo!

Nemo: installation from source

Installing the source code and compiling:

nemo (v2.3.56):

```
wget -q https://sourceforge.net/projects/nemo2/files/Nemo/2.3.56/Nemo-2.3.56-src.tgz
```

```
tar xzf Nemo-2.3.56-src.tgz
```

```
cd Nemo-2.3.56
```

```
make -j8
```

`(sudo) make install` (check and change install directory in `Makefile`)

`nemo2.3.56` (checking that nemo is directly accessible on your command `PATH`)

Nemo: installation from source

Installing the source code and compiling:

nemo (v2.3.56):

```
wget -q https://sourceforge.net/projects/nemo2/files/Nemo/2.3.56/Nemo-2.3.56-src.tgz
tar xzf Nemo-2.3.56-src.tgz
cd Nemo-2.3.56
make -j8
(sudo) make install
```
(check and change install directory in `Makefile`)

`nemo2.3.56` (checking that nemo is directly accessible on your command `PATH`)

nemo-age (v0.32.0):

```
git clone https://bitbucket.org/ecoevo/nemo-age-release.git
cd nemo-age-release
make -j8
(sudo) make install
```
(check and change install directory in `Makefile`)

`nemoage0.32.0` (checking that nemo is directly accessible on your command `PATH`)

Nemo: installation on MacOS and Windows without compiling

To make things easier for people not using Linux, we have prepared binary files ready to copy and use.

For **nemo**, download the following packages:

`https://sourceforge.net/projects/nemo2/files/Nemo/2.3.56/Nemo-2.3.56-MacOSX.tgz`

`https://sourceforge.net/projects/nemo2/files/Nemo/2.3.56/Nemo-2.3.56-Win64.tgz`

For **nemoage**:

`https://bitbucket.org/ecoevo/nemo-age-release/downloads/NemoAge-0.32.0-MacOSX.tgz`

`https://bitbucket.org/ecoevo/nemo-age-release/downloads/NemoAge-0.32.0-Win64.tgz`

1. Expand the `tar` archives and look for the program files in:

`MacOSX:` `MacOSX/binaries`

`Windows:` `Win/binaries`

2. Copy (manually) the program file into your dedicated directory: `C:/cygwin/home/your-user-name/bin` on Windows; `/Users/your-user-name/bin` on MacOS.

Linux trick #1: the command PATH

The PATH is an *environment variable* of your *shell* environment. It sets the paths to the directories that contain program (executable files) on your computer. Every time you type a command in your terminal window, your shell looks into each directory on your PATH until it finds the program corresponding to the command you typed.

What is your PATH? Check it:

```
echo $PATH
```

Adding another directory to your PATH:

```
PATH="$HOME/bin:$PATH"
```

```
export PATH
```

`echo $PATH` (checking)

Now, the different versions of Nemo that you installed in your $HOME directory are accessible directly on the command line.

Linux trick #1: the command PATH

But, we don't want to set the path every time we open a shell (terminal window). It would be better if that was done automatically. For that, we will add the PATH-modification code to a local system file, either `.bashrc` or `.profile` (or `.zhrc` on macOS 11+):

`cd` (`cd` without argument should take you `$HOME`)
`ls -al` (check which hidden file you have in your home directory)
edit the file, for e.g., `.bashrc`
`nano .bashrc`
once in the file, navigate somewhere towards the end of the file and add (write) the lines we had above:
`PATH="$HOME/bin:$PATH"`
`export PATH`
then save the file and exit `nano` with: CTRL-o, CTRL-x

Note: this should work on MacOS and Windows although you may want to use a graphical text editor. Also, on Windows, the files to edit are in your home directory in the `CygWIn` environment (in `C:/cygwin/home/your-user-name/`).

SLiM

Download the program from: https://messerlab.org/slim/

Follow the instructions provided in SLiM manual. If you are not on MacOS, you will have to compile the source code on your machine. The SLiM manual provides detailed instructions on how to do so. The easiest is to compile from scratch.

As we are using CygWin on Windows for Nemo, it would be wise to also use CygWin to compile SLiM instead of using MySys as advised in SLiM's manual. To do so, simply follow the compiling procedure described for Linux and execute them in CygWin.

Linux trick #2: how to execute a program

As mentioned before, when typing a command to execute in a terminal window, the shell will look for the program in the directories set by the `PATH` variable. This is not the only way to execute a program though. Here are two tricks that are useful to know:

- To execute a program located in your working directory, simply type:
  `./progname`
  This way, the program is not looked for on the `PATH` but in your current working directory.
- Similarly, you can provide the path to the program file explicitly:
  `bin/nemo2.3.56`
  This would launch nemo provided that the `bin` folder is in your working directory from where you execute that command. The path to the program file can also be absolute (start with a dash '/').