

User Manual

A Computational Model of Paradoxical Sleep

Eden DARNIGE
Arthur GRIMAUD
Amélie GRUEL
Alexia KUNTZ

Paul BIELLE
Lola DENET
Charles GUINOT
Tongyuxuan HUI
Wenli NIU

Supervised by Dr. Charlotte HÉRICÉ

May 2019

May 2020

Master de Bioinformatique de Bordeaux

Contents

1	Setting up the environment	2
1.1	Operating system	2
1.2	Required downloads and installations	2
1.2.1	Python 3	2
1.2.2	R	3
1.3	Retrieving the model	4
2	Running the simulation	5
2.1	Launching the model	5
2.2	Uploading and defining parameters	5
2.3	Simulation	8
3	Visualizing and manipulating the results	9
3.1	Graphical visualization	9
3.2	Statistical analyses	10
4	Potential sources of errors and resolutions	12
4.1	Parameters	12
4.2	Simulation	12

Introduction

This manual is an addition to the report "Developing a Computational Model of Paradoxical Sleep" and "Élaboration d'un modèle computationnel modélisant le cycle éveil/sommeil chez le rongeur". It describes how to set up the user environment in order to run the described simulations. The model's main goal is to provide tools that aid in predicting the role of different types of neurons in the sleep cycle, and studying the effect of lesions between different neuronal populations. In addition, it provides insight into the effects of microinjections of certain neurotransmitters on REM sleep generation and maintenance and the sleep cycle in rodents.

In this manual, the required operating system, downloads, and packages needed to get started will be stated. Then, the steps that prepare the running of the simulation are described. Finally, the different options for the treatment of the results can be explored at the end of the document. This program has been developed and modified by two groups of students in 2019 and 2020. For more information on the architecture and design of the model, please refer to "Developing a Computational Model of Paradoxical Sleep" and "Élaboration d'un modèle computationnel modélisant le cycle éveil/sommeil chez le rongeur".

Chapter 1

Setting up the environment

1.1 Operating system

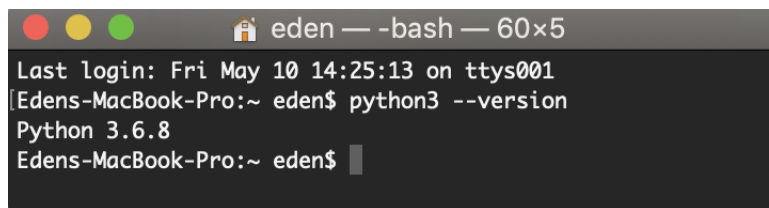
The simulation is designed to run in Unix operating systems, whether it be on Linux variants or Mac OS. It requires a terminal emulator that provides a command line interface to the operating system when used in conjunction with a Unix shell.

1.2 Required downloads and installations

1.2.1 Python 3

The model is written almost entirely in Python 3. Downloading the most recent version of Python 3 can be done at:

<https://www.python.org/downloads/>

A screenshot of a terminal window. The title bar shows three colored window control buttons (red, yellow, green) and the text 'eden — -bash — 60x5'. The terminal content shows: 'Last login: Fri May 10 14:25:13 on ttys001', 'Edens-MacBook-Pro:~ eden\$ python3 --version', 'Python 3.6.8', and 'Edens-MacBook-Pro:~ eden\$' followed by a cursor.

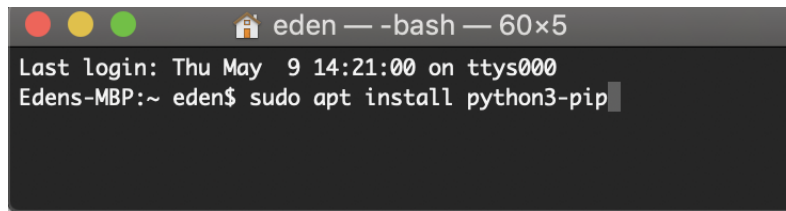
```
eden — -bash — 60x5
Last login: Fri May 10 14:25:13 on ttys001
Edens-MacBook-Pro:~ eden$ python3 --version
Python 3.6.8
Edens-MacBook-Pro:~ eden$
```

Figure 1.1: How to check which version of Python 3 is installed from the terminal.

Python libraries and modules

A number of Python libraries and modules are used in this model in order to perform computations and to produce different graphics. It is necessary to have them all installed for the model to function.

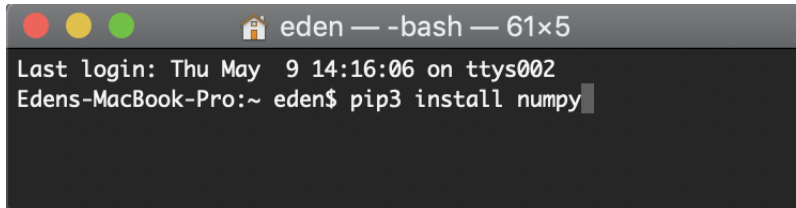
It can be helpful to have the Python 3 package manager, pip3, installed to help with further installations. To do so, enter within the terminal:



```
eden — -bash — 60x5
Last login: Thu May  9 14:21:00 on ttys000
Edens-MBP:~ eden$ sudo apt install python3-pip
```

Figure 1.2: Screenshot of the command to download pip3 from the terminal.

The following list of libraries and modules can be installed in the terminal using:



```
eden — -bash — 61x5
Last login: Thu May  9 14:16:06 on ttys002
Edens-MacBook-Pro:~ eden$ pip3 install numpy
```

Figure 1.3: Screenshot of the command to download NumPy from the terminal.

- Tkinter - Graphical user interface
- NumPy - Calculations and array manipulation
- Matplotlib.pyplot - Graphics and plotting
- Pylab - Plotting features
- Python Standard Library - Extensive, offers a wide range of facilities :
 - CSV - Read and write tabular data in CSV format
 - math - Access to the mathematical functions
 - OS - Interface with the underlying operating system
- Graphviz - Graphic descriptions in the DOT language

1.2.2 R

The statistics portion of the program is written in R. It is therefore necessary to retrieve the most recent version from:

<https://www.r-project.org/>



```
eden — -bash — 60x15
Last login: Fri May 10 14:25:29 on ttys001
Edens-MacBook-Pro:~ eden$ R --version
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)
```

Figure 1.4: How to check which version of R is installed from the terminal.

R packages

Within R, the following packages must be installed using:

```
install.packages("name of package")
```

- Ggplot2 - Create graphics
- gridExtra - Align multiple graphs

1.3 Retrieving the model

All of the files involved in the model can be downloaded from GitHub:

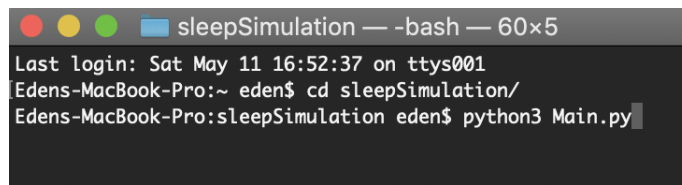
https://github.com/lola-denet/PdP_Sommeil_Rongeur

Chapter 2

Running the simulation

2.1 Launching the model

With the model files placed together in one working directory, the simulation can be run the "Main" Python script. Only one line of code, seen in 2.1, must be executed in the terminal.



```
sleepSimulation — -bash — 60x5
Last login: Sat May 11 16:52:37 on ttys001
Edens-MacBook-Pro:~ eden$ cd sleepSimulation/
Edens-MacBook-Pro:sleepSimulation eden$ python3 Main.py
```

Figure 2.1: Screenshot of the command to execute in order to launch the model from the terminal.

A window containing the graphical interface pops up. It contains 5 tabs: **Main**, **Parameters**, **Run**, **Visualization**, and **Statistics**. All of which can be clearly seen in Figure 2.2.

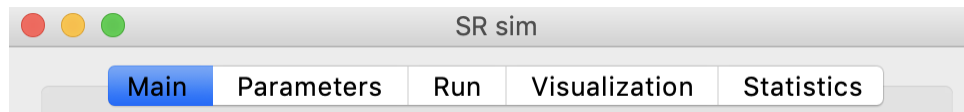


Figure 2.2: Screenshot of all 5 tabs.

2.2 Uploading and defining parameters

In the **Main** tab (Figure 2.3), the default parameter text file can be uploaded by clicking on "Load model." Once the parameters have been uploaded, "Display network" generates a graph of the cycle network created. "Display connections" prints out the connections defined in the terminal, and entering a cycle phase (Wake, REM, NREM) in "PrintCompParamAndType" displays the parameters related to the state. These last two tools are useful in debugging.

Many models (3 populations human and rodent, 6 populations rodent) are provided in the GitHub download folder containing the simulation.

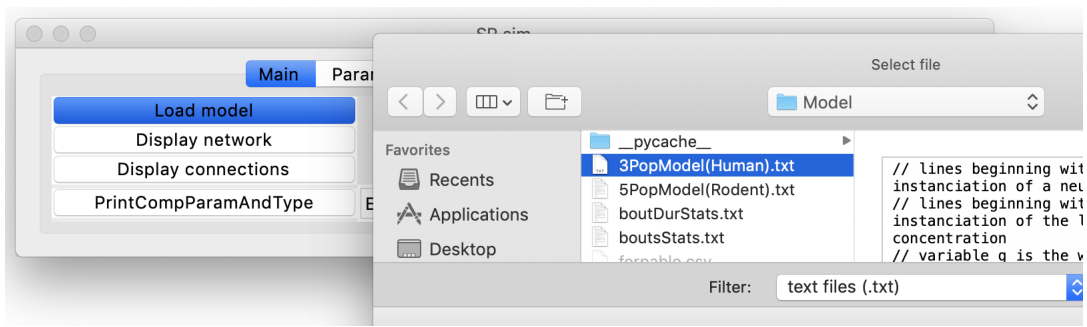


Figure 2.3: Screenshot showing how the user can upload the parameters file of their choice by selecting the "Load model" button.

Then, in the **Parameters** tab, a neuronal population, homeostatic sleep drive or connection object can be added by clicking "Add Object to Network" as shown in figure 2.4.

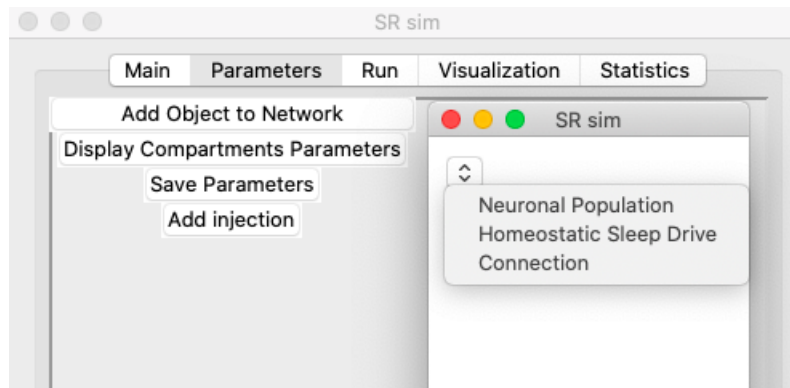


Figure 2.4: add object

In this same tab, the parameters can be modified by clicking on "Display Compartments Parameters" as depicted in Figure 2.5. It can be used to execute a lesion on a connection or to isolate an entire population.

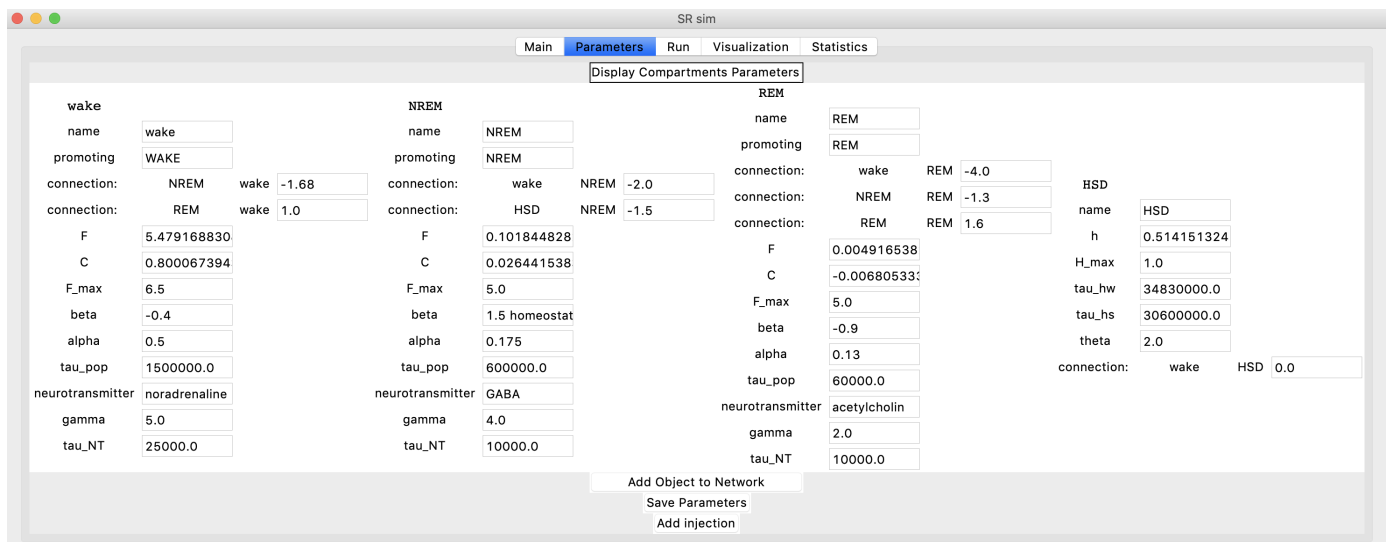


Figure 2.5: Screenshot showing how the user can easily modify the parameters.

Modifications of parameters can be saved by clicking "Save Parameters".

Furthermore, the bouton "Add injection" is in charge of the additional function of micro-injection. Type of injection must be selected in the window as figure 2.6.

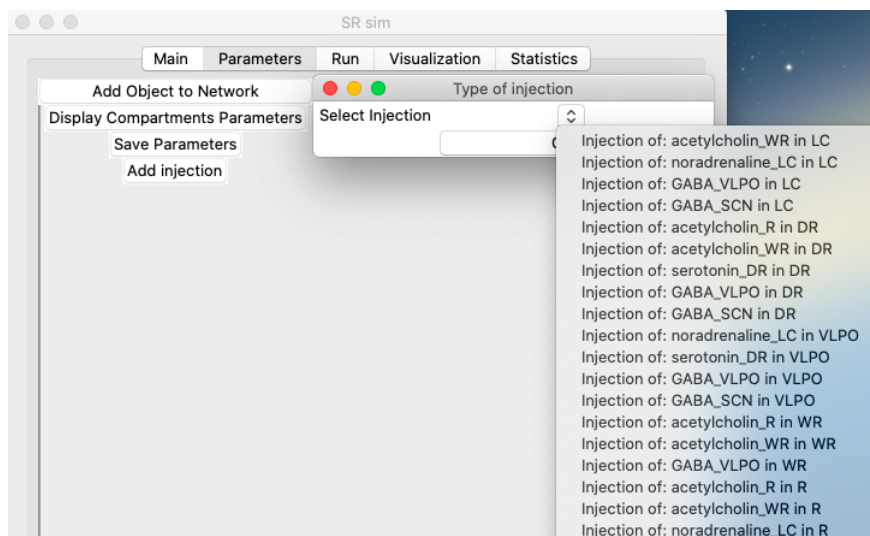


Figure 2.6: Add injection.

In the figure 2.7, the parameters of neuro-transmitter chosen will be affiched automatically, and they can be modified directly by user. After clicking on "create" button, the micro-injection of neurotransmitter agonist or antagonist will be simulated.

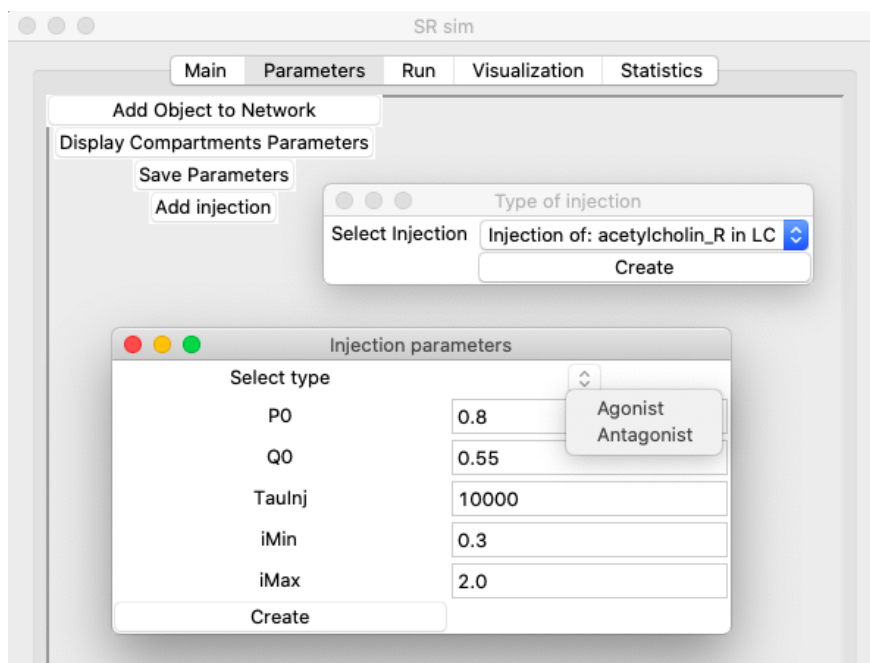


Figure 2.7: The parameters of neuro-transmitter chosen

2.3 Simulation

Finally, in the **Run** tab, options defining the run time (in seconds), the resolution, and the save rate can be modified. In addition, noise can be added in Hz by defining the mean and standard deviation of the additive white Gaussian noise (Figure 2.8). The threshold determining the state based on neuromodulator concentration can be modified. Choose between the Euler method and Runge-Kutta of the 4th order by clicking on the arrows by "Select Resolution Method."

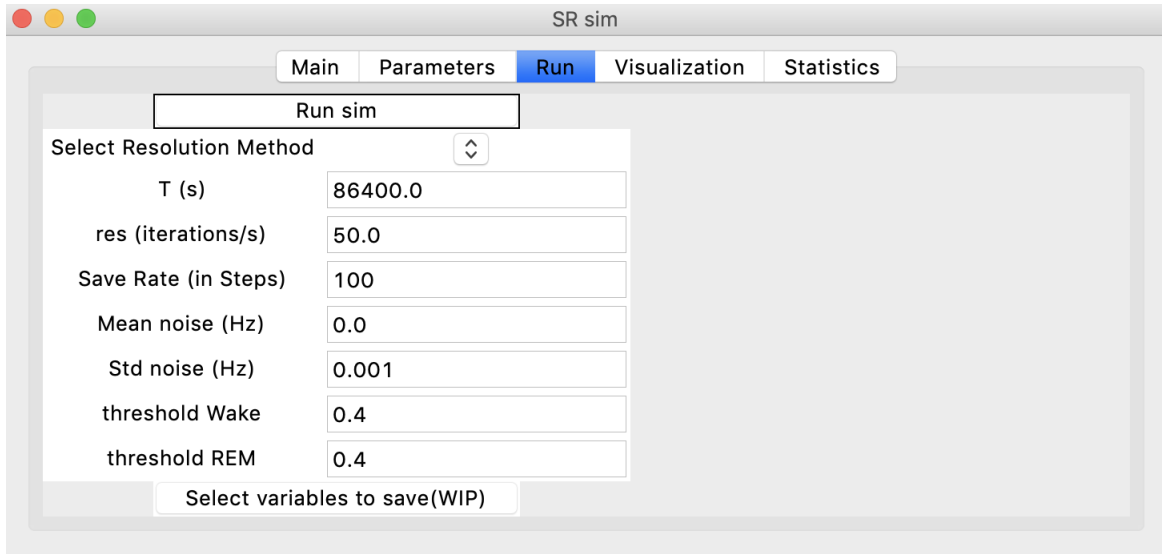


Figure 2.8: Screenshot showing how the user can modify the parameters before running the simulation.

Depending on the purpose, the "Select variables to save" button allows for the customization of the output CSV file. "Run sim" begins the simulation. The results file can be saved under any desired name in .CSV format. To re-run, the model parameters must be re-uploaded in **Main**.

Chapter 3

Visualizing and manipulating the results

3.1 Graphical visualization

With the data saved, the **Visualization** tab provides four tools to display the results (figure 3.1).

A graphic depicts the firing rates and neuro-transmitter concentrations evolution over time, as well as a hypnogram. It can be produced directly from the simulation's results or from precedent results saved in a CSV file. The mean of multiple results can also be represented on a graph, with or without the standard deviation, as shown in Figure 3.2. By clicking on "Compare with a control," two graphs can be compared by superposing the graphs of the results (or mean of results) on the lightened control graph. The graphs can be saved directly from the window by selecting the save icon.

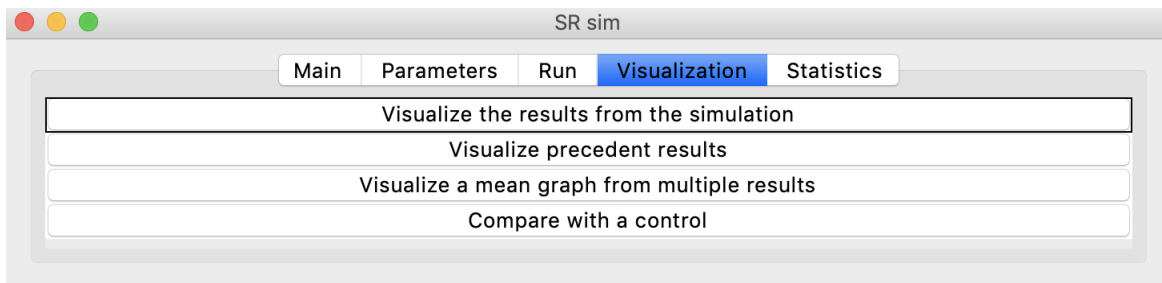


Figure 3.1: Screenshot of visualization options.

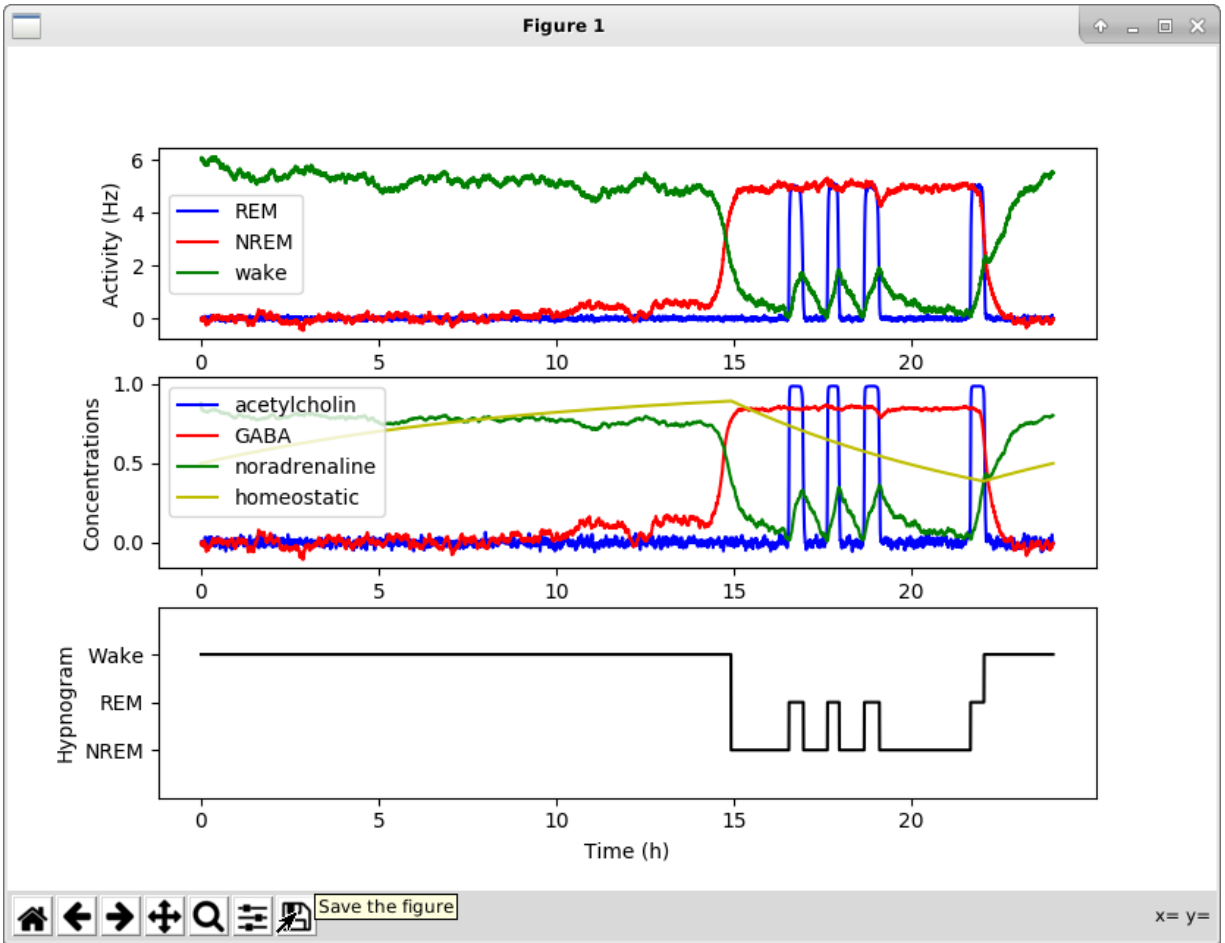


Figure 3.2: The displayed graph corresponds to what can be obtained after a simulation of 24 hours, with added noise and without injections or lesions.

3.2 Statistical analyses

When noise is introduced into a simulation, the same simulation using the same parameters may be run multiple times.

The last tab, **Statistics**, provides an automatic analysis of the collective hypnogram results. The result files, all located within the same directory, are selected after clicking "Select results files."

The output is an image containing bar graphs depicting the mean percent time spent in each state, the mean number of bouts in each state, and the mean duration of bouts by state in minutes (Figure 3.3). Each graph has a corresponding statistical analysis .txt file (ANOVA and Tukey HSD Post-Hoc). The output files are located in the same directory as the selected results files.

Within the R script containing the statistical analysis, there is a guide that explains how to manually import and format the data to be used within the script. This can be useful in the case of modifications to the statistical analysis, such as adding additional tests.

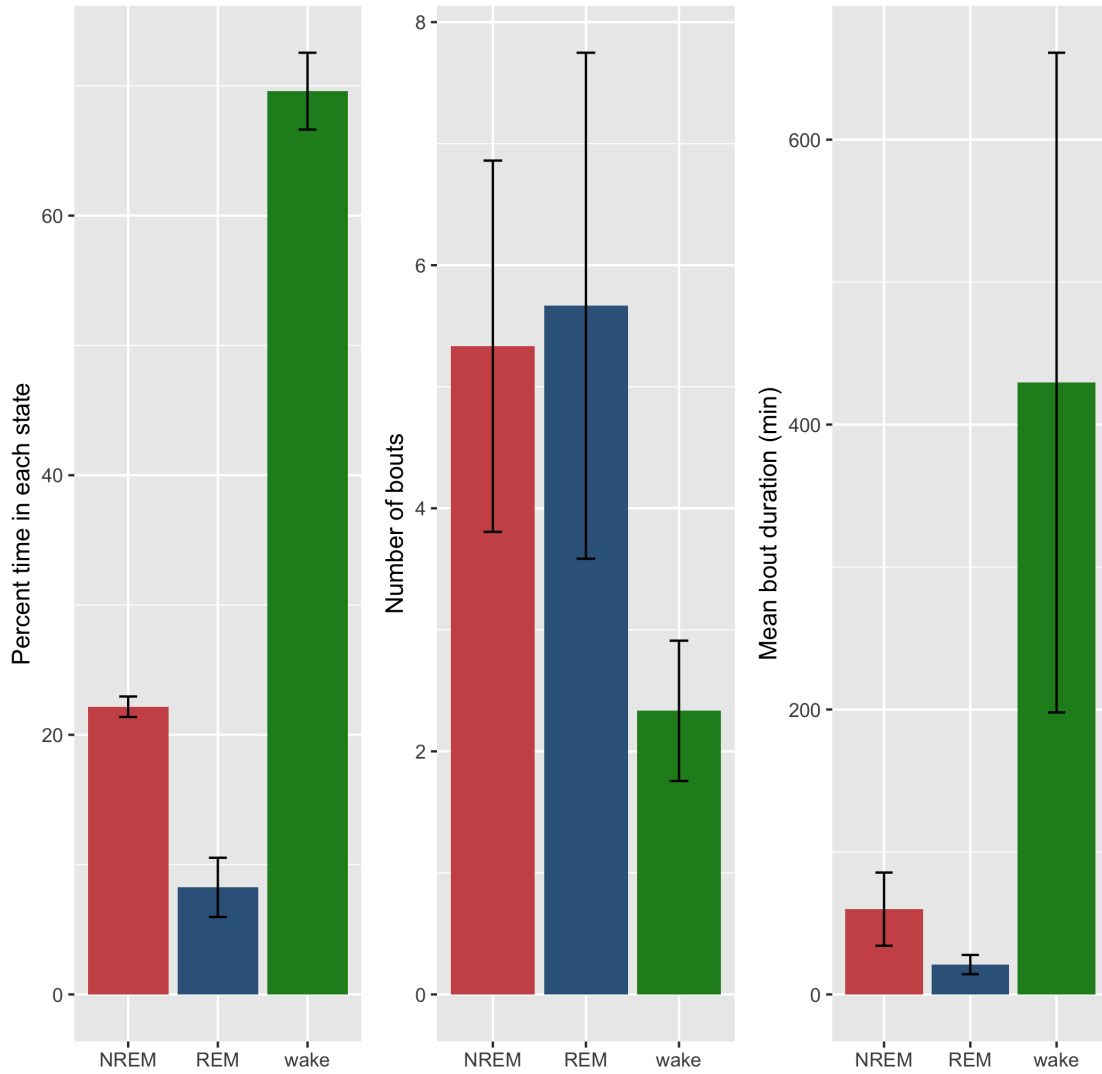


Figure 3.3: Example of statistics graphic saved to the directory containing the model as "plot-Stats.png." Bars represent mean values from all input data with standard deviation. In this example, $n = 3$ simulations.

Chapter 4

Potential sources of errors and resolutions

4.1 Parameters

It is important to be aware of the units of measurement when entering the parameters. At this time, the model omits the units in the GUI. Figure 4.1 describes the units for each parameter.

4.2 Simulation

As it stands, the user must use a resolution ("res" in iterations per second) of 50.0 when running the simulation. Otherwise, the time elapsed within the simulation is affected and the results are chronologically incorrect.

Neuronal Populations (wake, NREM, REM)	
F	ms ⁻¹
C	aU
F max	ms ⁻¹
beta	ms ⁻¹
alpha	ms ⁻¹
tau pop	ms ⁻¹
concentration	Neuromodulator (E,G, A)
gamma	ms ⁻¹
tau NT	ms
Homeostatic sleep drive	
h	aU
H max	aU
tau hw	ms
tau hs	ms
theta X	ms ⁻¹
Injection	
P0	aU
TauInj	ms
iMin	aU
iMax	aU

Figure 4.1: The displayed graph corresponds to what can be obtained after a simulation of 24 hours, with added noise and without injections or lesions.