sim2bids

Release 1.1.2

Dinara Issagaliyeva

Jul 03, 2023

CONTENTS

Version: 1.2.1

sim2bids is a Python package to created to convert computational data to BIDS standard as proposed by Michael Schirner and Petra Ritter.

The specification proposes a data structure schema for neural network computer models that aims to be generically applicable to all kinds of neural network simulation software, mathematical models, computational models, and data models, but with a focus on dynamic circuit models of brain activity.

Importantly, they not only propose suggestions for a BIDS schema for computer models, but they also propose extensions to the entire BIDS standard that solve several other problems.

Note: This project is under active development.

CHAPTER

ONE

WHERE TO GO FROM HERE

1.1 Getting Started

1.1.1 Installation

Get the package

Simply run the following command to get the app up and running:

```
$ pip install sim2bids
```

Alternatively, either fork or obtain the latest sim2bids version by running the following:

```
$ git clone https://github.com/dissagaliyeva/sim2bids
$ cd sim2bids
$ python setup.py install
```

Then, open up your notebook and import the following packages to set up the notebook:

```
import sim2bids
import panel as pn
from sim2bids.sim2bids import MainArea
from sim2bids.app import app
pn.extension('tabulator', 'ace', 'jsoneditor', 'ipywidgets', sizing_mode='stretch_width',
↪ notifications=True)
```

Provide software-specific information

This app aims to help you and future users reproduce the results of your simulations. Specify the required fields before running the app to make the process easier. **Please pay attention to these fields *only if* input code meets the following conditions**:

- non-Python code (e.g., MATLAB, R, Julia)
- Python code with more than one rhythm-specific parameters (e.g., separate parameters for alpha and delta rhythms)
- Python code with a list of parameters, e.g., G values from 0.1 to 1.0 with a step of 0.15

MODEL_NAME
: Name of the model used in your simulation. Currently accepted models: *ReducedWongWang*, *HindmarshRose*, and *Generic2dOscillator*. The models follow the same default values as specified in TheVirtualBrain.

- ReducedWongWang

  a=270.0, b=108.0, d=0.154, gamma=0.000641, tau_s=100.0, w=0.9, J_N=0.2609, I_o=0.3, G=2.0, sigma_noise=1.e-09, tau_rin=100

- HindmarshRose

  r=0.001, a=1.0, b=3.0, c=1.0, d=5.0, s=1.0, x_1=-1.6

- Generic2dOscillator

  tau=1.25, a=1.05, b=0.2, omega=1.0, upsilon=1.0, gamma=1.0, eta=1.0

Example:

```
app.MODEL_NAME = 'ReducedWongWang'
```

MODEL_PARAMS

Model parameters used in the code. **If you have a Python file with up to one rhythm, the app supplements parameters without assistance**.

Example code for each cases above:

```
# Example 1: non-Python code
app.MODEL_NAME = 'ReducedWongWang'
app.MODEL_PARAMS = dict(a=1., b=2., c=3., G=np.arange(0.1, 1., 0.15))

# Example 2: Python code with more than one rhythm-specific parameters
app.MODEL_PARAMS = dict(alpha=dict(a=1., b=3.),
                        delta=dict(a=2., b=1.))

# Example 3: Python code with a list of parameters
app.MODEL_PARAMS = dict(G=np.arange(0.1, 1., 0.15))
```

Here are some templates that you can use right after import statements. The list will keep getting updated as the app grows.

TheVirtualBrain (TVB) users

```
# set required fields for current TVB version
app.SoftwareVersion = 2.6
app.SoftwareRepository = 'https://github.com/the-virtual-brain/tvb-root/releases/
↪tag/2.6'
app.SoftwareName = 'TVB'
```

```
# set required fields for older TVB versions, e.g. 1.5.10
app.SoftwareVersion = '1.5.10'
app.SoftwareRepository = 'https://github.com/the-virtual-brain/tvb-root/releases/
↪tag/1.5.10'
app.SoftwareName = 'TVB'
```

Warning: Please specify model parameters if your input code meets one of the following conditions:

- non-Python code (e.g., MATLAB, R, Julia)
- Python code with more than one rhythm-specific parameters (e.g., separate parameters for alpha and delta rhythms)

- Python code with a list of parameters (for parameter exploration), e.g., G values from 0.1 to 1.0 with a step of 0.15

Run the app

There are two ways to run the app:

Run locally

When you run the app locally (=not on a server, cluster, or anything of the sort), the app creates a localhost page in a new tab that will render the app. The page should have a name like this *http://localhost:58838/*, of course, with different numbers. Please note that the numbers will keep changing every time you run the app.

Here is the snippet to run the app:

```
pn.serve(MainArea().view())
```

Note: The app performs best if ran locally. It will open up a new tab running on a local host. It's a known problem in the HoloViz community (the package the app built on) that the components **do not** get rendered well if ran inline.

Run on a server

When you run the app on a server/cluster, you will need to run the app inline. The localhost will be created but won't be accessible. That's why it's recommended to run it inline.

Please note that this approach might not work properly because of the rendering issues. You might see text blocked but input fields or not be able to do select folders. If you encounter that, please keep restarting the notebook until the issue is fixed.

Here is the snippet to run the app:

```
MainArea().view().servable()
```

Note: We recommend saving all your simulations created on a server and running the app locally for best performance.

Complete script

Run locally

```
import sim2bids
import panel as pn
from sim2bids.sim2bids import MainArea
pn.extension('tabulator', 'ace', 'jsoneditor', 'ipywidgets', sizing_mode='stretch_
↪width', notifications=True)

# set required fields
sim2bids.app.app.SoftwareVersion = 2.6
sim2bids.app.app.SoftwareRepository = 'https://github.com/the-virtual-brain/tvb-
↪root/releases/tag/2.6'
sim2bids.app.app.SoftwareName = 'TVB'

pn.serve(MainArea().view())
```

Run on a server

```
import sim2bids
import panel as pn
from sim2bids.sim2bids import MainArea
pn.extension('tabulator', 'ace', 'jsoneditor', 'ipywidgets', sizing_mode='stretch_
↪width', notifications=True)

# set required fields
sim2bids.app.app.SoftwareVersion = 2.6
sim2bids.app.app.SoftwareRepository = 'https://github.com/the-virtual-brain/tvb-
↪root/releases/tag/2.6'
sim2bids.app.app.SoftwareName = 'TVB'

MainArea().view().servable()
```

1.2 Input files and structures

1.2.1 Accepted file extensions

- Text files (.txt)
- Tab-separated files (.tsv)
- Generic data files (.dat)
- Numpy array (.npy)
- MATLAB (.mat) files (v4 and v6-7.2 using scipy.io.loadmat, v7.3 using mat73 package)
- HDF5 (.h5) files
- zip folders containing all above file extensions

1.2.2 Required and recommended fields

Please note the following rules and recommendations:

- The most important file to have for conversions is **Structural Connectome** which is must be named `weights`.
- **Recommended files:**
  - `centres` (or `nodes` and `labels` separately),
  - `distances` (or `tract_lengths` which is the other name for distances, thus this file will be renamed to distances both in the input and output folders)
  - `Python`, `MATLAB` or `R` code that can reproduce the results
  - `empirical` and `simulated` time series and time stamps.
- `average_orientation` or `orientation` will be renamed to `normals` according to BEP034 both in the input and output folders.

1.2.3 Accepted files

Here is the list of files that are supported by the app categorized by their respective folders.

Table 1: Coordinates (`coord` folder)

File name	Description	Dimensions	Notes
centres	consists of nodes (nx1 vector) and labels (nx3 matrix) in that order.	nx4	See description of `nodes` and `labels` below
nodes	These are the region labels (e.g., lh_bankssts, lh_superiorfrontal).	nx1	recommended
delays	These are the 3d coordinate centres	nx3	recommended
times	These are the time steps of the simulated time series.	nx1	recommended
bold_times	These are the time steps of the simulated BOLD time series.	nx1	recommended
areas	This is the estimated vector each region's area in mm^2.	nx1	optional
cortical	This is the vector that distinguishes cortical (1) from subcortical (0) regions.	nx1	optional
average orientations, normals	normals.	nx3	optional. If the file name is `average_orientation`, it will be named `normals` in input and output folders.
hemisphere	The vector that distinguishes right (1) from left (0) hemisphere.	nx1	optional.
faces	These are the faces of cortex surface triangulation.		optional.
vertices	These are the vertices of cortex surface triangulation.		optional.
map	This is the nxm matrix where the coordinates along rows are mapped to the coordinates along columns.	nxm	optional.

1.3 How to use the app

1.3.1 Preprocessing pipeline

1.3.2 App layout & features

Layout

When you run the app locally, you should see the following layout as shown below. There's going to be a small difference in the local and inline layouts but the functionality remains the same.

get_started/..\protect \T1\textunderscore static.png

_static.png

Selecting Files

Pears are green.

Oranges

Oranges are orange.