Sumatra: a toolkit for provenance capture and reuse

Andrew Davison
Unité de Neurosciences, Information et Complexité (UNIC)
CNRS, Gif sur Yvette, France

@apdavison
http://www.andrewdavison.info

Reproducibility in Computational and Experimental Mathematics
ICERM, Providence, RI. December 13th 2012
“I thought I used the same parameters but I’m getting different results”

“I can’t remember which version of the code I used to generate figure 6”

“The new student wants to reuse that model I published three years ago but he can’t reproduce the figures”

“It worked yesterday”

“Why did I do that?”
Why isn’t it easy to reproduce a computational experiment exactly?
Why isn’t it easy to reproduce a computational experiment exactly?

- **complexity**
  - dependence on small details, small changes have big effects

- **entropy**
  - computing environment, library versions change over time

- **human memory limitations**
  - forgetting, implicit knowledge not passed on
What can we do about it?

- **complexity**
  - use/teach good software-engineering practices
    - (loose coupling, testing...)

- **entropy**
  - plan for reproducibility from the start: run in different environments, write tests, record dependencies

- **human memory limitations**
  - record everything
What can we do about it?

- complexity
  - use/teach good software-engineering practices
    (loose coupling, testing...)

- entropy
  - plan for reproducibility from the start: run in different environments, write tests, record dependencies

- human memory limitations
  - record everything
• what code was run?
  – which executable?
    * name, location, version, compilation options
  – which script?
    * name, location, version
    * options, parameters
    * dependencies (name, location, version)
• what were the input data?
  – name, location, content
• what were the outputs?
  – data, logs, stdout/stderr
• who launched the computation?
• when was it launched/when did it run? (queueing systems)
• where did it run?
  – machine name(s), other identifiers (e.g. IP addresses)
  – processor architecture
  – available memory
  – operating system
• why was it run?
• what was the outcome?
• which project was it part of?
Recording all this by hand is tedious and error-prone

let’s automate it
Requirements

Different researchers, different workflows

- command-line
- GUI
- batch jobs
- solo or collaborative
- any combination of these for different components and phases of the project
Requirements

- Integrate into the day-to-day workflow

- Be very easy to use, or only the very conscientious will use it

\[ \text{Requirements} \]

\[ \text{Integrate into the day-to-day workflow} \]

\[ \text{Be very easy to use, or only the very conscientious will use it} \]
A core library of loosely-coupled components

Used to build interfaces:

• command-line interface for launching and capturing computations
• graphical interface for browsing/searching results
• remote server for sharing/communicating with others
• documentation-system interface for including results-with-provenance in publications
• integration with existing tools...
Installation

➢ Install Python bindings for your preferred version control system (pysvn, mercurial, GitPython, bzrlib)

➢ `pip install sumatra`
Command-line interface

$ cd myproject

$ smt init MyProject
$ python main.py default.param
$ python main.py default.param

$ smt run --executable=python --main=main.py default.param
$ python main.py default.param

$ smt run --executable=python --main=main.py default.param

$ smt configure --executable=python --main=main.py
$ python main.py default.param

$ smt run --executable=python --main=main.py default.param

$ smt configure --executable=python --main=main.py

$ smt run default.param
$ smt run default.param
$ smt run default.param
Code has changed, please commit your changes.
$ smt run default.param
Code has changed, please commit your changes.

$ smt configure --on-changed=store-diff
$ smt run default.param
Code has changed, please commit your changes.

$ smt configure --on-changed=store-diff

$ smt run default.param
create new record

has the code changed?

yes

code change policy

diff

store diff

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

raise exception

error

has the code changed?
```plaintext
$ smt list
20110713-174949
20110713-175111

$ smt list -l

<table>
<thead>
<tr>
<th>Label</th>
<th>20110713-174949</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>2011-07-13 17:49:49.235772</td>
</tr>
<tr>
<td>Reason</td>
<td></td>
</tr>
<tr>
<td>Outcome</td>
<td></td>
</tr>
<tr>
<td>Duration</td>
<td>0.0548920631409</td>
</tr>
<tr>
<td>Repository</td>
<td>MercurialRepository at /path/to/myproject</td>
</tr>
<tr>
<td>Main file</td>
<td>main.py</td>
</tr>
<tr>
<td>Version</td>
<td>rf9ab74313efe</td>
</tr>
<tr>
<td>Script arguments</td>
<td>&lt;parameters&gt;</td>
</tr>
<tr>
<td>Executable</td>
<td>Python (version: 2.6.2) at /usr/bin/python</td>
</tr>
<tr>
<td>Parameters</td>
<td>seed = 65785</td>
</tr>
<tr>
<td></td>
<td>distr = &quot;uniform&quot;</td>
</tr>
<tr>
<td></td>
<td>n = 100</td>
</tr>
<tr>
<td>Input_Data</td>
<td>[]</td>
</tr>
<tr>
<td>Launch_Mode</td>
<td>serial</td>
</tr>
<tr>
<td>Output_Data</td>
<td>[example2.dat(43a47cb379df2a7008fdeb38c6172278d000fd)]</td>
</tr>
<tr>
<td>Tags</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
$ smt run --label=haggling --reason="determine whether the
gourd is worth 3 or 4 shekels" romans.param
$ smt comment "apparently, it is worth NaN shekels."
"Eureka! Fields Medal here we come."
$ \text{smt tag "Figure 6"}$
$ smt run --reason="test effect of a smaller time constant" default.param tau_m=10.0
$ smt repeat haggling
The new record exactly matches the original.
The new record does not match the original. It differs as follows.

Record 1                : haggling
Record 2                : haggling_repeat
Executable differs      : no
Code differs            : yes
  Repository differs   : no
  Main file differs    : no
  Version differs      : no
  Non checked-in code  : no
  Dependencies differ  : yes
Launch mode differs     : no
Input data differ       : no
Script arguments differ : no
Parameters differ       : no
Data differ             : no
$ smt
Usage: smt <subcommand> [options] [args]

Simulation/analysis management tool, version 0.4

Available subcommands:
  init
  configure
  info
  run
  list
  delete
  comment
  tag
  repeat
  diff
  help
  upgrade
  export
  sync
Browser interface

$ smtweb -p 8008 &
Browser interface
Output files

/Data
20121025/MV_HFV_012_170722_phases.png  c955f84ca3c19123d24ccc4c87d197514d9e01e  image/png 82.2 KB
20121025/MV_HFV_012_170722_sand.png  20bd5420d37ee589f3e2542a12438cb78663974b  image/png 34.3 KB
20121025/MV_HFV_012_170722_histogram.png e7884dc5f3e9ce2c0d713cb3e9ddff0b07bf2010c  image/png 27.7 KB

Parameters

no parameters

Dependencies

<table>
<thead>
<tr>
<th>Name</th>
<th>Path</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>dateutil</td>
<td>/usr/lib/pymodules/python2.6/dateutil</td>
<td>1.4.1</td>
</tr>
<tr>
<td>glib</td>
<td>/usr/lib/pymodules/python2.6/gtk-2.0/glib</td>
<td>unknown</td>
</tr>
<tr>
<td>gobject</td>
<td>/usr/lib/pymodules/python2.6/gtk-2.0/gobject</td>
<td>unknown</td>
</tr>
<tr>
<td>matplotlib</td>
<td>/usr/lib/pymodules/python2.6/matplotlib</td>
<td>0.99.11</td>
</tr>
<tr>
<td>mpl_toolkits</td>
<td>/usr/lib/pymodules/python2.6/mpl_toolkits</td>
<td>unknown</td>
</tr>
<tr>
<td>numpy</td>
<td>/usr/lib/python2.6/dist-packages/numpy</td>
<td>1.3.0</td>
</tr>
<tr>
<td>pytz</td>
<td>/usr/lib/python2.6/dist-packages/pytz</td>
<td>2010b</td>
</tr>
<tr>
<td>scipy</td>
<td>/usr/lib/python2.6/dist-packages/scipy</td>
<td>0.7.0</td>
</tr>
<tr>
<td>wx</td>
<td>/usr/lib/python2.6/dist-packages/wx-2.8-gtk2-unicode/wx</td>
<td>2.8.10.1 (gtk2-unicode)</td>
</tr>
</tbody>
</table>

Platform information

<table>
<thead>
<tr>
<th>Name</th>
<th>IP address</th>
<th>Processor</th>
<th>Architecture</th>
<th>System type</th>
<th>Release</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu01</td>
<td>127.0.1.1</td>
<td>i686</td>
<td>32bit ELF</td>
<td>Linux</td>
<td>2.6.32-24-generi</td>
<td>#39-Ubuntu SMP Wed Jul 28 06:07:29 UTC 2010</td>
</tr>
</tbody>
</table>

Stdout & Stderr

```
/usr/lib/python2.6/dist-packages/numpy/lib/function_base.py:272: DeprecationWarning:
The histogram semantics being used is now deprecated and
will disappear in NumPy 1.4. Please update your code to
use the default semantics.

*****, DeprecationWarning)
1699.875 65.0
```
<table>
<thead>
<tr>
<th>Reason</th>
<th>Parameters are now in a separate file</th>
<th>Fixed to work with the new numpy.histogram0 function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>25/10/2012 17:28:33</td>
<td>26/10/2012 17:45:45</td>
</tr>
<tr>
<td>Duration</td>
<td>3.85s</td>
<td>3.85s</td>
</tr>
<tr>
<td>Executable</td>
<td>Python version 2.6.5 (/usr/bin/python)</td>
<td>Python version 2.7.3 (/usr/bin/python)</td>
</tr>
<tr>
<td>Launch mode</td>
<td>serial</td>
<td>serial</td>
</tr>
<tr>
<td>Repository</td>
<td>/home/bob/Projects/Glass</td>
<td>/home/bob/Projects/Glass</td>
</tr>
<tr>
<td>Main file</td>
<td>glass_sem_analysis.py</td>
<td>glass_sem_analysis.py</td>
</tr>
<tr>
<td>Version</td>
<td>432ff7ef3f45</td>
<td>924fa39a0d24c</td>
</tr>
</tbody>
</table>

File name: 20121025/MV_HFV_012_172836_phases.png

File name: 20121026/MV_HFV_012_174557_phases.png

Digest: c9955f84a3c19123d24ccc4c87d197514d9e01e

Digest: 7f8ed0c6ef97b8317af8e2d9ab9f856a193c2687

Dependencies:

<table>
<thead>
<tr>
<th>Name</th>
<th>Path</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>dateutil</td>
<td>/usr/lib/pymodules/python2.6/dateutil</td>
<td>1.4.1</td>
</tr>
<tr>
<td>glib</td>
<td>/usr/lib/pymodules/python2.6/gtk-2.0/glib</td>
<td>unknown</td>
</tr>
<tr>
<td>gobject</td>
<td>/usr/lib/pymodules/python2.6/gtk-2.0/gobject</td>
<td>unknown</td>
</tr>
<tr>
<td>matplotlib</td>
<td>/usr/lib/pymodules/python2.6/matplotlib</td>
<td>0.99.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Path</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIL</td>
<td>/usr/lib/python2.7/dist-packages/PIL</td>
<td>unknown</td>
</tr>
<tr>
<td>PyQt4</td>
<td>/usr/lib/python2.7/dist-packages/PyQt4</td>
<td>unknown</td>
</tr>
<tr>
<td>apt</td>
<td>/usr/lib/python2.7/dist-packages/apt</td>
<td>unknown</td>
</tr>
<tr>
<td>dateutil</td>
<td>/usr/lib/python2.7/dist-packages/dateutil</td>
<td>1.5</td>
</tr>
<tr>
<td>glib</td>
<td>/usr/lib/python2.7/dist-packages/glib</td>
<td>unknown</td>
</tr>
</tbody>
</table>
### Dependencies:

<table>
<thead>
<tr>
<th>Name</th>
<th>Path</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>dateutil</td>
<td>/usr/lib/pymodules/python2.6/dateutil</td>
<td>1.4.1</td>
</tr>
<tr>
<td>glib</td>
<td>/usr/lib/pymodules/python2.6/glib</td>
<td>unknown</td>
</tr>
<tr>
<td>gobject</td>
<td>/usr/lib/pymodules/python2.6/gobject</td>
<td>unknown</td>
</tr>
<tr>
<td>matplotlib</td>
<td>/usr/lib/pymodules/python2.6/matplotlib</td>
<td>0.99.1.1</td>
</tr>
<tr>
<td>mpl_toolkits</td>
<td>/usr/lib/pymodules/python2.6/mpl_toolkits</td>
<td>unknown</td>
</tr>
<tr>
<td>numpy</td>
<td>/usr/lib/python2.6/dist-packages/numpy</td>
<td>1.3.0</td>
</tr>
<tr>
<td>pytz</td>
<td>/usr/lib/python2.6/dist-packages/pytz</td>
<td>2010b</td>
</tr>
<tr>
<td>scipy</td>
<td>/usr/lib/python2.6/dist-packages/scipy</td>
<td>0.7.0</td>
</tr>
<tr>
<td>wx</td>
<td>/usr/lib/python2.6/dist-packages/wx-2.8-gtk2-unicode/wx</td>
<td>2.8.10.1 (gtk2-unicode)</td>
</tr>
</tbody>
</table>

### Platform information:

1. Name: Ubuntu01
2. IP address: 127.0.0.1
3. Processor: i686
4. Architecture: 32bit
5. System type: Linux
6. Release: 2.6.32-24-generic

1. Name: Ubuntu01
2. IP address: 127.0.0.1
3. Processor: i686
4. Architecture: 32bit
5. System type: Linux
6. Release: 3.2.0-32-generic
advantage that the network can be parallelized using MPI. Otherwise, the only important difference between ``multiAMPAexp`` and ``NetCon`` is that the former has a dead time of one millisecond after a conductance step in which any incoming spikes have no effect.

```
$ hg update -r 7                      # replaced multiAMPAexp with ExpSyn
$ python demo_cx05_N=500b_LTS.py
$ python plot.py spiketimes_cx05_LTS500b.dat numspikes_cx05_LTS500b.dat ..
```

Despite this difference, the models give comparable results.

```
.. smtimage:: 20120919-173558
  :digest: 26f6ad85aab0ef1e995042c0a3b3029e303a90a6
```
dynamics uses NEURON's NetCon mechanism to communicate, which has the advantage that the network can be parallelized using MPI. Otherwise, the only important difference between multiAMPAexp and NetCon is that the former has a dead time of one millisecond after a conductance step in which any incoming spikes have no effect.

```
$ hg update -r 7  # replaced multiStimexp, multiAMPAexp and multiGABAexp with ExpSyn
$ python demo_cx05_N=500b_LTS.py
$ python plot.py spiketimes_cx05_LTS500b.dat numspikes_cx05_LTS500b.dat Vm170_cx05_LTS500b.dat
```

Despite this difference, the models give comparable results.
import numpy
import sys

def main(parameters):
    numpy.random.seed(parameters["seed"])
    distr = getattr(numpy.random, parameters["distr"])
    data = distr(size=parameters["n"])  
    output_file = "Data/example.dat"
    numpy.savetxt(output_file, data)

parameter_file = sys.argv[1]
parameters = {}
execfile(parameter_file, parameters)  # this way of reading parameters
                                       # is not necessarily recommended
main(parameters)
import numpy
import sys
from sumatra.parameters import build_parameters
from sumatra.decorators import capture

@capture
def main(parameters):
    numpy.random.seed(parameters["seed"])
    distr = getattr(numpy.random, parameters["distr"])
    data = distr(size=parameters["n"])
    output_file = "Data/%s.dat" % parameters["sumatra_label"]
    numpy.savetxt(output_file, data)

parameter_file = sys.argv[1]
parameters = build_parameters(parameter_file)
main(parameters)
Sumatra components
Code versioning and dependency tracking

the code, the whole code and nothing but the code

1. Recursively find imported/included libraries
2. Try to determine version information for each of these, using
   1. code analysis
   2. version control systems
   3. package managers
   4. etc.
Code versioning and dependency tracking
the code, the whole code and nothing but the code

sumatra.dependency_finder.python
sumatra.dependency_finder.matlab
sumatra.dependency_finder.R
sumatra.dependency_finder.fortran

sumatra.versioncontrol.subversion
sumatra.versioncontrol.mercurial
sumatra.versioncontrol.git
sumatra.versioncontrol.bazaar
Launching computations
locally, remotely, serial or parallel

sumatra.launch

SerialLaunchMode
  generate_command()
  get_platform_information()
  run()

DistributedLaunchMode
  generate_command()
  get_platform_information()
  run()

BatchLaunchMode
  generate_command()
  get_platform_information()
  run()

QueuedLaunchMode
  generate_command()
  get_platform_information()
  run()
## Parameter handling

### Simple

```
a = 2
b = 3
c = [4, 5, 6]
```

### Config

```
[foo]
a: 2
b: 3

[bar]
c: [4, 5, 6]
```

### JSON

```
{
  'foo': {
    'a': 2,
    'b': 3
  },
  'bar': {
    'c': [4, 5, 6]
  }
}
```

---

*sumatra.parameters*
Data handling
telling Sumatra where to find the data generated by your code and what to do with it

• Data generated on local file system
  ➡ FileSystemDataStore

• Data on local file system and automatically archived
  ➡ ArchivingFileSystemDataStore

• Data on local file system and mirrored to web (e.g. Dropbox)
  ➡ MirroredFileSystemDataStore

• Data generated in a relational database
  ➡ RelationalDataStore

• Data automatically pushed to FigShare
  ➡ FigShareDataStore
Storing provenance information
for solo or collaborative projects

- local filesystem

- remote server

**ShelveRecordStore**
- list_projects()
- create_project()
- save()
- get()
- list()
- delete()
- delete_by_tag()

**DjangoRecordStore**
- list_projects()
- create_project()
- save()
- get()
- list()
- delete()
- delete_by_tag()

**HttpRecordStore**
- list_projects()
- create_project()
- save()
- get()
- list()
- delete()
- delete_by_tag()
Remote record store

RESTful API (JSON over HTTP):

/                                               GET
/<project_name>/[?tags=<tag1>,<tag2>,...]       GET
/<project_name>/tagged/<tag>/           GET, DELETE
/<project_name>/<record_label>/    GET, PUT, DELETE
/<project_name>/permissions/              GET, POST
Network simulations of self-sustained activity in networks of adaptive exponential integrate and fire neurons

The aim of this project is to convert an original NEURON model written by Alain Destexhe, published in the following article:


from Hoc to PyNN. Using PyNN makes it possible to simulate the model using the NEST and Brain simulators, and using the FACETS / BrainScaleS neuromorphic hardware, as well as NEURON.

The project is also intended as a case study of converting a NEURON model to PyNN.

<table>
<thead>
<tr>
<th>Label</th>
<th>Reason</th>
<th>Outcome</th>
<th>Duration</th>
<th>Date/Time</th>
<th>Executable</th>
<th>Repository</th>
<th>Version</th>
<th>Script</th>
<th>Args</th>
</tr>
</thead>
<tbody>
<tr>
<td>20120928-114020</td>
<td>Plot data from the last simulation run with NEURON</td>
<td></td>
<td>2.20 s</td>
<td>2012-09-28 11:40:20</td>
<td>Python 2.7.1</td>
<td><a href="https://bitbucket.org/apdavison/destexhe_cns_2009">https://bitbucket.org/apdavison/destexhe_cns_2009</a></td>
<td>5f6221aa3a14</td>
<td>plot.py</td>
<td>spiknum</td>
</tr>
<tr>
<td>20120928-113952</td>
<td>Plot data from the last simulation run with NEST</td>
<td></td>
<td>2.26 s</td>
<td>2012-09-28 11:39:52</td>
<td>Python 2.7.1</td>
<td><a href="https://bitbucket.org/apdavison/destexhe_cns_2009">https://bitbucket.org/apdavison/destexhe_cns_2009</a></td>
<td>5f6221aa3a14</td>
<td>plot.py</td>
<td>spiknum</td>
</tr>
<tr>
<td>20120928-110307</td>
<td>Plot data from the previous simulation run</td>
<td></td>
<td>2.49 s</td>
<td>2012-09-28 11:03:07</td>
<td>Python 2.7.1</td>
<td><a href="https://bitbucket.org/apdavison/destexhe_cns_2009">https://bitbucket.org/apdavison/destexhe_cns_2009</a></td>
<td>05844ae81d7c</td>
<td>plot.py</td>
<td>spiknum</td>
</tr>
<tr>
<td>20120928-105047</td>
<td>Running pure PyNN version with NEURON backend</td>
<td>Network structure different due to different random number generator. Mean firing</td>
<td>59.93 s</td>
<td>2012-09-28 10:50:47</td>
<td>Python 2.7.1</td>
<td><a href="https://bitbucket.org/apdavison/destexhe_cns_2009">https://bitbucket.org/apdavison/destexhe_cns_2009</a></td>
<td>05844ae81d7c</td>
<td>demo_cx05_N=500b_LTS.py</td>
<td>neuron</td>
</tr>
</tbody>
</table>
Remote record store

The aim of this project is to convert an original NEURON model written by Alain Destexhe, published in the following article:


_NeuroPython_ (PyNN): http://neuralensemble.org/PyNN/

_NEST_ : http://www.nest-initiative.org/

_Brian_ : http://brian simulator.org/

_BrainScaleS_:

http://brain scales.kip.uni-heidelberg.de/
Remote record store

Clients:

- browser
- HttpRecordStore (part of sumatra package)
- curl

Server implementations:

- Django-based (https://bitbucket.org/apdavison/sumatra_server/)
Plans / Ideas

• Dependency finders for R, Fortran, C/C++, Ruby

• Better support for projects with build steps/integration with build tools

• LaTeX package

• Export in W3C PROV-XML or PROV-O format

• Better support for pipelines

• Support for parameter searching ("smt batch")

• IPython Notebook integration?

• Export of recipes enabling recreation of environment

• Alternative web views, e.g. diary format - more like a traditional lab notebook
Community

- 6 contributors (including 1 GSoC student)
- mailing list has 39 members
- previous version had 1222 downloads in 18 months, current version 248 downloads in two months
- http://neuralensemble.org/sumatra
- (mirror) https://bitbucket.org/apdavison/sumatra
Sumatra Simulation Management Tool

http://neuralensemble.org/sumatra
Sumatra
Simulation Management Tool
Computational Experiment

http://neuralensemble.org/sumatra
Sumatra

Nothing to do with Java
Sumatra

Not a million miles from Madagascar

Indian Ocean map by Tentotwo https://commons.wikimedia.org/wiki/File:Indian_Ocean_laea_location_map.svg
Conclusions

To be accepted by busy scientists, a tool to assist with making research more reproducible should:

- be part of day-to-day workflow
- be easy to use
- require minimal changes to existing workflows
- provide immediate benefit

As tool developers, we should think about making as much as possible of our functionality available as libraries, so others can find new ways to use it