Automated tracking of computational experiments using Sumatra

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

Reproducible Research: Tools and Strategies for Scientific Computing
Reproducibility
Replicability

- Reproduction of the original results using the same tools
  by the original author on the same machine

- Reproduction using different software, but with access to the original code
  by someone in the same lab/using a different machine

- Completely independent reproduction based only on text description, without access to the original code
  by someone in a different lab

Reproducibility
Completely independent reproduction based only on text description, without access to the original code

Reproduction using different software, but with access to the original code

Reproduction of the original results using the same tools

by the original author on the same machine

by someone in the same lab/using a different machine

by someone in a different lab
Replicability

http://www.flickr.com/photos/36266791@N00/70150248/
“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

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

> memory limitations

record everything
What do we need to record?

› the code that was run
› how it was run (parameter files, input data, command-line options)
› the platform on which it was run
› why was it run?
› what was the outcome? (output data, figures, qualitative interpretation)
Recording the code that was run

- store a copy of the executable
- or of the source code
- including that of any libraries used
- as well as the compiler used
- and the compilation procedure
Recording the code that was run

- the version of the interpreter
- and any options used in compiling it
- a copy of the simulation script
- and of any external modules or packages that are imported/included
Recording the code that was run instead of storing a copy of the code we can store the repository URL and version number.
Recording platform information

- processor architecture
- operating system
- number of processors
Recording all this by hand is tedious and error-prone
### A Story Told in File Names:

Location: `C:\user\research\data`

<table>
<thead>
<tr>
<th>Filename</th>
<th>Date Modified</th>
<th>Size</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>data_2010.05.28_test.dat</td>
<td>3:37 PM</td>
<td>420 KB</td>
<td>DAT file</td>
</tr>
<tr>
<td>data_2010.05.28_re-test.dat</td>
<td>4:29 PM</td>
<td>421 KB</td>
<td>DAT file</td>
</tr>
<tr>
<td>data_2010.05.28_re-re-test.dat</td>
<td>5:43 PM</td>
<td>420 KB</td>
<td>DAT file</td>
</tr>
<tr>
<td>data_2010.05.28_calibrate.dat</td>
<td>7:17 PM</td>
<td>1,256 KB</td>
<td>DAT file</td>
</tr>
<tr>
<td>data_2010.05.28_huh???.dat</td>
<td>7:20 PM</td>
<td>30 KB</td>
<td>DAT file</td>
</tr>
<tr>
<td>data_2010.05.28_WTF.dat</td>
<td>9:58 PM</td>
<td>30 KB</td>
<td>DAT file</td>
</tr>
<tr>
<td>data_2010.05.29_aaarrrgh.dat</td>
<td>12:37 AM</td>
<td>30 KB</td>
<td>DAT file</td>
</tr>
<tr>
<td>data_2010.05.29_#$@*%&amp;!!.dat</td>
<td>2:40 AM</td>
<td>0 KB</td>
<td>DAT file</td>
</tr>
<tr>
<td>data_2010.05.29_crap.dat</td>
<td>3:22 AM</td>
<td>437 KB</td>
<td>DAT file</td>
</tr>
<tr>
<td>data_2010.05.29_notbad.dat</td>
<td>4:16 AM</td>
<td>670 KB</td>
<td>DAT file</td>
</tr>
<tr>
<td>data_2010.05.29_woohoo!!.dat</td>
<td>4:47 AM</td>
<td>1,349 KB</td>
<td>DAT file</td>
</tr>
<tr>
<td>data_2010.05.29_USETHISONE.dat</td>
<td>5:08 AM</td>
<td>2,894 KB</td>
<td>DAT file</td>
</tr>
<tr>
<td>analysis_graphs.xls</td>
<td>7:13 AM</td>
<td>455 KB</td>
<td>XLS file</td>
</tr>
<tr>
<td>ThesisOutline!.doc</td>
<td>7:26 AM</td>
<td>38 KB</td>
<td>DOC file</td>
</tr>
<tr>
<td>Notes_Meeting_with_ProfSmith.txt</td>
<td>11:38 AM</td>
<td>1,673 KB</td>
<td>TXT file</td>
</tr>
<tr>
<td>JUNK...</td>
<td>2:45 PM</td>
<td></td>
<td>Folder</td>
</tr>
<tr>
<td>data_2010.05.30_startingover.dat</td>
<td>8:37 AM</td>
<td>420 KB</td>
<td>DAT file</td>
</tr>
</tbody>
</table>
Recording all this by hand is tedious and error-prone

let’s automate it
What should this automated lab notebook look like?
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

- automate as much as possible, prompt the user for the rest
- interact with version control systems (Subversion, Git, Mercurial, Bazaar ...)
- support serial, distributed, batch simulations/analyses
- link to data generated by the simulation/analysis
- support all and any (command-line driven) simulation/analysis programs
- support both local and networked storage of simulation/analysis records
Requirements

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

Heated at 850°C for 10 minutes in a crucible.

Crucible exploded.

No sample.
Sumatra

- a Python package to enable systematic capture of the environment of numerical simulations/analyses
- can be used directly in your own code
- or as the basis for interfaces
Current

- a command line interface, smt
- a web interface, smtweb

Future

- could be integrated into existing GUI-based tools
- or new desktop/web-based GUIs written from scratch
Sumatra Simulation Management Tool

http://neuralensemble.org/sumatra

Sawahs in West Sumatra by CharlesFred http://www.flickr.com/photos/charlesfred/2869003149/
Sumatra Simulation Management Tool
Computational Experiment

http://neuralensemble.org/sumatra
Sumatra

Nothing to do with Java
Dependencies

- Python bindings for your preferred version control system (*pysvn*, *mercurial*, *PyGit*, *bzr-lib*)

- Django (only needed for web interface)

- mpi4py (if running distributed computations), *httplib2*
Installation

`easy_install sumatra`
smt

$ cd myproject

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

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

$ smt run default.param

$ smt run --simulator=python --main=main.py default.param
create new record

has the code changed?

yes

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

code change policy

error

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?

no

find dependencies

get platform information

run simulation/analysis

record time taken

find new files

add tags

save record

no

raise exception

store diff

diff

has the code changed?
$ 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(43a47cb379df2a7008fdeb38c6172278d000f26078c3276)]</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 \text{ comment } "\text{apparently, it is worth NaN shekels.}"
$ smt comment 20110713-174949 "Eureka! Nobel prize here we come."
$ 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.
$ smt info
Sumatra project
----------------

<table>
<thead>
<tr>
<th>Name</th>
<th>MyProject</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default executable</td>
<td>Python (version: 2.6.2) at /usr/bin/python</td>
</tr>
<tr>
<td>Default repository</td>
<td>MercurialRepository at /path/to/myproject rf9ab74313efe (main file is main.py)</td>
</tr>
<tr>
<td>Default main file</td>
<td>main.py</td>
</tr>
<tr>
<td>Default launch mode</td>
<td>serial</td>
</tr>
<tr>
<td>Data store (output)</td>
<td>./Data</td>
</tr>
<tr>
<td>(input)</td>
<td>/</td>
</tr>
<tr>
<td>Default launch mode</td>
<td>serial</td>
</tr>
<tr>
<td>Record store</td>
<td>Relational database record store using the Django ORM (database file=.smt/records)</td>
</tr>
<tr>
<td>Code change policy</td>
<td>error</td>
</tr>
<tr>
<td>Append label to</td>
<td>None</td>
</tr>
</tbody>
</table>
$ 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
$ smtweb -p 8002 &
<table>
<thead>
<tr>
<th>Label</th>
<th>Reason</th>
<th>Duration</th>
<th>Processes</th>
<th>Simulator Name</th>
<th>Repository</th>
<th>Script File</th>
<th>Version</th>
<th>Date</th>
<th>Time</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>20100709-154235</td>
<td>'Eureka! Nobel prize here we come.'</td>
<td>0.59 s</td>
<td>Python 2.5.2</td>
<td>main.py</td>
<td>/Users/andrew/tmp/SumatraTest</td>
<td>96c2020ca50</td>
<td>09/07/2010</td>
<td>15:42:55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20100709-154309</td>
<td>'determine whether the gourd is worth 3 or 4 shekels'</td>
<td>0.59 s</td>
<td>Python 2.5.2</td>
<td>main.py</td>
<td>/Users/andrew/tmp/SumatraTest</td>
<td>96c2020ca50</td>
<td>09/07/2010</td>
<td>15:43:09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>haggling</td>
<td>'apparently, it is worth NaN shekels.'</td>
<td>0.59 s</td>
<td>Python 2.5.2</td>
<td>main.py</td>
<td>/Users/andrew/tmp/SumatraTest</td>
<td>96c2020ca50</td>
<td>09/07/2010</td>
<td>15:43:20</td>
<td>foobar</td>
<td></td>
</tr>
<tr>
<td>20100709-154338</td>
<td>'test effect of a smaller time constant'</td>
<td>0.59 s</td>
<td>Python 2.5.2</td>
<td>main.py</td>
<td>/Users/andrew/tmp/SumatraTest</td>
<td>96c2020ca50</td>
<td>09/07/2010</td>
<td>15:43:38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>haggling_repeat</td>
<td>Repeat experiment haggling</td>
<td>0.58 s</td>
<td>Python 2.5.2</td>
<td>main.py</td>
<td>/Users/andrew/tmp/SumatraTest</td>
<td>96c2020ca50</td>
<td>09/07/2010</td>
<td>15:43:47</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Return to record list

Label: haggling
Reason: 'determine whether the gourd is worth 3 or 4 shekels'
Outcome: 'apparently, it is worth NaN shekels.'

Timestamp: 09/07/2010 15:43:20
Duration: 0.59 s
Executable: Python version 2.5.2 (/usr/local/bin/python)
Launch mode: serial
Repository: /Users/andrew/tmp/SumatraTest
Main file: main.py
Version: 396c2020ca50
Tags: foobar

Data files
/Users/andrew/tmp/SumatraTest/Data
exmaple2.dat 1.2 KB

Parameters
n = 50
seed = 34326
distr = normal

Dependencies
<table>
<thead>
<tr>
<th>Name</th>
<th>Path</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carbon</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/plat-mac/Carbon</td>
<td>unknown</td>
</tr>
<tr>
<td>Finder</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/Py2.5/plat-mac/lib-scriptpackages/Finder</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>Carbon</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/plat-mac/Carbon</td>
<td>unknown</td>
</tr>
<tr>
<td>Finder</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/plat-mac/lib-scriptpackages/Finder</td>
<td>unknown</td>
</tr>
<tr>
<td>PyQt4</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/PyQt4</td>
<td>unknown</td>
</tr>
<tr>
<td>Pyrex</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/Pyrex-0.9.8.5.0001-py2.5.egg/Pyrex</td>
<td>unknown</td>
</tr>
<tr>
<td>StdSuites</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/StdSuites</td>
<td>unknown</td>
</tr>
<tr>
<td>_builtinSuites</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/plat-mac/lib-scriptpackages/_builtinSuites</td>
<td>unknown</td>
</tr>
<tr>
<td>_xmlplus</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/PyXML-0.8.4.0003-py2.5-macosx-10.3-fat.egg/_xmlplus</td>
<td>0.8.4</td>
</tr>
<tr>
<td>dateutil</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/python_dateutil-1.4.0001-py2.5.egg/dateutil</td>
<td>1.4</td>
</tr>
<tr>
<td>entthought</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/TraitsGUI-3.0.2-py2.5.egg/entthought</td>
<td>unknown</td>
</tr>
<tr>
<td>matplotlib</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/matplotlib-0.98.3.0001-py2.5-macosx-10.3-fat.egg/matplotlib</td>
<td>0.98.3</td>
</tr>
<tr>
<td>mpl_toolkits</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/mpl_toolkits</td>
<td>unknown</td>
</tr>
<tr>
<td>numarray</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/numarray-1.5.2.0001-py2.5-macosx-10.3-fat.egg/numarray</td>
<td>1.5.2</td>
</tr>
<tr>
<td>numpy</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/numpy-1.1.1.0001-py2.5-macosx-10.3-fat.egg/numpy</td>
<td>1.1.1</td>
</tr>
<tr>
<td>pytz</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/pytz-2008c.0001-py2.5.egg/pytz</td>
<td>2008c</td>
</tr>
<tr>
<td>setuptools</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/setuptools-0.6c8.0002-py2.5.egg/setuptools</td>
<td>0.6c8</td>
</tr>
<tr>
<td>wx</td>
<td>/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/wxPython-2.8.7.1.0001_s-py2.5-macosx-10.3-fat.egg/wx</td>
<td>2.8.7.1 (mac-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>
</table>
### example2.dat

<table>
<thead>
<tr>
<th>Value</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.907044147908750314e-01</td>
<td></td>
</tr>
<tr>
<td>9.4770417401008509741e-02</td>
<td></td>
</tr>
<tr>
<td>1.675740714241656715e+00</td>
<td></td>
</tr>
<tr>
<td>8.097734317494595314e-01</td>
<td></td>
</tr>
<tr>
<td>-3.930586580052130481e-01</td>
<td></td>
</tr>
<tr>
<td>1.135962474942340410e+00</td>
<td></td>
</tr>
<tr>
<td>-3.365427212023050082e-01</td>
<td></td>
</tr>
<tr>
<td>-1.619924991387704220e-01</td>
<td></td>
</tr>
<tr>
<td>4.502319146511131598e-01</td>
<td></td>
</tr>
<tr>
<td>-6.22947130613413597e-01</td>
<td></td>
</tr>
<tr>
<td>-8.746021252920153908e-01</td>
<td></td>
</tr>
<tr>
<td>-1.022833101028255300e+00</td>
<td></td>
</tr>
<tr>
<td>1.631837671625364194e+00</td>
<td></td>
</tr>
<tr>
<td>7.07115339369577372e-01</td>
<td></td>
</tr>
<tr>
<td>-3.85494957160916838e+00</td>
<td></td>
</tr>
<tr>
<td>-1.938590502404218929e+00</td>
<td></td>
</tr>
<tr>
<td>-3.840079543034994069e-01</td>
<td></td>
</tr>
<tr>
<td>2.5467226464789079e-01</td>
<td></td>
</tr>
<tr>
<td>-1.590639681557504126e+00</td>
<td></td>
</tr>
<tr>
<td>9.755707717898647705e-01</td>
<td></td>
</tr>
<tr>
<td>3.394566109543630954e-01</td>
<td></td>
</tr>
<tr>
<td>1.546769138105918984e+00</td>
<td></td>
</tr>
<tr>
<td>-5.11740902801874121e-01</td>
<td></td>
</tr>
<tr>
<td>-1.863397017272495683e+00</td>
<td></td>
</tr>
<tr>
<td>6.56361556481744408e-01</td>
<td></td>
</tr>
<tr>
<td>1.05391519630216102e+00</td>
<td></td>
</tr>
<tr>
<td>2.70896780028555607e-01</td>
<td></td>
</tr>
<tr>
<td>-5.929809662700129458e-01</td>
<td></td>
</tr>
<tr>
<td>2.11793323489190226e+00</td>
<td></td>
</tr>
<tr>
<td>5.41196608757018065e-01</td>
<td></td>
</tr>
<tr>
<td>-6.02982754744773822e-01</td>
<td></td>
</tr>
<tr>
<td>-5.801808657982524793e-02</td>
<td></td>
</tr>
<tr>
<td>-3.70392935285061842e-01</td>
<td></td>
</tr>
<tr>
<td>-1.04902119461476934e+00</td>
<td></td>
</tr>
<tr>
<td>8.45006748157527792e-02</td>
<td></td>
</tr>
<tr>
<td>-5.192887644952853715e-01</td>
<td></td>
</tr>
<tr>
<td>-9.214658744538642536e-01</td>
<td></td>
</tr>
<tr>
<td>5.349458364200462279e-01</td>
<td></td>
</tr>
<tr>
<td>-5.85941068844664182e-01</td>
<td></td>
</tr>
</tbody>
</table>
Using sumatra within your own scripts

```python
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
import time
from sumatra.projects import load_project
from sumatra.parameters import build_parameters

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)

project = load_project()
record = project.new_record(parameters=parameters,
                             main_file=__file__,
                             reason="reason for running this simulation")
parameters.update({'sumatra_label': record.label})
start_time = time.time()
main(parameters)

record.duration = time.time() - start_time
record.output_data = record.datastore.find_new_data(record.timestamp)
project.add_record(record)

project.save()
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)
Supported parameter file formats

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]  
  }  
}
```
Finding dependencies

- requires per-language implementation

  currently supported: Python, Hoc, GENESIS script language

  planned: Matlab, Octave, R (collaborators wanted...)

- version finding based on various heuristics:

  some language specific (e.g. in Python check for __version__,
  get_version(), ...)

  some generic (where dependency code is under version control
  system, managed by package manager...)
Linking to input and output data

▶ Intention to support different data stores (filesystem, relational database, ...)

▶ Stores SHA1 digest of data to ensure file contents haven’t changed

▶ smtweb has ‘smart’ display for certain data types (e.g. csv is displayed as HTML table)
multiple ways to store experiment records, to support both solo/local and collaborative/distributed projects:

- simple (no dependencies, does not support smtweb)
- Django/SQLite-based (default)
- remote (HTTP+JSON)
Remote record store & Sumatra Server

- RESTful API (JSON over HTTP):

  /<project_name>/[?tags=<tag1>,<tag2>,...] GET
  /<project_name>/tagged/<tag>/ GET, DELETE
  /<project_name>/<record_label>/ GET, PUT, DELETE
  /<project_name>/permissions/ GET, POST

- Client: HttpRecordStore (part of sumatra package)

- Server: Django site including sumatra_server app (https://bitbucket.org/apdavison/sumatra_server/)

  (Bartosz Telenczuk has also started to implement a MongoDB-based server https://github.com/btel/Sumata-MongoDB)
Version control

- your code is required to be under version control
- currently supports Subversion, Git, Mercurial, Bazaar

Discussion points:

- could just store copy of code, but want to promote best practice
- version control by stealth?
Plans
(contributions of code and ideas welcome)

➢ determine compilation options for executable, where possible
➢ determine version, compilation options for Python C-extensions, shared libraries more generally (interact with Linux package managers?)
➢ Implement other DataStores, e.g. based on FTP, HDF5, SQLite, DropBox
➢ Add dependency_finder sub-modules for Matlab, Octave, R, C/C++
➢ add remote launch option (ssh-based)
➢ implement BatchLaunchMode
➢ add support for simple workflows (running multiple computations sequentially)
➢ enable launching computations from the web interface
Summary

Sumatra

- a toolbox for automated metadata capture for computational experiments
- basic metadata captured for any language, logging dependencies requires language-specific plugin

smt

- requires no changes to existing code
- requires minimal changes to workflow
- “Be very easy to use, or only the very conscientious will use it”
Sumatran orangutan

http://neuralensemble.org/sumatra

@apdavison

http://www.andrewdavison.info

by BelalangJantan http://www.flickr.com/photos/7164478@N07/3575735482/