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
AMP 2011, Vancouver. July14 2011

Reproducibility

attack of the clone santas by slowburn, http://www.flickr.com/photos/36266791@N00/70150248/

Replicability ⟷ Reproducibility

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

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

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

Replicability ⟷ Reproducibility

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

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

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

Replicability

attack of the clone santas by slowburn, 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

Filename	Date Modified	Size	Type
data_2010.05.28_test.dat	3:37 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_re-test.dat	4:29 PM 5/28/2010	421 KB	DAT file
data_2010.05.28_re-re-test.dat	5:43 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_calibrate.dat	7:17 PM 5/28/2010	1,256 KB	DAT file
data_2010.05.28_huh??.dat	7:20 PM 5/28/2010	30 KB	DAT file
data_2010.05.28_WTF.dat	9:58 PM 5/28/2010	30 KB	DAT file
data_2010.05.29_aaarrrgh.dat	12:37 AM 5/29/2010	30 KB	DAT file
data_2010.05.29_#$@*&!!.dat	2:40 AM 5/29/2010	0 KB	DAT file
data_2010.05.29_crap.dat	3:22 AM 5/29/2010	437 KB	DAT file
data_2010.05.29_notbad.dat	4:16 AM 5/29/2010	670 KB	DAT file
data_2010.05.29_woohoo!!.dat	4:47 AM 5/29/2010	1,349 KB	DAT file
data_2010.05.29_USETHISONE.dat	5:08 AM 5/29/2010	2,894 KB	DAT file
analysis_graphs.xls	7:13 AM 5/29/2010	455 KB	XLS file
ThesisOutline!.doc	7:26 AM 5/29/2010	38 KB	DOC file
Notes_Meeting_with_ProfSmith.txt	11:38 AM 5/29/2010	1,673 KB	TXT file
JUNK...	2:45 PM 5/29/2010		Folder
data_2010.05.30_startingover.dat	8:37 AM 5/30/2010	420 KB	DAT file

Type: Ph.D Thesis Modified: too many times

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

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

http://neuralensemble.org/sumatra

Sawahs in West Sumatra by CharlesFred http://www.flickr.com/photos/charlesfred/2869003149/

Sumatra

Simulation Management Tool

http://neuralensemble.org/sumatra

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`, `bzrlib`)
- Django (only needed for web interface)
- mpi4py (if running distributed computations), httplib2

Installation

```
> easy_install sumatra
```

smt

```
$ cd myproject
```

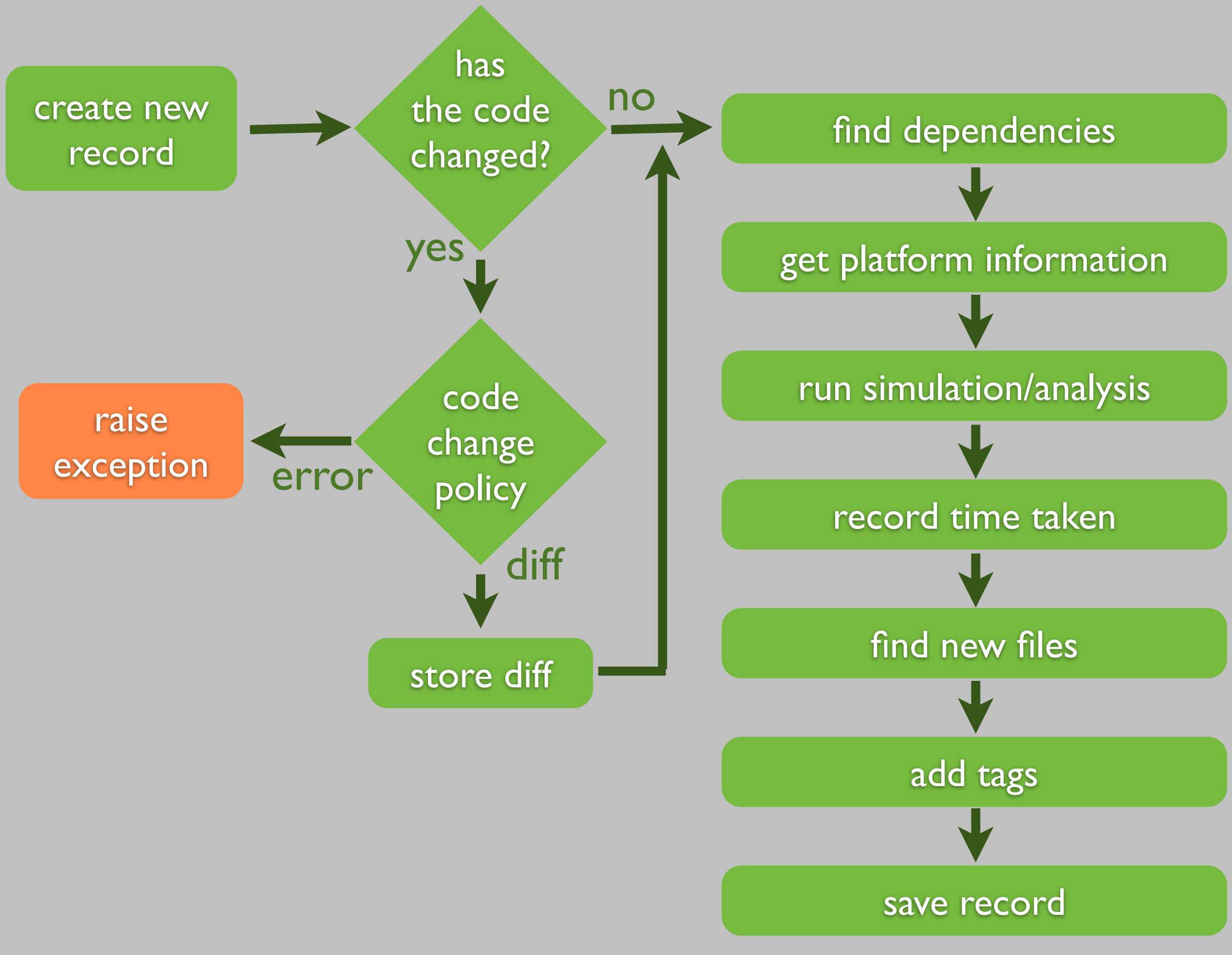
```
$ smt init MyProject
```

```
$ 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?
no
yes
find dependencies
get platform information
run simulation/analysis
record time taken
find new files
add tags
save record
code change policy
error
raise exception
diff
store diff

```
$ smt list
20110713-174949
20110713-175111

$ smt list -l
-------------------------------------------------
Label                : 20110713-174949
Timestamp            : 2011-07-13 17:49:49.235772
Reason               :
Outcome              :
Duration             : 0.0548920631409
Repository           : MercurialRepository at /path/to/myproject
Main file            : main.py
Version              : rf9ab74313efe
Script arguments     : <parameters>
Executable           : Python (version: 2.6.2) at /usr/bin/python
Parameters           : seed = 65785
                     : distr = "uniform"
                     : n = 100
Input_Data           : []
Launch_Mode          : serial
Output_Data          :[example2.dat(43a47cb379df2a7008fdeb38c6172278d000f
Tags                 :
.
.
.
```

```
$ 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."
```

```
$ 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
---------------
Name                   : MyProject
Default executable     : Python (version: 2.6.2) at /usr/bin/python
Default repository     : MercurialRepository at /path/to/myproject
                         rf9ab74313efe (main file is main.py)
Default main file      : main.py
Default launch mode    : serial
Data store (output)    : ./Data
.          (input)     : /
Default launch mode    : serial
Record store           : Relational database record store using the
                         Django ORM (database file=.smt/records)
Code change policy     : error
Append label to        : None
```

```
$ 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 &
```

Sumatra: TestProject: List of records

http://127.0.0.1:8002/

Google

TestProject: List of records

Delete include data	Label	Reason	Outcome	Duration	Processes	Simulator		Script			Date	Time	Tags
						Name	Version	Repository	Main file	Version			
☐	20100709-154255		'Eureka! Nobel prize here we come.'	0.59 s		Python	2.5.2	/Users/andrew/tmp/SumatraTest	main.py	396c2020ca50	09/07/2010	15:42:55	
☐	20100709-154309			0.59 s		Python	2.5.2	/Users/andrew/tmp/SumatraTest	main.py	396c2020ca50	09/07/2010	15:43:09	
☐	haggling	'determine whether the gourd is worth 3 or 4 shekels'	'apparently, it is worth NaN shekels.'	0.59 s		Python	2.5.2	/Users/andrew/tmp/SumatraTest	main.py	396c2020ca50	09/07/2010	15:43:20	foobar
☐	20100709-154338	'test effect of a smaller time constant'		0.59 s		Python	2.5.2	/Users/andrew/tmp/SumatraTest	main.py	396c2020ca50	09/07/2010	15:43:38	
☐	haggling_repeat	Repeat experiment haggling	The new record exactly matches the original.	0.58 s		Python	2.5.2	/Users/andrew/tmp/SumatraTest	main.py	396c2020ca50	09/07/2010	15:43:47	

Sumatra: TestProject: haggling

http://127.0.0.1:8002/haggling/

Google

TestProject: *haggling*

Save changes | Delete

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

example2.dat 1.2 KB

Parameters

n = 50
seed = 34326
distr = normal

Dependencies

Name	Path	Version
Carbon	/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/plat-mac/Carbon	unknown
Finder	/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/plat-mac/lib-scriptpackages/Finder	unknown

Dependencies

Name	Path	Version
Carbon	/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/plat-mac/Carbon	unknown
Finder	/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/plat-mac/lib-scriptpackages/Finder	unknown
PyQt4	/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/PyQt4	unknown
Pyrex	/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/Pyrex-0.9.8.5.0001-py2.5.egg/Pyrex	unknown
StdSuites	/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/plat-mac/lib-scriptpackages/StdSuites	unknown
_builtinSuites	/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/plat-mac/lib-scriptpackages/_builtinSuites	unknown
_xmlplus	/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	0.8.4
dateutil	/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/python_dateutil-1.4.0001-py2.5.egg/dateutil	1.4
enthought	/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/TraitsGUI-3.0.2-py2.5.egg/enthought	unknown
matplotlib	/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	0.98.3
mpl_toolkits	/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/mpl_toolkits	unknown
numarray	/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	1.5.2
numpy	/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	1.1.1
pytz	/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/pytz-2008c.0001-py2.5.egg/pytz	2008c
setuptools	/Library/Frameworks/Python.framework/Versions/4.0.30002/lib/python2.5/site-packages/setuptools-0.6c8.0002-py2.5.egg/setuptools	0.6c8
wx	/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	2.8.7.1 (mac-unicode)

Platform information

Name	IP address	Processor	Architecture	System type	Release	Version
dhcp-12-56.ens.fr	129.199.12.56	i386 i386	32bit	Darwin	9.8.0	Darwin Kernel Version 9.8.0: Wed Jul 15 16:55:01 PDT 2009; root:xnu-1228.15.4~1/RELEASE_I386

TestProject: *haggling*

Return to record

example2.dat

```
7.907044147908758314e-01
9.477014701088509741e-02
1.675740714241656715e+00
8.087734934749459814e-01
-3.930586588052103481e-01
1.135906247492430410e+00
-3.364521723023050082e-01
-1.618924993187704220e-01
4.502319146511311598e-01
-6.229447130613413597e-01
-8.749602125292015309e-01
-1.022383310028285530e+00
1.631837671625364194e+00
7.071715339639573772e-01
-3.857494957160916838e-01
-1.938590502404218929e+00
-3.840079545304994069e-01
2.543672264647489079e-01
-1.590639681557504126e+00
9.755700771898647705e-01
3.394566100954360954e-01
1.546769138105918984e+00
-5.617409082801874121e-01
-1.863397012724956836e+00
6.564861554841714408e-01
1.053391199630216102e+00
2.709896780028555607e-01
-5.829809662700129458e-01
2.117353834389190226e+00
5.411866008757018065e-01
-6.029287547744773823e-01
-5.801084967982524793e-02
-3.708293525285061842e-01
-1.048902119461476934e+00
-9.423948053155544180e-01
8.458066748157527792e-02
-5.192887844952853715e-01
-9.214658744583642536e-01
5.349458364200462279e-01
```

Using `sumatra` within your own scripts

```
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)

Record stores

- 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):

`/`	`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`

- 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”

http://neuralensemble.org/sumatra

@apdavison

http://www.andrewdavison.info

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

Project
name
description
default_executable
default_repository
default_main_file
default_launch_mode
data_store
record_store
on_changed
data_label
save()
info()
new_record()
launch()
update_code()
add_record()
get_record()
delete_record()
delete_by_tag()
format_records()
most_recent()
add_comment()
add_tag()
remove_tag()
compare()
show_diff()

RecordStore
save()
get()
list()
labels()
delete()
delete_by_tag()
most_recent()

Executable
name
path
version

Record
label
timestamp
reason
duration
executable
repository
main_file
version
parameters
input_data
script_arguments
launch_mode
datastore
outcome
output_data
dependencies
platforms
tags
diff
user
on_changed
register()
run()
describe()
difference()
delete_data()

Dependency
name
path
version
diff

WorkingCopy
path
repository
current_version()
use_version()
use_latest_version()
status()
has_changed()
diff()

Repository
url
checkout()
get_working_copy()

DataKey
path
digest

DataStore
find_new_data()
archive()
get_data_item()
get_content()
delete()

ParameterSet
save()
update()
__getitem__()
pretty()

LaunchMode
generate_command()
get_platform_information()
pre_run()
run()

Project
name
description
default_executable
default_repository
default_main_file
default_launch_mode
data_store
record_store
on_changed
data_label
save()
info()
new_record()
launch()
update_code()
add_record()
get_record()
delete_record()
delete_by_tag()
format_records()
most_recent()
add_comment()
add_tag()
remove_tag()
compare()
show_diff()

PythonExecutable
name
path
version

NEURONSimulator
name
path
version

NESTSimulator
name
path
version

ShelveRecordStore
save()
get()
list()
labels()
delete()
delete_by_tag()
most_recent()

DjangoRecordStore
save()
get()
list()
labels()
delete()
delete_by_tag()
most_recent()

HttpRecordStore
save()
get()
list()
labels()
delete()
delete_by_tag()
most_recent()

Record
label
timestamp
reason
duration
executable
repository
main_file
version
parameters
input_data
script_arguments
launch_mode
datastore
outcome
output_data
dependencies
platforms
tags
diff
user
on_changed
register()
run()
describe()
difference()
delete_data()

Dependency
name
path
version
diff
on_changed

SubversionWorkingCopy
path
repos
current
use_v
use_l
status
has_c
diff()

MercurialWorkingCopy
path
repos
current
use_ve
use_la
status(
has_ch
diff()

GitWorkingCopy
path
repository
current_version()
use_version()
use_latest_version()
status()
has_changed()
diff()

SubversionRepository
url
wo
che

MercurialRepository
url
w
ch

GitRepository
url
working_copy
checkout()

FileSystemDataStore
find_new_files()
archive()
list_files()
get_content()

SimpleParameterSet
save()
update()
__g
pre

NTParameterSet
save()
upd
__g
pre

JSONParameterSet
sav
up
__
pr

ConfigParser ParameterSet
save()
update()

SerialLaunchMode
generate_command()
get
pre
ru

DistributedLaunchMode
generate_command()
get_platform_information()
pre_run()
run()