Using Git and Emacs for Managing Experiments

Luka Stanisic, Arnaud Legrand and Jean-François Méhaut

MESCAL and NANOSIM teams, Grenoble

June 4, 2013
After some time even authors very often don’t know how and why they did an experiment.

Describing every experiment and making it repeatable (reproducible) takes a lot of work.

Computational experiments require specific hardware, non-trivial software installation, and complex manipulations to obtain results.

Real measurements are even worse as results can be unstable, environment changes etc.

**GOAL**: Do experiments in a clean, coherent, systematic and reproducible way.
Experiment Workflow

1. Modifications of source code
Experiment Workflow

1. Modifications of source code
2. Execute experiments
Experiment Workflow

1. Modifications of source code
2. Execute experiments
3. Store data

![Diagram showing the workflow with nodes labeled source, experiment, and data connected by arrows.]
Experiment Workflow

1. Modifications of source code
2. Execute experiments
3. Store data
4. Analyze data
Experiment Workflow

1. Modifications of source code
2. Execute experiments
3. Store data
4. Analyze data
5. Repeat the process
**Experiment Workflow**

1. Modifications of source code
2. Execute experiments
3. Store data
4. Analyze data
5. Repeat the process
6. Publish results
Experiment Workflow - Challenges

1. Source: self-content source directory, several external source repositories, external dependencies
2. Experiment: deployment scripts, compilation, kernel settings, external dependencies
3. Data: storing large files, connect data and source that produced it
4. Analysis: storing large results, how and why they were generated
5. Repetition: keeping track of previous steps
6. Publish: articles, slides, web-page etc.

- IDEALLY: Go quickly through steps and add new features, but at the same time keep track of everything and easily replay the experiments
Outline

1 Reproducible Research Tools
   - CDE
   - Sumatra
   - VisTrails
   - Collage
   - VCR

2 Literate programming
   - Sweave/Knitr
   - IPython
   - Org-babel

3 Using Git for Reproducible Research

4 Using Emacs and Org-mode for Managing Experiments

5 Conclusion
Outline

1. Reproducible Research Tools
   - CDE
   - Sumatra
   - VisTrails
   - Collage
   - VCR

2. Literate programming
   - Sweave/Knitr
   - IPython
   - Org-babel

3. Using Git for Reproducible Research

4. Using Emacs and Org-mode for Managing Experiments

5. Conclusion
Creating a package with cde

```bash
cd /home/pg/expt/
cde python predict_weather.py
```

```
cde-package/
```

```
/usr/bin/python
/usr/lib/libpython2.6.so
predict_weather.py
/usr/bin/R
/usr/local/R/stdlib.R
/usr/local/R/weatherMod.so
weather_models.R
/usr/bin/WeatherSim
```

```
cde-root/usr/bin/python
cde-root/usr/lib/libpython2.6.so
cde-root/home/pg/expt/predict_weather.py
cde-root/usr/bin/R
cde-root/usr/local/R/stdlib.R
cde-root/usr/local/R/weatherMod.so
cde-root/home/pg/expt/weather_models.R
cde-root/usr/bin/WeatherSim
```
Executing a package with cde-exec

cd cde-package/cde-root/home/pg/expt/
cde-exec python predict_weather.py
Pros and Cons

- **Author**: automatically generates completely reproducible environment
- **Reviewer**: executing a CDE package requires no installation, setup, or root permissions
- **Good way to avoid dependency hell**
Pros and Cons

- **Author:** automatically generates completely reproducible environment
- **Reviewer:** executing a CDE package requires no installation, setup, or root permissions
- **Good way to avoid dependency hell**
- **Cannot change the experiment design (only datasets used)**
- **No workflow or description of experiments**
- **Execution is slower and packages might be incomplete**
- **Works only for x86 and Linux 2.6**
### General info

- **Label**: 20121017-133232
- **Reason**: Eurolai Nobel prize here we come.
- **Outcome**: 
- **Timestamp**: 17/10/2012 13:32:32
- **Duration**: 0.12s
- **Executable**: Python version 2.7.1 ([home/andrew/en/song_plenary/2012/python2.7/bin/python](<BR>))
- **Launch mode**: serial
- **Repository**: [home/andrew/en/song_plenary](<BR>)
- **Main file**: main.py
- **Version**: d14972b346d31306a565ca8eb8c55b56b78
- **Arguments**: `<parameters>`
- **Tags**: 

### Parameters

- **1019**
  - **Name**: 45246
  - **Value**: 1019

### Dependencies

<table>
<thead>
<tr>
<th>Name</th>
<th>Path</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>distutils</td>
<td><a href="BR">home/andrew/en/song_plenary/py27_distutils</a></td>
<td>2.7.1</td>
</tr>
<tr>
<td>encodings</td>
<td><a href="BR">home/andrew/en/song_plenary/py27_encodings</a></td>
<td>unknown</td>
</tr>
<tr>
<td>nose</td>
<td><a href="BR">home/andrew/en/song_plenary/py27_nose</a></td>
<td>1.1.2</td>
</tr>
<tr>
<td>numpy</td>
<td><a href="BR">home/andrew/en/song_plenary/py27_numpy</a></td>
<td>1.6.2</td>
</tr>
</tbody>
</table>

### Platform information

- **Name**: redhat
  - **IP address**: 197.5.11.1
  - **Processor**: x86_64 x86_64
  - **Architecture**: 64bit ELF
  - **System type**: Linux
  - **Release**: 2.6.38-11-generic
  - **Version**: i586 Ubuntu SMP Mon Sep 12 21:17:25 UTC 2011

### Stdout & Stderr

No output.
## Comparison of simulations

Click the records you would like to compare:

<table>
<thead>
<tr>
<th>20121017-133237</th>
<th>haggling</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Reason</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Duration</th>
<th>Executable</th>
<th>Launch mode</th>
<th>Repository</th>
<th>Main file</th>
<th>Version</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>17/10/2012 13:32:37</td>
<td>0.12s</td>
<td>Python version 2.7.1</td>
<td>serial</td>
<td>/home/andrew/tmp/example</td>
<td>main.py</td>
<td>d1497e294b4df5f03a45e65a9f40c56b8e874</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tags</th>
<th>Input files</th>
<th>Output files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>no input files</td>
<td>/Data/output.dat ccb626788c2c793304ed39ed9ad3b3a99 2.4 KB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Path</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>distutils</td>
<td>/home/andrew/env/sumatra_py27_django13/lib/python2.7/distutils</td>
<td>2.7.1</td>
</tr>
<tr>
<td>encodings</td>
<td>/home/andrew/env/sumatra_py27_django13/lib/python2.7/encodings</td>
<td>unknown</td>
</tr>
<tr>
<td>nose</td>
<td>/home/andrew/env/sumatra_py27_django13/lib/python2.7/site-packages/nose</td>
<td>1.1.2</td>
</tr>
<tr>
<td>numpy</td>
<td>/home/andrew/env/sumatra_py27_django13/lib/python2.7/site-packages/numpy</td>
<td>1.6.2</td>
</tr>
</tbody>
</table>

Platform Information:

1. Name: relina
2. IP address: 177.0.1.1
3. Processor: x86_64 x86_64
4. Architector: x86_64

1. Name: relina
2. IP address: 177.0.1.1
3. Processor: x86_64 x86_64
4. Architector: x86_64
Pros and Cons

- Automatically keeps track of many information for you (command line options, lib version, source revision, experiment parameters etc.)
- Nice way for basic comparing of multiple experiment
- Convenient for daily usage
- Source code is under version control system
- Support parallel computation - MPI

Limited description level for the reasons and summaries of experiment, now work support
Tracked information may be hard to extend
Exploiting meta-information is not that convenient
Limited support for automatically finding dependencies
Finally Sumatra does not guarantee reproducibility
Pros and Cons

- Automatically keeps track of many information for you (command line options, lib version, source revision, experiment parameters etc.)
- Nice way for basic comparing of multiple experiment
- Convenient for daily usage
- Source code is under version control system
- Support parallel computation - MPI
- Limited description level for the reasons and summaries of experiment, no workflow support
- Tracked information may be hard to extend
- Exploiting meta-information is not that convenient
- Limited support for automatically finding dependencies
- Finally Sumatra does not guarantee reproducibility
VisTrails
Pros and Cons

- Has good workflow description and supposedly parallel support
- Powerful for complex computational science with complicated workflows
Pros and Cons

- Has good workflow description and supposedly parallel support
- Powerful for complex computational science with complicated workflows
- Not keeping track of anything (code has to be linked with git manually)
- Not providing a reproducible environment
- Documentation is inside each module
Collage Authoring Environment

[Image of the Collage Authoring Workbench interface]

New experiment: Shape Retrieval with the Signature Quadratic Form Distance

- Copying shapes: Bash with collage-e2.elsevier.com
- Computing mat files: Matlab R2010b with collage-e2.elsevier.com
- Computing dissimilarity: Matlab R2010b with collage-e2.elsevier.com
Pros and Cons

- Enabling reviewers and readers to reexamine the reported results by executing the parts of code that generated it as well as access raw data
- Produces reproducible environment
- Same comprehensible interface for both authors and readers, which facilitates reviewing process
Pros and Cons

- Enabling reviewers and readers to reexamine the reported results by executing the parts of code that generated it as well as access raw data
- Produces reproducible environment
- Same comprehensible interface for both authors and readers, which facilitates reviewing process
- More oriented to final publishing than for daily usage
- Oriented to the computational science not real measurements
- There is no version control system for the code
- Comparing to VisTrails, it assumes simple sequential workflow, with no patterns (all file names are hardcoded)
Figure 2

Table 1

(a) Pattern of gene expression in dorsal fibroblasts at senescence.

<table>
<thead>
<tr>
<th></th>
<th>BJ</th>
<th>C1</th>
<th>MA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5%FBS</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>10%FBS</td>
<td>2.2</td>
<td>2.2</td>
<td>2.2</td>
</tr>
</tbody>
</table>

(b) Pattern of gene expression in retinal pigment epithelial cells at senescence.

<table>
<thead>
<tr>
<th></th>
<th>RPE340</th>
<th>RPE340</th>
<th>RPE340</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1%FBS</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>10%FBS</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Discussion

Senescence-associated gene expression

All three cell strains reported here undergo an arrest characterized by a distinctive cell-cycle distribution, changes in cell morphology, and the shortening of telomeres. Senescent cells have a cell-cycle distribution that is clearly different from that of quiescent cells, as a substantial population of cells with G2 DNA content persists [9] (Figure 2). In high-serum conditions, this arrest includes expression of the CDK inhibitor p21, and growth arrest specific protein (Gas1) in BJ fibroblasts and RPE340 strains, but not in HUVECs. The overexpression of cyclin D2 was observed in all three cell strains and may reflect a common cell-cycle braking mechanism [19,20]. Interestingly, in none of the cell strains examined did we observe an upregulation of mRNA for the CDK inhibitor p16, as reported for other cell strains [21,22], and RT-PCR analysis of mRNA from senescent BJ fibroblasts confirmed the lack of induction of mRNA for p16 (data not shown). The induction of p21 at senescence has been reported to be transient in some cell lines, whereas the induction of p16 can occur after prolonged arrest [23], and this may also account for our observed results with different cell lines. The data presented here strongly suggest that even with a common initiating signal — shortened telomeres — the ensuing arrest may be triggered by dif-

whereas in senescent cultures the response was attenuated (Figure 3A). Similarly, mRNAs for G1R, tristetraprol-
Pros and Cons

- Automatically brands each publishable result produced by the computation
- Easy to use for both authors and readers
Pros and Cons

- Automatically brands each publishable result produced by the computation
- Easy to use for both authors and readers
- Completely oriented to publishing in computational science - nothing about workflows
- Does not guarantee reproducibility
- Very integrated, same issues about description and extension as Sumatra
- Limited documentation on web-site, doubt that project is still active
Outline

1. Reproducible Research Tools
   - CDE
   - Sumatra
   - VisTrails
   - Collage
   - VCR

2. Literate programming
   - Sweave/Knitr
   - IPython
   - Org-babel

3. Using Git for Reproducible Research

4. Using Emacs and Org-mode for Managing Experiments

5. Conclusion
**Sweave/Knitr**

---

**Kernel from MultiMAPS**

```
CPU ↔ L1 ↔ L2 ↔ L3 ↔ MEMORY
```

MultiMAPS(size, stride, nloops)
- allocate buffer size;
- timer start;
- for(i=1:nloops)
  - access elements in buffer by *stride*;
- timer stop;
- bandwidth = #accesses/time;
- deallocate buffer;

---

**SONGS plenary: WP8**

Luka Stanisic

June 4, 2013

20 / 42
Pros and Cons

- Pdf reports can be updated automatically if data or analysis change
- Analysis is done only at the moment of report compilation
- Provide reproducible papers
Pros and Cons

- Pdf reports can be updated automatically if data or analysis change
- Analysis is done only at the moment of report compilation
- Provide reproducible papers
- Files can become quite cumbersome, hard to read or browse
- Compilation time can be significant (especially first time)
- It is more oriented to "static" analysis
Simple spectral analysis
An illustration of the Discrete Fourier Transform

\[ X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \quad k = 0, \ldots, N - 1 \]

using windowing, to reveal the frequency content of a sound signal.

We begin by loading a datafile using SciPy’s audio file support:

```python
In [1]: from scipy.io import wavfile
      : rate, x = wavfile.read('test_mono.wav')
```

And we can easily view its spectral structure using matplotlib’s built-in `specgram` routine:

```python
In [2]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
      : ax1.plot(x);
      : ax1.set_title('Raw audio signal')
      : ax2.specgram(x);
      : ax2.set_title('Spectrogram');
```

![Raw audio signal and Spectrogram](image)
Pros and Cons

- Convenient for daily usage
- In web-browser, very descriptive, easy start for new users
Pros and Cons

- Convenient for daily usage
- In web-browser, very descriptive, easy start for new users
- Cut and paste is nontrivial
- Using other programming languages besides python is possible, but it is not always straightforward
- Little control on block layout
- No hierarchy for different sections of the whole document
- No workflow support
Improving Simulations of MPI Applications Using A Hybrid Network Model with Topology and Contention Support

Paul Bédardie
Loria/NRCA, Univ. of Nancy
Nancy, France
paul.bedardie@loria.fr

Arnaud Legrand
CNRS/NRCA, Univ. of Grenoble
Grenoble, France
arnaud.lagrang@imag.fr

Mark Stillwell
Engineering Computing, Cranfield University, U.K.
M.Stillwell@cranfield.ac.uk

Augustin Degomme
CNRS/NRCA, Univ. of Grenoble
Grenoble, France
Augustin.Degomme@imag.fr

George S. Markomanolis
INRIA LIP ENS Lyon
Lyon, France
gsm@lip6.fr

Frédéric Oster
NP3I Computing Center, CNRS
Lyon-Vieurbanne, France
foster@incs.ens-lyon.fr

Stéphane Genaud
University of Strasbourg, France
genaud@unistra.fr

Martin Quinson
Loria/NRCA, Univ. of Nancy
Nancy, France
martin.quinson@loria.fr

Briol Vidone
CNRS/Univ. of Grenoble
Grenoble, France
briol.vidone@imag.fr

1. INTRODUCTION

Correctly modeling collective communications, including their low-level breakdown into individual point-to-point messages and how these messages interact and interfere with one another within the network environment, is essential for understanding the high-level behavior of medium and large scale parallel applications. In this work we demonstrate how even relative minor deviations in low-level implementation can adversely affect the ability of simulations to predict real-world performance, and propose a new network model that extends previous LogP based approaches to account for topology and contention in high-speed TCP networks. This part of a larger effort to study distributed computing platform behavior on realistic objects, recognizing that the complexity of such systems makes it increasingly difficult to reason from a purely theoretical viewpoint. Indeed, the computation and communication behavior of many, if not most, modern scientific codes is understood at only the most basic level, even by application programmers. We propose to study these systems through simulation based on empirically validated models, thus refining the tools and techniques afforded by advanced computational resources back on these resources themselves.

This approach is further justified by the recent recognition of computational science and simulation as an essential “third pillar” of scientific methodology, on par with and distinct from experimentation and theory. Well thought out simulations can provide results that are more easily understood and analyzed than a physical experiment at significantly lower costs. Further, because of the difficulty in running the precise configuration of the experimental environment, in some fields the results from simulation are regarded as more accurate and informative than those from physical experiments. A notable example of this can be seen in the field of fluid mechanics, where the overall importance of micro-bubbles in reducing drag coefficients for moving objects (such as ships)...

Luka Stanisic
Pros and Cons

- Powerful combination of org features with its hierarchy and other programming languages
- Convenient for daily usage
- Some control of block layout and workflow
Pros and Cons

- Powerful combination of org features with its hierarchy and other programming languages
- Convenient for daily usage
- Some control of block layout and workflow
- Limited to Emacs users
- Has a long learning curve
- Error management is painful
- Computation intensive chunks freeze Emacs
- For now there is no support for parallel runs
Outline

1 Reproducible Research Tools
   - CDE
   - Sumatra
   - VisTrails
   - Collage
   - VCR

2 Literate programming
   - Sweave/Knitr
   - IPython
   - Org-babel

3 Using Git for Reproducible Research

4 Using Emacs and Org-mode for Managing Experiments

5 Conclusion
Many people use git to manage source code, but we want to store everything - source, data, analysis, scripts.

Using git for provenance, tracking dependencies in git history.

It is accomplished with small code addition to all scripts that are running experiments + script for merging branches.

There are two main branches:

1. **Src**: source code including analysis source code (R, Knitr, Org-babel)
2. **Data**: source code + data + analysis reports
2 Branches
Source Modifications

\begin{tikzpicture}

% Define nodes


\end{tikzpicture}
New Branch for Experimentation

```
src data
exp
```

Luka Stanisic
SONGS plenary: WP8
June 4, 2013 30 / 42
Analyzing Results
Merging Branches
New Experiment
Possible Conflicts

Cherry-Pick

Source modifications

Revert

src
data

exp
Pros and Cons

- Provides good provenance tracking
- Easy to reconfigure or extend
- Provides all benefits of git to the source code development
- Good combination with lab notebook
- Works fine on remote machines (ARM)
- Analysis and merging can be done on a local computer
Pros and Cons

- Provides good provenance tracking
- Easy to reconfigure or extend
- Provides all benefits of git to the source code development
- Good combination with lab notebook
- Works fine on remote machines (ARM)
- Analysis and merging can be done on a local computer

- Approach is not obvious to new users
- Code needs to be manually added to all necessary scripts
- Takes time to setup everything properly
Outline

1. Reproducible Research Tools
   - CDE
   - Sumatra
   - VisTrails
   - Collage
   - VCR

2. Literate programming
   - Sweave/Knitr
   - IPython
   - Org-babel

3. Using Git for Reproducible Research

4. Using Emacs and Org-mode for Managing Experiments

5. Conclusion
Laboratory Notebook

- Laboratory notebooks are used by biologist, chemists and all other scientists to document experiments and performed operations.

- Roles of lab notebook are:
  1. Write down hypothesis
  2. Analysis or interpretation of experiments
  3. Organization tool
  4. Memory aid

- Open Notebook Science (part of Open Science trend) promotes putting lab notebook together with all raw and processed data publicly available online.

- It is done on a regular basis, as data is generated.

- Numerous benefits from keeping the lab notebook in org-mode.
Labbook.org - Organization

---

**README:**
**General:**
**Experiments workflow:**
**Analysis workflow:**
* Template for data entry:

```
** data#
*** git:
#begin_src sh
git log -1
#end_src
*** Notes:
* Organization of git
** remote/origin/master branch:
** remote/origin/data# branches:
** remote/origin/data branch:
* Git TAGs:
** Stable versions:
*** stable1.0:
*** stable1.1:
*** stable2.0:
** papi_fix:
* Organization of code
** scripts:
** src/
** analysis/
** Parameters.txt:
* Additional feature:
** pagemap developed by Brice and Vincent:
** DONE arm source branch.= No need for now, added environment variable that solves this problem
** DONE Likwid or some other feature specific branch
** DONE PAPI counters branch.= No need for now, added environment variable that solves this problem
```
Using Emacs and Org-mode for Managing Experiments

Labbook.org - Data

* Memory allocation issue on ARM:...
* Striding phenomenon:...
* Data:
** data1:
** data2:
** dataSnow1:
** dataSnow2:
** dataSnow3:
*** git:
#+begin_src sh
git log -1
#+end_src

#+RESULTS:
| commit | df99b55c65f7867e9f91483474900805f459abd7 | Stanisic | <luka.stanisic@imag.fr> |        |        |        |
| Author: | Luka |            |        |        |        |
| Date: | Mon | Mar | 25 | 15:02:17 | 2013 | +0000 |
| Adding | data |        |        |        |        |

*** Notes:
**** Analysis from 1KB to 50KB:...
**** Analysis from 1KB to 28KB:
- Analysis showing how drop of performance is not occurring that often if we just go until 28KB:
  CacheData3.dat CacheData4.dat CacheData5.dat
  Pdf file is:
  analysis_pdf2
- Bad cases still exist (e.g. data9) but not that often
- It is still not clear whether we were just getting very lucky or OS is paying more attention on which physical p
  buffer is smaller than L1 cache?!

** dataKarma1
** dataSnow4:

--- LabBook.org 29% L259 Git-data (Org vl Wrap)
Pros and Cons

- Well structured and clean, easy to read and browse
- Everything is in one single file
- Can add executable code
- TAGS for better description of different parts
- TODO lists, priorities and checkboxes for future work
- Git-link for accessing directly to the desired version of the file
- Possible to directly export
Pros and Cons

- Well structured and clean, easy to read and browse
- Everything is in one single file
- Can add executable code
- TAGS for better description of different parts
- TODO lists, priorities and checkboxes for future work
- Git-link for accessing directly to the desired version of the file
- Possible to directly export

- Works only with Emacs, other editors will see it as a plain text
- Git see it as a plain text, which may cause problems with merging
Outline

1 Reproducible Research Tools
   - CDE
   - Sumatra
   - VisTrails
   - Collage
   - VCR

2 Literate programming
   - Sweave/Knitr
   - IPython
   - Org-babel

3 Using Git for Reproducible Research

4 Using Emacs and Org-mode for Managing Experiments

5 Conclusion
Conclusion and Future Work

- Although our approach is nontrivial for new users of Git, Emacs and org-mode, it provides numerous benefits.
- Takes time to set it up correctly, but saves time on long-term.
- Good trade-off between constraints and possibilities.

Prospective:
- Improve merging of Labbook.org
- For bigger projects consider adding deadlines, agenda, issue tracker
- Try this approach on different use cases.