

group meeting

march 2013

Burkhard Ritter. March 8, 2013.

Contents

1. Git in 13½ minutes
 1. Intro to Version Control Systems
 2. Hands-on Git
 3. Hands-on Bitbucket
2. Everything not physics in computational physics
 1. Common workflow
 2. Provenance
 3. Data management
 4. Thoughts on a Python-centric workflow

Before we start

- Computational physics: develop software
- Not professionals in software development
- Spend little time thinking about software development
- Collaboration on code level is rare
- Established best practices and proven concepts: rarely adapted
- Examples
 - Revision control
 - Unit tests and test-driven development

Version Control Systems

- Every notable software project uses VCS
- Manage evolving code or documents (e.g. Latex)
- Indispensable when working together in a team
- Keep track of changes: "revisions"

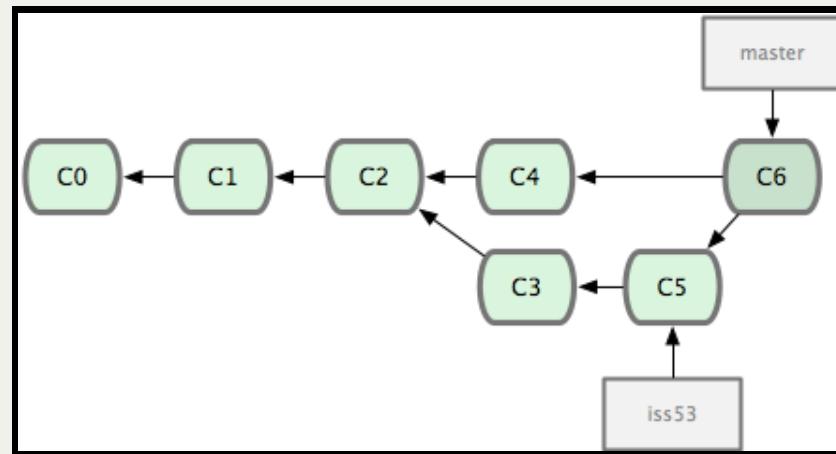


Image from: Pro Git book, <http://www.git-scm.com/book>

Version Control Systems

- Examples of VCS
 - Google Docs
 - Subversion
 - Distributed: Git, Mercurial
- Git
 - Originally developed for the Linux kernel
 - Made hugely popular by Github (<https://github.com/>)
- Version control can be complex and complicated
- Common use cases: dead simple, no reason not to use it

Hands-on: Git 1

```
$ mkdir my_cool_project
$ cd my_cool_project/
$ git init
Initialized empty Git repository in /Users/burkhard/my_cool_project/.git/
```

Hands-on: Git 2

```
$ vim funky_program.cpp
$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add ..." to include in what will be committed)
#
#       funky_program.cpp
nothing added to commit but untracked files present (use "git add" to track)

$ git add funky_program.cpp
$ git commit -a -m "Initial commit."
[master (root-commit) 1588d78] Initial commit.
 1 file changed, 4 insertions(+)
 create mode 100644 funky_program.cpp
```

Hands-on: Git 3

```
$ vim funky_program.cpp
$ git diff
diff --git a/funky_program.cpp b/funky_program.cpp
index c557e87..32e958e 100644
--- a/funky_program.cpp
+++ b/funky_program.cpp
@@ -1,4 +1,5 @@
int main ()
{
+    int whizbiz = 0;
    return 0;
}
$ git commit -a -m "Implemented the new whizbiz feature."
[master 8859ecf] Implemented the new whizbiz feature.
 1 file changed, 1 insertion(+)
```

Hands-on: Git 4

```
$ git log
commit 8859ecf6b0cbcd29407ddbfd3bc0f3ae5c953b2
Author: Burkhard Ritter
Date:   Thu Mar 7 18:36:30 2013 -0700

    Implemented the new whizbiz feature.

commit 1588d78bb6ee615499441a76ab3a8fb6a62241c5
Author: Burkhard Ritter
Date:   Thu Mar 7 18:34:29 2013 -0700

    Initial commit.
```

Hands-on: Git 5

```
$ git help  
[...verbose help message...]  
$ git help tag  
[...helpful manpage...]  
$ git tag -a -m "Version 1." v1  
$ git tag  
v1  
$ git describe  
v1  
  
$ vim funky_program.cpp  
$ git commit -a -m "More exciting features."  
[master 7684818] More exciting features.  
 1 file changed, 1 insertion(+)  
$ git describe  
v1-1-g7684818
```

Hands-on: Git 6

```
$ git help diff
$ git diff head^
diff --git a/funky_program.cpp b/funky_program.cpp
index 32e958e..0403944 100644
--- a/funky_program.cpp
+++ b/funky_program.cpp
@@ -1,5 +1,6 @@
 int main ()
 {
+    bool evil_bug = true;
        int whizbiz = 0;
        return 0;
 }
```

Hands-on: Git 7

- Immediate advantages
 - Simple
 - Keep track of changes
 - No different copies of code floating around in directories
 - Which code produces which result
 - Keep track of progress
 - Track down bugs

Collaboration

- Distributed VCS: everybody has full copy of repository
- Different collaboration workflows possible
- More details: **Pro Git book**
- Simple example: Hagen and Kriemhild work together on an epic poem, the Nibelungenlied (in Latex)

...

```
hagen@worms:~/nibelungenlied$ git pull kriemhild
hagen@worms:~/nibelungenlied$ git add chapter4.tex
hagen@worms:~/nibelungenlied$ edit chapter3.tex
hagen@worms:~/nibelungenlied$ git commit -a
hagen@worms:~/nibelungenlied$ git push kriemhild
```

...

```
kriemhild@worms:~/nibelungenlied$ edit chapter3.tex
kriemhild@worms:~/nibelungenlied$ git commit -a
kriemhild@worms:~/nibelungenlied$ git pull hagen
kriemhild@worms:~/nibelungenlied$ git pull ssh://kriemhild@worms/home/hage
n/nibelungen
```

Collaboration

More complex example: central server / repository

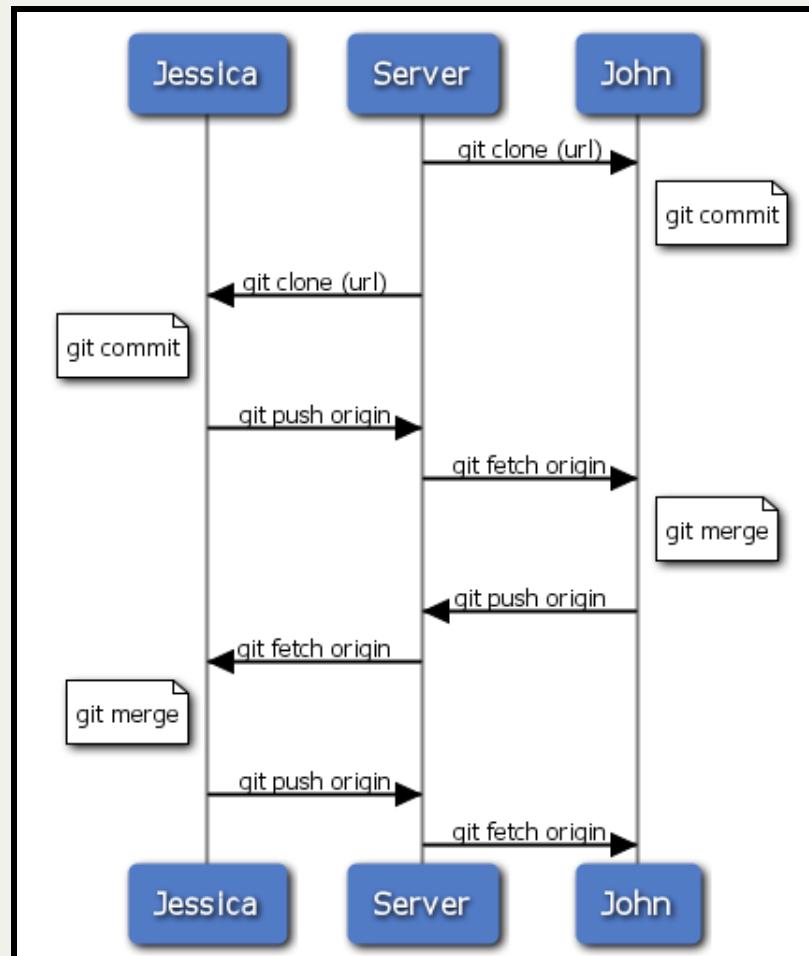


Image from: **Pro Git book**

Bitbucket

- Github
 - Extremely popular
 - Social coding
 - "Revolutionized open source" (**Wired article**)
 - Focus on individuals, not projects
 - Trend: publish everything, everything is public
(!= free/libre software)
- Bitbucket
 - Similar, but not hip
 - Unlimited private repositories, 5 collaborators
 - Unlimited academic plan!
- Could or should our code be public?

Bitbucket

- Why use Bitbucket?
 - Even easier to use Git
 - Web GUI
 - Backup
 - Sync between computers
 - Collaborate
 - Direct write access
 - Fork, create pull request
 - Teams

Hands-on: Bitbucket 1

The screenshot shows a web browser window with the Bitbucket interface. The URL in the address bar is https://bitbucket.org/meznom/gpsa_constitution. The repository name is **gpsa_constitution**, owned by **meznom**.

Repository Overview:

- Overview:** GPSA Constitution
- The Constitution of the [Graduate Physics Student Association](#) at the [University of Alberta](#). The Constitution is now maintained as a Latex document.
- March 2013, Burkhard Ritter burkhard@ualberta.ca

Recent activity:

- meznom pushed 1 commit to [meznom/gpsa_constitution](#) 23 hours ago
050c791 - Updated Meetings and Elections section.
- meznom pushed 1 commit to [meznom/gpsa_constitution](#) yesterday
e301a3a - Some smaller fixes. Two new sections.
- meznom pushed 1 commit to [meznom/gpsa_constitution](#) yesterday
27a3183 - Updated Council positions.
- meznom pushed 1 commit to [meznom/gpsa_constitution](#) 2 days ago

Repository Statistics:

SSH	git@bitbucket.org:meznom/gpsa_c
1	Branch
1	Tag
0	Forks
1	Follower

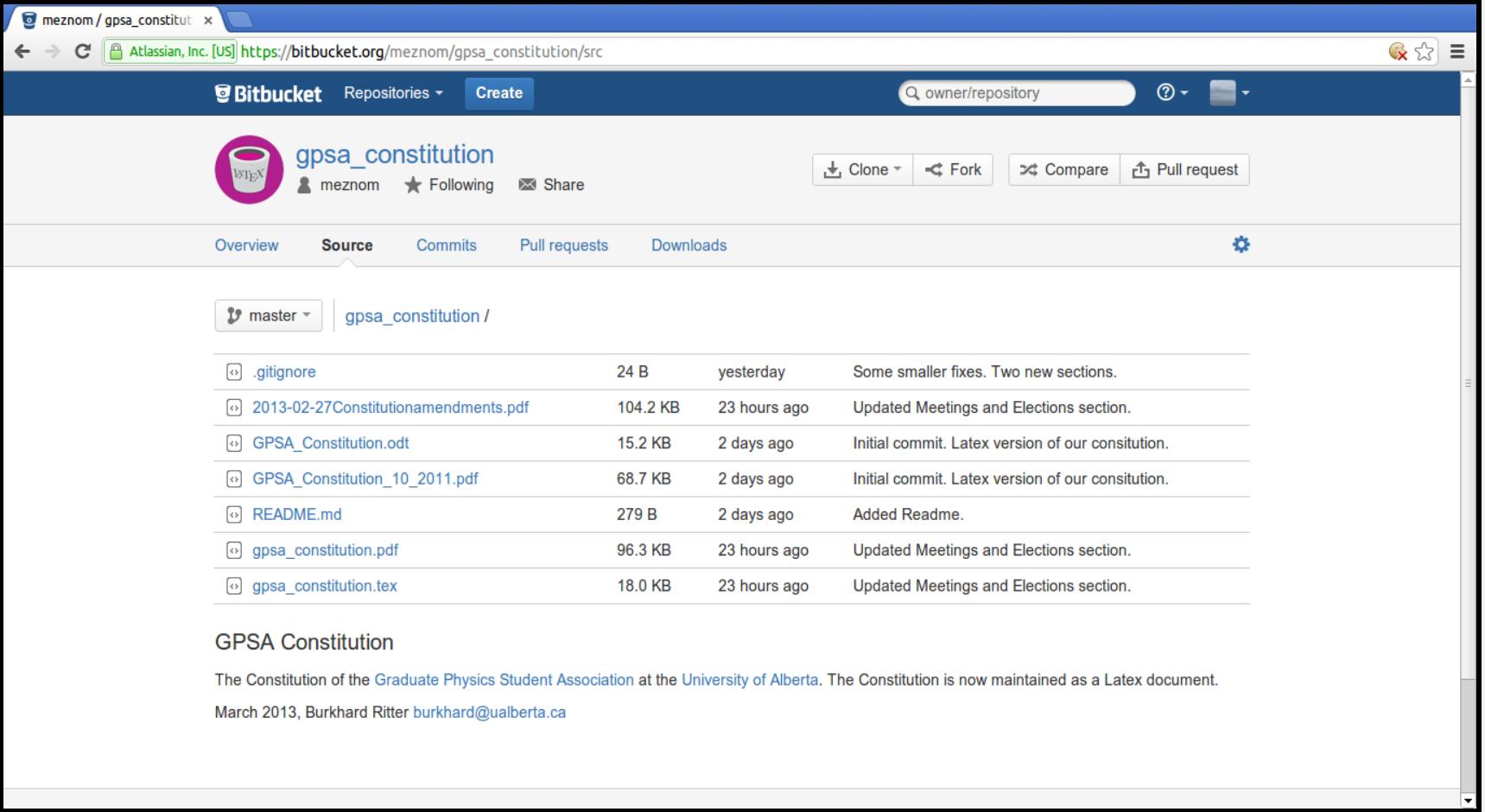
Repository Details:

- Owner: meznom
- Access level: Public
- Type: Git
- Language: LaTeX
- Last updated: 23 hours ago
- Created: 2013-03-06
- Size: 653.1 KB ([download](#))

Invite users to this repo:

[Send invitation](#)

Hands-on: Bitbucket 2



The screenshot shows a web browser window displaying a Bitbucket repository page. The URL in the address bar is https://bitbucket.org/meznom/gpsa_constitution/src. The repository name is **gpsa_constitution**, owned by **meznom**. The repository has 1 star and 0 forks. The **Source** tab is selected, showing the file structure and recent commits:

File	Size	Age	Description
.gitignore	24 B	yesterday	Some smaller fixes. Two new sections.
2013-02-27Constitutionamendments.pdf	104.2 KB	23 hours ago	Updated Meetings and Elections section.
GPSA_Constitution.odt	15.2 KB	2 days ago	Initial commit. Latex version of our constitution.
GPSA_Constitution_10_2011.pdf	68.7 KB	2 days ago	Initial commit. Latex version of our constitution.
README.md	279 B	2 days ago	Added Readme.
gpsa_constitution.pdf	96.3 KB	23 hours ago	Updated Meetings and Elections section.
gpsa_constitution.tex	18.0 KB	23 hours ago	Updated Meetings and Elections section.

GPSA Constitution
The Constitution of the [Graduate Physics Student Association](#) at the [University of Alberta](#). The Constitution is now maintained as a Latex document.
March 2013, Burkhard Ritter burkhard@ualberta.ca

Hands-on: Bitbucket 3

The screenshot shows the Bitbucket interface for creating a new repository. The URL in the address bar is <https://bitbucket.org/repo/create>. The main form is titled "Create a new repository" and contains the following fields:

- Name*: my_cool_project
- Description: Presenting my amazing coding chops.
- Access level: This is a private repository
- Repository type: Git Mercurial
- Project management: Issue tracking Wiki
- Language: C++ (dropdown menu)

Below the form, there are two informational boxes:

- New to Bitbucket?**: Learn the basics of using Git and Mercurial by exploring the Bitbucket 101.
- Working in a team?**: Create a team account to consolidate your repos and organize your team's work.

At the bottom of the page, there are links for navigation and footer information:

Blog · Report a bug · Support · Documentation · API · Forum · Server status · Terms of service · Privacy policy
Git 1.7.10.3 · Mercurial 2.2.2 · Django 1.3.1 · Python 2.7.3 · 23859b528861 / 81bc6031f2d7 @ bitbucket16

Hands-on: Bitbucket 4

The screenshot shows a web browser window for Bitbucket. The URL in the address bar is https://bitbucket.org/meznom/my_cool_project. The page title is "my_cool_project". The main content area displays the "Import an existing repository" wizard. On the left, there is a sidebar with four options: "Get started", "Make changes and push", "Invite your friends", and "Get to work". The "Import an existing repository" section contains instructions: "You already have a Git repository on your computer. Let's publish it to Bitbucket." Below the instructions is a code block:

```
$ cd /path/to/my/repo  
$ git remote add origin ssh://git@bitbucket.org/meznom/my_cool_project.git  
$ git push -u origin --all # to push changes for the first time
```

At the bottom right of this section is a "Next" button. At the very bottom of the page, there is a footer with links: Blog, Report a bug, Support, Documentation, API, Forum, Server status, Terms of service, Privacy policy, Git 1.7.10.3, Mercurial 2.2.2, Django 1.3.1, Python 2.7.3, and a server identifier 23859b528861 / 81bc6031f2d7 @ bitbucket02.

Hands-on: Bitbucket 5

```
burkhard@macheath:~/my_cool_project$ git remote add origin ssh://git@bitbucket.org/meznom/my_cool_project.git

burkhard@macheath:~/my_cool_project$ git push -u origin --all
[...]

burkhard@macheath:~/my_cool_project$ git status
# On branch master
nothing to commit, working directory clean

burkhard@macheath:~/my_cool_project$ git pull
Already up-to-date.
```

Hands-on: Bitbucket 6

The screenshot shows a web browser window for Bitbucket. The URL in the address bar is https://bitbucket.org/meznom/my_cool_project. The page displays a repository named "my_cool_project" owned by "meznom". The repository has a green C++ icon and is marked as "Following". The main navigation tabs are Overview, Source, Commits, Pull requests, and Downloads. On the right side, there are buttons for Clone, Fork, Compare, and Pull request, along with an SSH cloning link: `git clone git@bitbucket.org:meznom/my_cool_project`. A tooltip for this link provides help with cloning. Below the repository details, there's a sidebar with links: Get started, Make changes and push, Invite your friends, and Get to work. The main content area is titled "Make changes and push" and explains that every project needs a README file. It includes a code block with a sample git commit message:

```
$ echo "# This is my README" >> README.md  
$ git add README.md  
$ git commit -m "First commit. Adding a README."  
$ git push -u origin master
```

A "Next" button is visible at the bottom right of this section. At the very bottom of the page, there are links to Blog, Report a bug, Support, Documentation, API, Forum, Server status, Terms of service, Privacy policy, and system information: Git 1.7.10.3, Mercurial 2.2.2, Django 1.3.1, Python 2.7.3, and a server ID.

Hands-on: Bitbucket 7

```
burkhard@lise:~$ git clone git@bitbucket.org:meznom/my_cool_project.git  
[...]  
burkhard@lise:~$ cd my_cool_project/  
burkhard@lise:~/my_cool_project$ vim README.md  
burkhard@lise:~/my_cool_project$ git add README.md  
burkhard@lise:~/my_cool_project$ git commit -a -m "Added Readme"  
[...]  
burkhard@lise:~/my_cool_project$ git push  
[...]
```

...

```
burkhard@macheath:~/my_cool_project$ git pull  
[...]  
From ssh://bitbucket.org/meznom/my_cool_project  
 7684818..a4b5daa master      -> origin/master  
Updating 7684818..a4b5daa  
[...]  
burkhard@macheath:~/my_cool_project$ git log -n 1  
commit a4b5daad03272a7598b1d2263f946d7ca34fdfe9  
Author: Burkhard Ritter  
Date:   Thu Mar 7 22:37:55 2013 -0700  
  
    Added Readme
```

Hands-on: Bitbucket 8

The screenshot shows a web browser window with the Bitbucket interface. The URL in the address bar is https://bitbucket.org/meznom/my_cool_project/src. The repository name is **my_cool_project**, owned by **meznom**. The repository has a C++ icon and is marked as **Following**. The **Source** tab is selected, showing the contents of the **master** branch. The files listed are **README.md** (67 B, 10 minutes ago, Added Readme) and **funky_program.cpp** (77 B, 4 hours ago, More exciting features.). Below the files, there is a section titled **My amazing project** with the sub-section **Informative instructions.** containing a bulleted list:

- Step 1
- Step 2

 At the bottom of the page, there are links to [Blog](#), [Report a bug](#), [Support](#), [Documentation](#), [API](#), [Forum](#), [Server status](#), [Terms of service](#), and [Privacy policy](#). The footer also includes system information: [Git 1.7.10.3](#), [Mercurial 2.2.2](#), [Django 1.3.1](#), [Python 2.7.3](#), and the commit hash [23859b528861 / 81bc6031f2d7 @ bitbucket20](#).

Hands-on: Bitbucket 9

The screenshot shows a Bitbucket commit page for a project named "my_cool_project". The commit hash is a4b5daad03272a7598b1d2263f946d7ca34fdf9. The commit was made by Burkhard Ritter 11 minutes ago. The commit message is "Added Readme". There are no comments yet. The commit has changed 1 file, README.md, with 6 additions and 0 deletions. The diff view shows the following content:

```
1 +# My amazing project
2 +
3 +Informative instructions.
4 +
5 +* Step 1
6 +* Step 2
```

At the bottom of the page, there are links to Blog, Report a bug, Support, Documentation, API, Forum, Server status, Terms of service, and Privacy policy. There is also a footer note: CH 1.7.10.3 - Mercurial 3.3.2 - Django 1.3.1 - Python 2.7.3 - 22850b5238c1/31b26021d27 © bitbucket.org

Everything not physics in computational physics

- In everyday work: activities, procedures, issues not related to physics
- Similar for all of us
- Problem and project independent
- Examples
 - Software management
 - Data management
 - Plotting
 - Publishing
- Similarities are a result of common workflow

Everything not physics in computational physics

- We do not often talk about these non-physics issues
- Goal
 - Exchange ideas
 - Establish best practices
 - Possibly share code
- This talk
 - Open discussion
 - No answers

Common workflow

```
/----> develop / change code  
|  
|   /-> input parameter set 1      input parameter set 2      ...  
|  
|       run                      run                  ...  
|  
|       output data 1            output data 2      ...  
|  
|       store raw data  
|  
|       postprocess output data  
|  
|       store processed data  
|  
|       \-- plotting  
\\----- results, final plots
```

Common workflow

- Common aspects
 - Fast evolving code
 - Results depend crucially on code version
 - Multiple input parameter sets (possibly: parallelize)
 - Input set -> one output data point
 - Postprocess output
 - Plot processed output
 - Store raw and processed data, results and plots
- Opportunities for best practices and collaboration

Provenance

- Verifiability and reproducibility are at the heart of science
- Anybody at any time in the future
 - Take any (published) result
 - Go back and be able to reproduce and verify it
- Requires
 - Meticulously document every single step that led to the result
 - Quite involved for computer simulations

Provenance

- Pushed by Alps (ETH Zürich, <http://alps.comp-physics.org/>)
- VisTrails
 - vistrails.org, Polytechnic Institute of New York University
 - "Scientific workflow and provenance system"
 - Evolve workflows, document workflows
 - Example
 - Graph in paper
 - VisTrails opens workflow
 - Reproduce (rerun simulations)
- Complete and complex provenance system (too complex?)
- Take inspiration and learn

Provenance

VisTrails Builder - itk_vtk.xml

Pipeline

Properties

Version Tag: EEG, and Power | Change

User: eranders
Date: 04 Apr 2007 15:12:47

Notes

The MRI is shown with an interactive clipping plane as well as the three principle axes. The orthogonal cuts are controlled by the sphere widget at the center of the interactive cutting plane.

EEG information (time series) is shown below with each channel's corresponding power spectrum.

VisTrails - Spreadsheet - Untitled

Views

A: Axial MRI slice showing a red clipping plane. **B**: 3D rendering of the brain with a red clipping plane.

2: Coronal MRI slice.

3: Lateral MRI slice.

4: Four time series plots labeled 1, 2, 3, and 4, showing power spectral density (PSD) in dB over time. Each plot has a "pan/zoom mode" button below it.

Sheet 1

Provenance

- For every result, keep
 - Code version, `git describe`
 - Input parameters
 - Scripts / programs (or their configuration) for processing raw data
 - Plotting scripts / instructions
- Central code repository
- Publish code (i.e. open source)

Data management

“

In the long run, your data matters more than your code. It's worth investing some effort to keep your data in good shape for years to come.

”

Konrad Hinsen, "Caring for Your Data," Computing in Science and Engineering, vol. 14, no. 6, pp. 70-74, Nov.-Dec., 2012; available on computer.org

Data management

- Motivation
 - Publications: data shown and discussed
 - Traditionally
 - Data management low priority (e.g. processing, storage)
 - Data formats undocumented
 - Format conversion error prone
 - Down the road: hard or impossible to interpret data

Data management

- Data model design
 - Equivalent to software design
 - Describe data model in abstract but plain language
 - Equivalent to pseudo code
 - Specification
 - Documentation
 - Guidelines
 - Avoid redundancy
 - Keep extensibility in mind
 - More details in article

Data management

- Distinguish data model and representations of data model
- Representations
 - In memory (i.e. in program code)
 - On disk (i.e. file formats)
- In memory
 - Code easier to understand
 - Encourages code modularity
 - Different representations in different languages
- On disk: file formats
 - Different formats for different requirements
 - Binary for performance, e.g. HDF5
 - Ascii for readability, e.g. XML, JSON
 - Same data model: convert between formats easily and without loss

Data management

Example: XML

```
<molecule name="water">
  <atoms>0 H1 H2</atoms>
  <bonds>
    <bond atoms="0 H1" order=1 />
    <bond atoms="0 H2" order=2 />
  </bonds>
</molecule>
```

Example: JSON

```
{
  "type": "molecule",
  "name": "water",
  "atoms": ["0", "H1", "H2"],
  "bonds": [{"order": 1, "atoms": ["0", "H1"]},
             {"order": 1, "atoms": ["0", "H2"]}]} 
```

Data management

- Can we establish guidelines and best practices?
- Document our data models and formats
- Central data repository?
- Same class of problems: similar data models
 - E.g. Monte Carlo
- Use standardized file formats
 - HDF5
 - JSON
- Possibly: share code for data handling, input, output

Data management

Examples / ideas for Monte Carlo: HDF 5

```
/experiment
    attributes: id, description
/measurement
    attributes: count
/0
    attributes: config
/observables
    /Magnetization
        /data
        /jack
        /bins
/run
    attributes: count
/0
    /observables
        /Magnetization
            /data
```

Data management

Examples / ideas for Monte Carlo: JSON

```
{  
  "info": {  
    "program": "SSEMonteCarlo",  
    "version": "unknown",  
    "state": "done",  
    "seedvalue": 42,  
    "run": {  
      "0": {  
        "startdate": "2013-03-08T09:41:43Z",  
        "enddate": "2013-03-08T09:41:43Z"  
      },  
      "1": {  
        "startdate": "2013-03-08T09:46:13Z",  
        "enddate": "2013-03-08T09:46:13Z"  
      }  
    },  
  },  
}
```

Data management

Examples / ideas for Monte Carlo: JSON (continued)

```
"type": "ssemontecarlo.montecarlo.MonteCarlo",
"params": {
    "type": "ssemontecarlo.montecarlo.Struct",
    "N": 10,
    "beta": 1,
    "h": 10,
    "J": {
        "type": "ssemontecarlo.montecarlo.NNAFHeisenbergInteraction",
        "J": 1
    }
},
"mcparams": {
    "type": "ssemontecarlo.montecarlo.Struct",
    "t_warmup": 100,
    ...
},
"observables": [...],
```

Data management

Examples / ideas for Monte Carlo: JSON (continued)

```
"data": {  
    "ExpansionOrder": {  
        "mean": 246.39998046875,  
        "error": 0.9593617279141327,  
        "binCount": 200,  
        "binSize": 1  
    },  
    "Magnetization": {  
        "mean": -0.060000009536743164,  
        "error": 0.18963782783471672,  
        "binCount": 200,  
        "binSize": 1  
    }  
}
```

A Python-centric workflow

- IPython is awesome (ipython.org)
- Browser-based notebooks
 - Similar to Mathematica
 - Might be good fit for some workflows
- Comprehensive library and tools for parallelization

A Python-centric workflow

- Idea: (versioned) Python scripts control complete workflow
 - Set input parameters
 - Run program (in parallel)
 - Postprocess data
 - Store raw and processed data (possibly in MongoDB)
 - Do plotting (matplotlib)
- Core program itself
 - Written in Python (SciPy; PyPy -- pypy.org)
 - Written in C++, compiled as Python module (e.g. with Boost)

The End