EPItome-xl
on
Harford

The perhaps most spectacular application of complex systems is the dynamics of the human brain. If the brain is regarded as a complex system of neural cells, then its dynamics can be modeled by the nonlinear dynamics of neural networks. The emergence of mental states (e.g., recognition, emotions, thoughts) is modeled by the dynamics of (macroscopic) order parameters of cerebral assemblies which are caused by nonlinear neurochemical interactions of neural cells in learning strategies. Cell assemblies with mental states are interpreted as attractors of phase transitions.

Complex Systems and Nonlinear Dynamics of Nature and Society
KLAUS MAINZER

Joseph D. Viviano
Department of Biology
York University
Toronto, ON, CA
joseph.d.viviano@gmail.com

Contents

1 Introduction

This server was built by a complete stranger to server-building, and I undoubtedly made some silly mistakes along the way. This document is my effort to crystallize these decisions for those who paid for it (henceforth *the money*) and whoever takes over responsibility from me (henceforth *the unlucky*). Hopefully this isn't my last job ever, and I still walk this earth. If that is true, feel free to contact me at any time joseph.d.viviano@gmail.com

2 Connecting

To facilitate easy access to the server from your personal computer, on or off campus, we have configured three standard access methods.

2.1 Samba: Access to Shared Files On Your Desktop

All non-MRI data is accessible and editable through the finder/explorer window right on your desktop through a file-sharing service called Samba, assuming you have permission to do so. Contact an administrator to change permissions.

This is a good system for accessing behavioural data stored on the server. Some of this data is protected, and may not be accessible unless you are a member of the cnnlabdata group. An administrator can add you to this group if you need access.

2.1.1 Mac OSX

First, click on your desktop. In your menu bar, select go, connect to server, and type smb://130.63.40.171, after which you need only to enter your username and password for the server.

2.1.2 Windows

First, click on your start menu, and open run. Type
130.63.40.171, hit enter, and enter your username / password for the server. You should now have access to the CNNLAB folders in your Windows Explorer program.

2.1.3 Administering Samba

Samba is annoying and maintains its own set of passwords and permissions. This is particularly annoying because you will need to find an elegant way for users to set this password up. I haven't.

First, you need to add an existing user to samba using `smbpasswd -a [username]`. Next we add `[username]` to the `/etc/samba/smbusers` file in the form `[username] = "[username]"`. I've configured samba to mount `/srv/CNNLAB` as a writable directory structure in `/etc/samba/smb.conf` under the `[CNNLAB]` section. More directories could be added in this way.

2.2 SSH

If you are comfortable with terminal-only access, you can use the `ssh` command on all Mac or Linux systems (normally, this is already installed). Windows users should install PuTTY to gain access to the `ssh` command in the Windows terminal.

The major advantage of terminal-only access is lag-free operation on very slow network connections, and the ability to log in to the server from any computer with zero configuration. To access the server, enter the command `ssh -p 45100 [your username]@130.63.40.171`, where `[your username]` is the login name registered on Harford. If you require a username, please contact one of the administrators. You will next need to enter your case-sensitive password associated with your user account and should have access to the server.

2.3 x2go

Normally, you will want to view a full graphic-user interface while you interact with the server. This will allow you to easily view the data remotely, and make use of various programs not available with terminal-only access. To do this, we use x2go (download x2go client here). On Windows and Mac, you should be able to open x2go by clicking on the installed icon. On Linux, you may need to type `x2goclient` into the terminal.

Configuring x2go is fairly straightforward. First, click the 'new session' button, and set these values:

- Session name: [whatever you want ... I chose Harford]
- Host: 130.63.40.171

- Login: [your username]
- SSH port: 45100
- Session type: KDE (or XFCE on Windows or Linux only!)

You might notice that this is almost identical to the information that we use to `ssh` into the server, because that is what we are actually doing! The only additional setting regards the `session type`, as we now need to specify the graphic user interface.

A quick note on KDE & XFCE. If you are on a Macintosh computing device, and you try to use XFCE, your keyboard will not work properly. Therefore, you must select KDE.

The rest of the items can remain blank. You do not need to configure anything specifically under the connection tab. However, here you can set performance values to your liking, especially to compensate for a slow connection. Try moving the slider to the left before changing the picture quality settings manually.

Finally, you can customize session settings such as screen resolution (which will greatly affect your speed), sound, and other misc. settings. These values can be safely left at their defaults. Now you can click 'OK' and log into the server by clicking on the newly created button on the right hand side. You will need to enter your password for the server here.

2.3.1 Troubleshooting x2go

In some cases, you will receive an error involving the RSA key, and will not be allowed to log into the server. Your RSA keys are stored in `~/.ssh/known_hosts`. You can inspect the contents of this file by typing `cat ~/.ssh/known_hosts` into the terminal. If you find an entry that starts with `130.63.40.171`, remove it using either: `gedit ~/.ssh/known_hosts` or `vi ~/.ssh/known_hosts` (here are some instructions for vi). If you are having trouble doing this, you can simply reset the `known_hosts` file by removing it with `rm ~/.ssh/known_hosts` (this might affect other systems relying on RSA keys, but I will assume it won't if you are following these instructions). You should now be able to log into the server using x2go.

2.4 Passwords

Your default password can be changed to something more easily remembered by doing the following:

1. SSH into the server using port 45100.
    - Harford uses a non-standard SSH port to prevent random attacks. This requires you to specify it at the command line with the following command: `ssh -p 45100 [your username]130.63.40.171`.
2. Use the `passwd` command.
    - If you type passwd at the command line, you will be prompted for your old annoying password, and asked to enter a new one.Your new password must satisfy the following conditions:
        - 1 Uppercase letter
        - 1 lowercase letter
        - 1 number
        - 1 symbol (i.e., *, _#$%)
        - 12 characters length minimum.
    - This does not need to be hard to remember! Computer no think like human.
        - `#sQ_` is easy for a computer to guess.
        - `Br4in-fun-4eva!` is much harder.

Be sure to write your password down. If you forget your password, an administrator can reset it for you. If you *are* an administrator, you should know you can change `username`'s password by dropping to `root` and typing `passwd [username]`.

Also, recall that samba and unix passwords are *not* automatically synchronized. [This really should be fixed.]

3 Hardware

Harford is a `Thinkserver TS440 (8×3.5") HDD Hot-Swappable` server loaded with 4 HHD at the moment. Hard drive caddy part `# 03X3969, FRU HS 3.5'' HDD Tray V3.0`. It is loaded with 4 × Western Digital RED 4.0 TB 5400 HDD, configured in RAID 1+0. This currently leaves us with 8 TB of usable hard drive

space, striped arrays, and on-site redundancy. It can easily be expanded to 2× that with an additional 4 drives.

3.1 Hardware List

Currently, Harford contains:

1. Thinkserver TS440 [70AQ000CUX].
2. 20 GB RAM [2×8 GB added, 2×2 GB stock].

4 Server Architecture

4.1 Partitions

At the moment, Harford consists of three partitions:

1. `/boot`: a 200 MB partition that is integral to the life of the server. This contains the *Linux kernels* of the system – the server's brainstem. Do not mess with this partition.
2. `/`: a 75 GB partition containing all user `/home` folders, software, and the operating system. This *should* be large enough for indefinite expansion, if people don't store data in their home folders.
3. `/srv`: a 6.59 TB partition containing all of our data.

NB: this means that the lion's share of the disk space is found under `/srv` and therefore large files should *always* be kept there!

4.2 Operating System

I chose Ubuntu Server 12.04 LTS for its excellent support, modern features, and compatibility with the NeuroDebian project (`neuro.debian.net`). I named the server Harford after the Kubrickian hero. When I first installed the server, I was left only with a basic terminal. To get things normal-looking, I had to `sudo apt-get install` the following:

- openssh-server
- xfce4

- kde-plasma-desktop
- synaptic
- lightdm-gtk-greeter
- jockey-gtk
- dmz-cursor-theme
- xubuntu-icon-theme
- elementary-icon-theme

This will produce a basic desktop environment for you to work in. I include both the `xfce4` and `kde` desktop environments, which each have their own merits and utility. They essentially control the graphic user interface of the server but do not differ substantially in actual usability.

A note on what these basic programs do. `openssh-server` enables SSH access through the terminal and is how people can control the server remotely. `xfce4` and `kde-plasma-desktop` are two alternative graphic user interfaces for the computer. `synaptic` allows one to probe the Internet for possible software installations and largely automates that process. Most of the software on the server can be installed and/or uninstalled using this simple program. `lightdm-gtk-greeter` presents the user with a login screen. The remaining installs are all graphic user interface niceties.

4.3 Directories

The Unix file system is hierarchical: the top of this tree is at `/`. As mentioned previously, the bulk of the data is mounted on a separate partition under `/srv/`. Some optional software that wasn't installed by using `apt-get` / `synaptic` resides in `/opt/`. Under`/home/` resides each user's personal folder containing their desktop, personal files, and personal settings (such as the `~/.bashrc` file). Everything else is pretty much standard.

Under `/srv/` reside the following major directories:

- CNNLAB: Contains shared documents and behavioural data.
- CODE: Contains the pipeline & analysis code.
- FTP: A web-accessible FTP for data-sharing across the globe. Currently not password protected.
- LOGS: Log files generated by `CRON` jobs.
- MOVER: Files that only exist to be sorted through after import. I hope this folder isn't permanent.

- MRI: All the MRI (and possibly MEG) data.

In MRI, four directories exist: ANALYSIS, QUARANTINE, RAW, and WORKING. Under each of these folders is a mirrored set of experimental name directories. Each experiment is given a short name (e.g., 'TRSE', 'BEBASD', 'COP'). ANALYSIS is a convenience folder for those who would like to use the server's resources to analyze their experimental data. QUARANTINE is a safe place for data one would like to remove from the WORKING directory, but not delete. Generally 'bad runs' should be moved here instead of being deleted. WORKING is where the NIFTI-converted data resides and is accessed by the pipeline. RAW contains exact copies of the raw DICOM data, however it was found.

Under each experiment directory, the following structure is found:

- Subject IDs. Experiment-wide files are placed here.
- Subject file types (e.g., REST, T1, TASK). Names are arbitrary but must be consistent across subjects.
- File type session folders. Any across-session data also ends up here.
- Run folders. Each run folder should contain exactly one NIFTI file. In some cases it is appropriate to place run-specific data here (e.g., physiological recordings). Intermediate stages of pre-processing are also stored here.

4.4 Permissions

The MRI data is separated into 4 main branches: RAW, WORKING, QUARANTINE, and ANALYSIS. These different tiers are more or less editable by various users of the server to strike a balance between data-security and usability. As a general rule, we also don't let unauthorized people look at data they aren't supposed to for both privacy and competitive reasons. These permissions are maintained by a root-owned nightly CRON job maintain_permissions.sh.

The grandvizier user should be used by the system administrators to preform various tasks without the need of dropping to root, which I am trying to discourage as it can be dangerous to spend too much time with so much power. If you can see a file owned by the grandvizier, that typically means it is being protected.

4.4.1 RAW

These are DICOM files. Mostly used for archival purposes. Shouldn't be edited.

```
owner = grandvizier, rwx
group = staff , r-x
else = --
```

4.4.2 WORKING

These are the files manipulated by the pipeline code. Generally, these files should only be accessed by the pipeline and not manually. Right now, experiment specific group-wise permissions allow for you to go in and delete everything *except* the input RAW data at the bottom of the WORKING tree. This allows you to 'reset' problem subjects or whole users using the `cleanup_X.sh` programs, or manually remove problem files.

```
owner = grandvizier, rwx
group = [experiment], rwx (except inputs which are r-x)
else = , --
```

4.4.3 ANALYSIS

These folders are where people *are* allowed to mess around. Generally, outputs from the pipeline can be copied into the ANALYSIS tree for manipulation. This is zero-risk as there is always an identical copy of the pre-processed files in the WORKING directory.

```
owner = grandvizier, rwx
group = [exrperiment], rwx
else = , --
```

4.4.4 Privacy Notes

The `grandvizier` user is special, and shared among *the money* and *the unlucky*. The `grandvizier` is also capable of destroying millions of tax-payer dollars in a one-line command, so it shouldn't be used by anyone unless required.

All pipeline code will be run by the individuals within an experiment group – and the code will work so long as the individual is permitted to interact with a given data set. This is safe because the pipeline code isn't editable by the users in the first place, so we can't do undue harm to our data by mistake.

4.5 Security

4.6 Backup

5 Software & Configuration

The following is a list of the software installed on the server, and if you are lucky, it is even up-to-date.

- NeuroDebian:
  - FSL 5.0.6
  - DICOM2NIFTI
  - AFNI
  - FreeSurfer
  - Caret
  - Python: NiBabel
- General Computing:
  - R
  - gedit
  - Terminator
  - LibreOffice
  - Evince
  - Firefox
  - Inkscape
  - Gimp
  - Dia
  - VLC
  - Samba
  - Oracle Sun-Grid Engine
- Python Packages:
  - NumPy
  - SciPy
  - Pandas
  - matplotlib
  - scikit-learn
  - scikit-image
  - NetworkX

  - PyMVPA
  - iPython

- Misc – Installed Manually in `/opt`:
  - Gephi
  - MATLAB Compiler Runtime
  - McRetro

5.1 Oracle Sun-Grid Engine

The server, and EPItome-xl, rely on Oracle's Sun-Grid engine to schedule jobs. At the moment this feature is merely cosmetic, but it will help immensely when multiple users are attemping to run their experiments through the pipeline at the same time (such as 4 days before SfN abstracts are due). The EPItome-xl pipeline actually handles most of the nitty-gritty here, which I will document later (i.e., never). In order to install and configure the scheduler, I followed these instructions.

5.2 MRI Tools

5.3 Programming Languages

6 The Pipeline

For the pipeline to work, your RAW data must be converted to the appropriate format in the WORKING directory. EPItome-xl should be installed and configured properly on your server (code & up-to-date installation instructions are available from GitHub).

What follows is a description of what each module in EPItome-xl does. These modules can be easily chained together manually, or by using the command-line interface included with the pipeline. New modules are simply bash scripts which call various programs, including FSL, Freesurfer, AFNI, and custom python programs. Therefore, functionality of EPItome is easily extended without perturbing the function of older modules.

The types of modules included can be roughly split into 4 categories: freesurfer, pre-processing, quality-control, and cleanup.

6.1 Freesurfer

Right now, the default freesurfer recon-all is run on every participant before further processing. This is to produce surface files that can be used for cortical smoothing / data visualization, and the automatic generation of tissue masks which can be used for the generation of nuisance regressors.

6.1.1 fsrecon.py

Usage: `fsrecon.py <data_directory> <experiment> <modality> <cores>`

> data_directory – full path to your MRI/WORKING directory.
> experiment – name of the experiment being analyzed.
> modality – image modality to import (normally T1).
> cores – number of cores to dedicate (one core per run).

This sends each subject's T1s through the Freesurfer pipeline. It uses multiple T1s per imaging session, but does not combine them between sessions. Data is output to the dedicated FREESURFER directory, and should be exported to the MRI analysis folders using `fsexport.py`.

6.1.2 fsexport.py

Usage: `fsexport.py <data_directory> <experiment>`

> data_directory – full path to your MRI/WORKING directory.
> experiment – name of the experiment being analyzed.

Imports processed T1s from Freesurfer to the experiment directory.

6.2 Pre-Processing

This contains the lion's share of the pipeline. Every run of EPItome begins with `init_EPI`, which contains a non-contentious set of pre-processing steps for EPI images. THe following stages can be chained together at will to preform de-noising, spatial transformations, projections to surface-space, and spatial smoothing.

6.2.1 init_EPI

Usage: `init_EPI <data_quality> <del_tr> <t_pattern>`

data_quality – 'low' for poor internal contrast, otherwise 'high'.
del_tr – number of TRs to remove from the beginning of the run.
t_pattern – slice-timing at acquisition (from AFNI's 3dTshift).

Works from the raw data in each RUN folder. It performs general pre-processing for all fMRI data:

- Orients data to RAI
- Deletes initial time points (optionally)
- Removes data outliers
- Slice time correction
- Deobliques & motion corrects data
- Creates session mean deskulled EPIs and whole-brain masks
- Scales each run to have mode = 1000, removing small values
- Calculates various statistics + time series

6.2.2 combine_volumes

Usage: `combine_volumes <func1_prefix> <func2_prefix>`

func1_prefix – functional data prefix (eg., smooth in func_smooth).
func2_prefix – functional data prefix (eg., smooth in func_smooth).

Combines two functional files via addition. Intended to combine the outputs of `surfsmooth` & `surf2vol` with `volsmooth`, but could be used to combine other things as well. The functional files should not have non-zeroed regions that overlap, or the output won't make much sense.

6.2.3 linreg_calc_AFNI

Usage: `linreg_calc_AFNI <cost> <reg_dof> <data_quality>`

cost – cost function minimized during registration.
reg_dof – 'big_move' or 'giant_move' (from align_EPI_anat.py).
data_quality – 'low' for poor internal contrast, otherwise 'high'.

Uses AFNI's align_EPI_anat.py to calculate linear registration between EPI <–> T1 <–> MNI152, and generate an EPI template registered to T1 & T1 registered to EPI (sessionwise). Specific options can be found in the command-line interface's help function.

6.2.4 linreg_calc_FSL

Usage: `linreg_calc_FSL <cost> <reg_dof> <data_quality>`

> cost – cost function minimized during registration (see FSL FLIRT).
> reg_dof – 6, 7, 9, or 12 degrees of freedom (see FSL FLIRT),
> data_quality – 'low' for poor internal contrast, otherwise 'high'.

Uses FSL's FLIRT to calculate linear registration between EPI <–> T1 <–> MNI152, and generate an EPI template registered to T1 & T1 registered to EPI (sessionwise). Specific options can be found in the command-line interface's help function.

6.2.5 linreg_EPI2MNI_AFNI

Usage: `linreg_EPI2MNI_AFNI <func_prefix> <voxel_dims>`

> func_prefix – functional data prefix (eg.,smooth in func_smooth).
> voxel_dims – target voxel dimensions (isotropic).

Prepares data for analysis in MNI standard space.

6.2.6 linreg_EPI2MNI_FSL

Usage: `linreg_EPI2MNI_FSL <func_prefix> <voxel_dims>`

> func_prefix – functional data prefix (eg., smooth in func_smooth).
> voxel_dims – target voxel dimensions (isotropic).

Prepares data for analysis in MNI standard space.

6.2.7 linreg_FS2EPI_AFNI

Usage: `linreg_FS2EPI_AFNI`
Brings Freesurfer atlases in register with single-subject EPIs.

6.2.8 linreg_FS2EPI_FSL

Usage: `linreg_FS2EPI_FSL`
Brings Freesurfer atlases in register with single-subject EPIs.

6.2.9 linreg_FS2MNI_FSL

Usage: `linreg_FS2MNI_FSL`
Brings Freesurfer atlases in register with MNI standard space.

6.2.10 gen_regressors

Usage: `gen_regressors <func_prefix>`

func_prefix – functional data prefix (eg.,smooth in func_smooth).

Creates a series of regressors from fMRI data and a freesurfer segmentation:

- white matter + eroded mask
- ventricles + eroded mask
- grey matter mask
- brain stem mask
- dialated whole-brain mask
- draining vessels mask
- local white matter regressors + 1 temporal lag
- ventricle regressors + 1 temporal lag
- draining vessel regressors + 1 temporal lag

6.2.11 gen_gcor

Usage: `gen_gcor <func_prefix>`

func_prefix – functional data prefix (eg.,smooth in func_smooth).

Calls an AFNI script to calculate the global correlation for each concatenated set of runs (across all sessions). Useful for resting state functional connectivity experiments.

6.2.12 filter

Usage: `filter <func_prefix> <detrend_order> <gs_regress_flag>`

> func_prefix – functional data prefix (eg.,smooth in func_smooth).
> detrend_order – polynomial order to detrend each voxel against.
> gs_regress_flag – if == on, regress global signal.
> ventricle_regress_flag – if == on, regress mean ventricle.
> draining_vessel_regress_flag – if == on, regress draining vessels.
> local_white_matter_regress_flag – if == on, regress local WM.

This computes detrended nuisance time series, fits each run with a, computed noise model, and subtracts the fit. Computes temporal SNR. This program always regresses the motion parameters & their first lags, as well as physiological noise regressors generated my McRetroTS if they are available. The rest are optional, and generally advisable save global mean regression.

6.2.13 TRdrop

Usage: `TRdrop <func_prefix> <head_size> <FD_thresh> <DV_thresh>`

> func_prefix – functional data prefix (eg.,smooth in func_smooth).
> head_size – head radius in mm (def. 50 mm).
> thresh_FD – censor TRs with Δ motion > x mm (def. 0.3 mm).
> thresh_DV – censor TRs with Δ GS change > x % (def. 0.3 %).

This removes motion-corrupted TRs from fMRI scans and outputs shortened versions for connectivity analysis (mostly).

6.2.14 surfsmooth

Usage: `surfsmooth <func_prefix> <FWHM>`

> func_prefix – functional data prefix (eg.,smooth in func_smooth).
> FWHM – full-width half-maximum of the gaussian kernel convolved with the surface data.

This spatially-smooths cortical data along the surface mesh, estimated by Freesurfer.

6.2.15 surf2vol

Usage: `surf2vol <func_prefix> <target_prefix>`

func_prefix – functional data prefix (eg.,smooth in func_smooth).
target_prefix – target data prefix (eg.,smooth in func_smooth).

This projects surface data back into a functional volume with the same properties as <target_prefix>.

6.2.16 vol2surf

Usage: `vol2surf <func_prefix>`

func_prefix – functional data prefix (eg.,smooth in func_smooth).

Projects functional data from volume space to a Freesurfer generated cortical mesh.

6.2.17 volsmooth

Usage: `volsmooth <func_prefix> <mask_prefix> <FWHM>`

func_prefix – functional data prefix (eg., smooth in func_smooth).
mask_prefix – mask data prefix (eg., EPI_mask in anat_EPI_mask).
func_prefix – functional data prefix (eg.,smooth in func_smooth).

Re-samples a mask containing one or more labels to the functional data and smooths within unique values. All zero values in the mask are zeroed out in the output. The output of this can be combined with the outputs of `surfsmooth` & `surf2vol` using `combine_volumes`.

6.3 Quality Control

7 Further Reading