A Method to Automate Processing Provenance in AFNI

Thomas J Ross[1] and Ziad S Saad[2]

[1]Neuroimaging Research Branch, National Institute on Drug Abuse-Intramural Research Program, Baltimore, MD, United States
[2]Scientific and Statistical Computing Core, National Institute on Mental Health-Intramural Research Program, Bethesda, MD, United States
Contact: tross@mail.nih.gov

National Institute on Drug Abuse
National Institute of Neurological Disorders and Stroke
National Institute of Mental Health

AIM

A lightweight approach to document and replicate all processing steps on a dataset

Motivation

- Considerable heterogeneity in data analysis pipelines
  - ◊ Carp (2012) showed almost every study uses a unique combination of analysis parameters/procedures
  - ◊ Due in part to improving tools and changes in “best practices”
- Much processing in neuroimaging is done through scripts (e.g. AFNI or FSL)
  - ◊ Asymmetric relationship between scripts and the files they create: Can determine from a script what files it creates, not vice versa
- File provenance (place or source of origin) is highly desirable for reproducible research

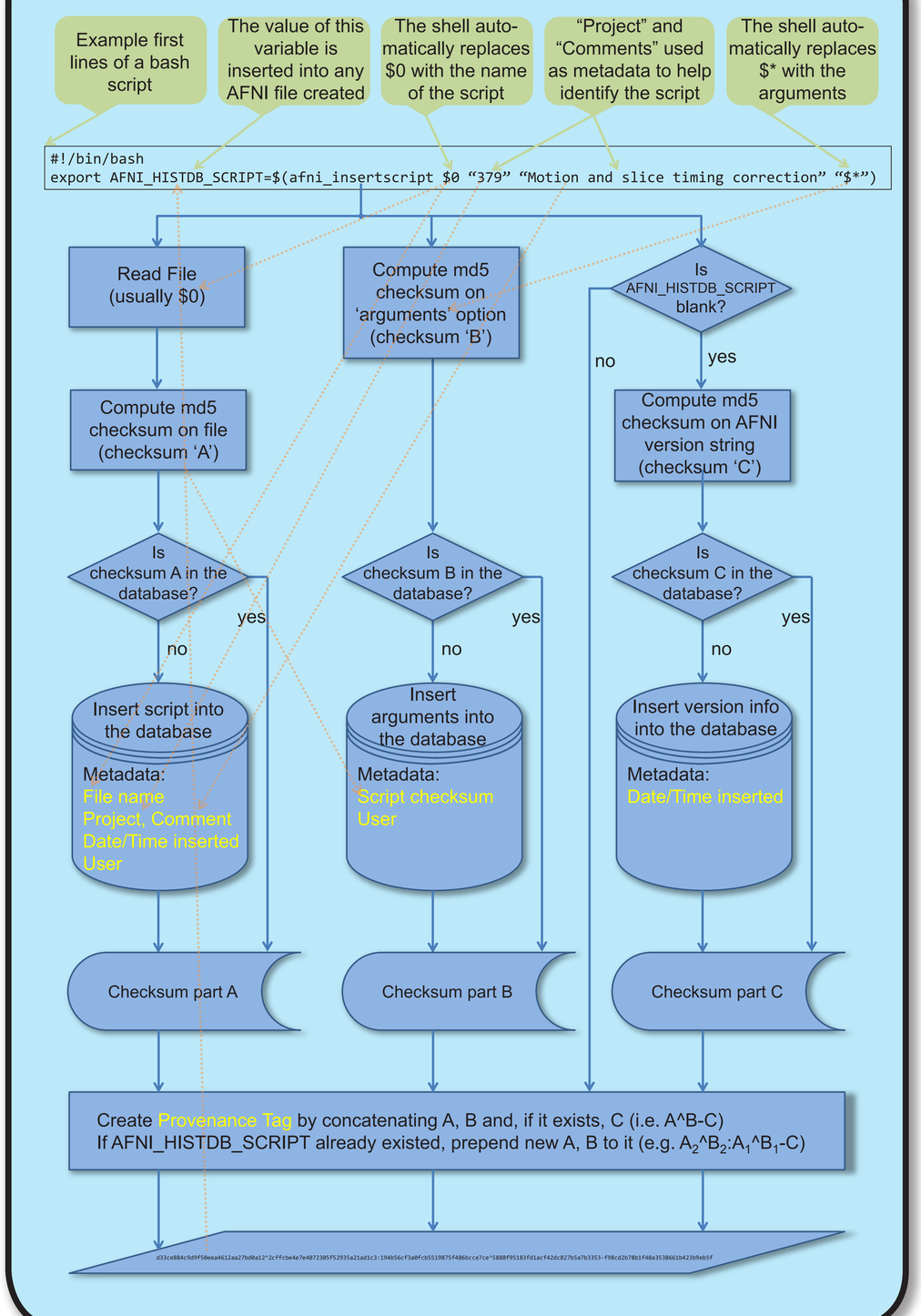
Prior Work

- The first provenance challenge (Moreau (2008)) specifically addressed the problem of an fMRI pipeline
  - ◊ 16 teams responded to the challenge, but it seems that none of the proffered solutions are in routine use in the neuroimaging community
    - » Many require using a specific ‘workflow’ that performs the appropriate documentation
      - * Often more difficult to create workflows than it is simple scripts
    - » Some require operating in a special environment that tracks all system calls
      - * Lacks simple annotation capabilities
      - * Querying the system quite challenging
- Arguably, the most popular neuroimaging provenance system is within the LONI pipeline (MacKenzie-Graham (2008))
  - ◊ Workflow based system with all of those advantages/disadvantages
- AFNI already has a history system that records the command line that created a file, and often the command lines of its children, etc.
  - ◊ Lacks the logic that went into the commands
  - ◊ Would be very difficult to reapply the same steps to new data (or the same data)

Approach

- Create provenance ‘tags’ of the form A^B-C.
  - ◊ All 3 are md5 checksums (a 128 bit cryptographic hash function)
  - ◊ A is the checksum of the script, B is the checksum of the command line arguments and C is the checksum of an AFNI version string
  - ◊ For example: 194b56cf3a0fcb5519875f486bcce7ce^5888f95183fd1acf42dc827b5a7b3353-f98cd2b78b1f48a3538661b423b9eb5f
- Provenance tags can be chained (e.g. A_2^B_2:A_1^B_1-C)
  - ◊ Happens automatically in subscripts (i.e. a script called from another script)
    - » Files created in the subscript can thus retrieve their own script, the calling script, its calling script, etc
    - » Only one C tag, as it is assumed that the AFNI version does not change during the running of the script
  - ◊ Can be used to manually associate a file(s) with an AFNI dataset(s) (e.g. associating the covariates file to the output of 3dttest++)
- Provenance tags automatically get inserted into AFNI datasets through the use of an environment variable, whose value is automatically saved as an ‘Attribute’ of any AFNI file
  - ◊ By using environment variables, this provenance method works with practically any programming language (e.g. bash, tcsh, python)
- The actual items associated with the provenance tags are inserted into an SQL database
  - ◊ Many advantages to using a database over, for example, storing the script itself within the AFNI dataset
    - » Only 1 copy of each unique item is stored
    - » Scalable from a single computer/user to a large site
    - » Acts as a backup of all versions of all scripts used in a project, without the need to learn user-unfriendly version control systems
    - » Searchable by metadata such as user, project, date
    - » Scripts are retrievable even in the absence of the neuroimaging datasets
  - ◊ Common queries (e.g. all scripts in a project) built into the retrieval program, with the full power of SQL available for more complex queries and sophisticated users
  - ◊ Name of the database stored in an environment variable; easily set by site, user or not at all
  - ◊ Currently supports sqlite, with plans to support mysql and postgresql (using the libzdb library)
- All functionality implemented in 3 new AFNI programs
  - ◊ afni_insertscript
    - » Creates the database, computes the checksums, performs database inserts, helps to set the environment variable
  - ◊ afni_getscript
    - » Lists scripts and afni version associated with a dataset, retrieves scripts, performs simple queries
  - ◊ afni_deletescript
    - » Removes scripts from the database

Program Logic/Data Flow



References/Acknowledgement

Carp (2012) Neuroimage 63:298
MacKenzie-Graham et al. (2008) Neuroimage 42:178
Moreau et al. (2008) Concurrency and Computation: Practice and Experience 20:409

This work was sponsored by the Intramural Research Programs of the National Institute on Drug Abuse, the National Institute of Mental Health and the National Institute of Neurological Disorders and Stroke, The National Institutes of Health.

A Complete Example

Contents of preprocess.sh (which calls the next script)

```
#!/bin/tcsh
#script needs 2 arguments: $1 = subject, $2 = run
setenv AFNI_HISTDB_SCRIPT `afni_insertscript $0 “demoproject” “perform all preprocessing steps” “$*”`
cd /demoproject/$1
to3d -epan -time:zt 39 75 0 alt+z -prefix ${1}-run$2 -save_outliers outliers.1D run${2}/*

#The following finds the minimum index of the outlier file
set base=`python -c “import numpy; print numpy.loadtxt(‘outliers.1D’)[:,0].argmin()”`

/demoproject/scripts/volreg.sh $1 $2 $base
```

The **ONE** line you need to add to your scripts

Contents of volreg.sh

```
#!/bin/bash
#script needs 3 arguments: $1 = subject, $2 = run, $3 = volreg base
export AFNI_HISTDB_SCRIPT=$(afni_insertscript $0 “demoproject” “volreg and slice timing” “$*”)

3dvolreg -base $3 -1Dfile ${1}-run${2}-motion.1D -tshift 0 \
 -prefix tempfile ${1}-run${2}+orig

#can even associate another file with the volreg output
export AFNI_HISTDB_SCRIPT=$(afni_insertscript ${1}-run${2}-motion.1D “demoproject” “motion file”)
3dcopy tempfile ${1}-run${2}-volreg
export AFNI_HISTDB_SCRIPT=$(afni_insertscript -pop)

rm *motion* tempfile*
```

Associate a text file with an AFNI file

Example commands

```
/demoproject/tom> ../scripts/preprocess.sh tom 1
/demoproject/tom> ls
run1  tom-run1+orig.BRIK  tom-run1+orig.HEAD  tom-run1-volreg+orig.BRIK  tom-run1-volreg+orig.HEAD
/demoproject/tom> afni_getscript -l tom-run1-volreg+orig
AFNI version used for this file:
Version AFNI_2011_12_21_1014
[[Precompiled binary linux_openmp_64: Jun 25 2013]]

Script number: 1
        checksum = 91455926c825a62a15a868633f171ebd
         program = tom-run1-motion.1D
         project = demoproject
            user = tross
GMT time entered = 2014-05-20 22:03:59
         comment = motion file

Script number: 2
        checksum = b1b1470d8d583d10ae320bd9a61e5154
         program = volreg.sh
         project = demoproject
            user = tross
GMT time entered = 2014-05-20 22:03:52
         comment = volreg and slice timing
            args = tom 1 37

Script number: 3
        checksum = e54b959a0a19a9ff0a99244a23093ef4
         program = preprocess.sh
         project = demoproject
            user = tross
GMT time entered = 2014-05-20 22:03:48
         comment = perform all preprocessing steps
            args = tom 1
/demoproject/tom> afni_getscript -s 2 tom-run1-volreg+orig > volreg2.sh
/demoproject/tom> diff volreg2.sh ../scripts/volreg.sh
/demoproject/tom>
```

Retrieve the AFNI version used to create the files

The scripts and file associated with the AFNI file

Can retrieve one of the scripts; it is identical to the original