

HD4630 Workshop II

First- & Second-Level Analysis

Elizabeth DuPre

Human Neuroscience Institute
Department of Human Development
Cornell University

Workshop I Recap

- ▶ Preprocessing
 - ▶ Discard pre-steady state TRs
 - ▶ Slice-timing correction
 - ▶ Rigid-body motion correction
 - ▶ Coregistration of functional & anatomical
 - ▶ Normalization to standard template
 - ▶ Spatial filtering (Smoothing)
- ▶ Quality Control
 - ▶ Visual inspection
 - ▶ Censoring or "scrubbing" motion

Plan for Today

- ▶ Discuss first- and second-level analyses in a traditional general linear model (GLM) framework
- ▶ Conduct a first-level, fixed-effects analysis in AFNI using `uber_subject.py`
- ▶ Conduct a second-level, random-effects analysis in AFNI using `uber_ttest.py`

First-level Analysis

- ▶ First-level analysis involves estimating the β -matrix in

$$Y = X\beta + \epsilon$$

by constructing the contrast matrix X

- ▶ A great review of the math underlying this is available from [Mumford Brain Stats](#)

First-Level Analysis, *cont.*

- ▶ We need to input the stimulus timing for each of the conditions in our task
 - ▶ We'll collapse the two Flanker 'congruent' and 'incongruent' conditions, ignoring participant accuracy



INCONGRUENT

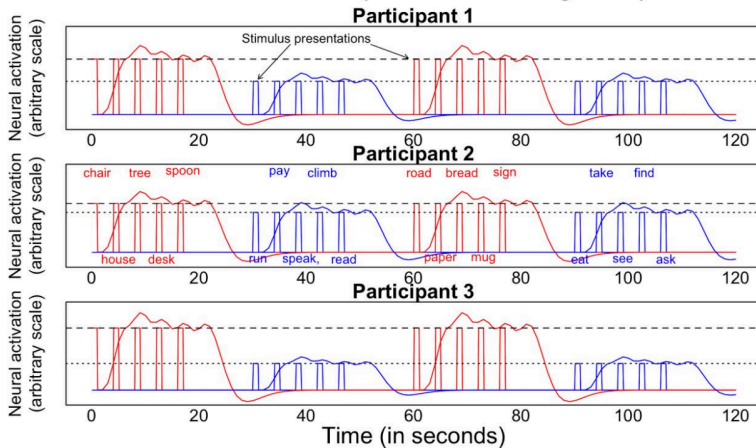


CONGRUENT



NEUTRAL

A: Standard model (stimulus effects ignored)



Westfall, Nichols, & Yarkoni

bioRxiv

Second-Level Analysis

- ▶ Second-level analysis as implemented in fMRI takes a summary-statistics approach

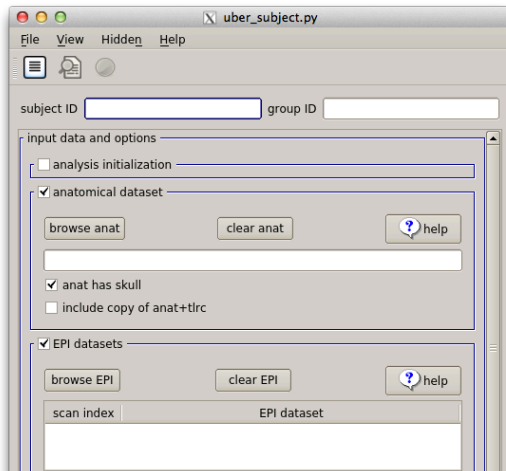
$$\hat{\beta} = X_g \beta_g + \eta^*$$

where beta-hats for each contrast, for each subject are carried forward from the first-level

- ▶ A great review of the math underlying this is available from [Mumford Brain Stats](#)

To Do: Set Preprocessing Options

- ▶ These were discussed in the last workshop
 - ▶ For a review, see last week's slides on the [course website](#)



Stimulus Timing Information

- ▶ All stimulus timing information is included in the `*events.tsv` files for each run
 - ▶ However, we need to convert this information into text files that AFNI will accept
- ▶ You'll be provided with a folder containing
 - ▶ A Python script to convert these files
 - ▶ The created text files themselves

To Do: Enter Stimulus Timing

- ▶ In `uber_subject.py`, select the provided text files in the section 'Stimulus Timing Files'
- ▶ You can also specify the basis function and the file 'type' (see `3dDeconvolve -help` for relevant options)

The screenshot shows a dialog box titled "stimulus timing files" with a checked checkbox. It contains several controls for configuring stimulus timing:

- Buttons: "browse stim", "clear stim", and a "help" button with a question mark icon.
- Table:

	index ▾	label	basis	type	stim (timing) file
1	1	Congruent	GAM	times	Flanker_01_Congruent.txt
2	2	Incongruent	GAM	times	Flanker_02_Incongruent.txt

Below the table, there are text input fields and dropdown menus:

- stim directory: `/home/student/data/Flanker_task/sub-01/func`
- wildcard form: `Flanker_0*ongruent.txt`
- stim file count: `2`
- init basis funcs: a dropdown menu set to "choose" and a text field containing "GAM".
- init file types: a dropdown menu set to "choose" and a text field containing "times".
- A checkbox labeled "use wildcard form" which is currently unchecked.

General Linear Tests (GLTs)

- ▶ AFNI refers to contrasts between conditions as General Linear Tests (GLTs)
 - ▶ This is because we're working in the General Linear Model (GLM)
- ▶ We'll need to specify what tests we would like to conduct
 - ▶ *Congruent > Incongruent*
 - ▶ *Incongruent > Congruent*

To Do: Specify GLTs

- ▶ When inputting stimulus timing files, AFNI will automatically associate each condition with your provided label
- ▶ You can use those labels to enter your desired GLTs
 - ▶ Select 'init with examples' to see example GLTs with your conditions

☒ symbolic GLTs

insert glt row init with examples

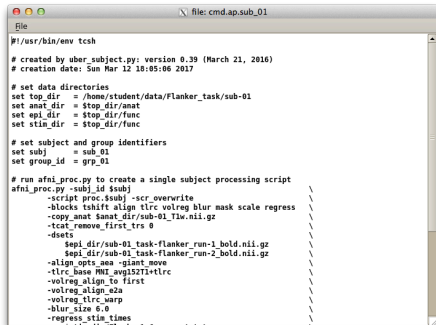
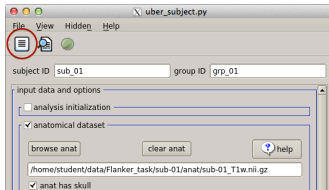
resize glt table clear glt table ? help

	label	symbolic GLT
1	C_I	+Congruent -Incongruent
2	I_C	-Congruent +Incongruent

Glt count 2

To Do: Review Processing Script

- ▶ Once you've entered all relevant parameters for preprocessing and first-level analysis, we can review and run the associated script
- ▶ Estimated run time is 40 minutes per subject



The Design Matrix

- ▶ All of the information we have entered thus far will form the X or 'design' matrix
- ▶ This is entered into the model

$$Y = X\beta + \epsilon$$

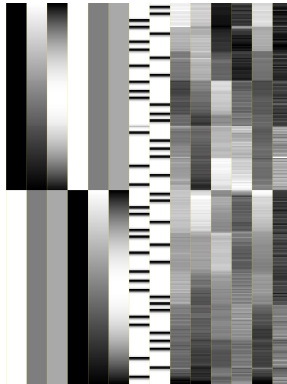
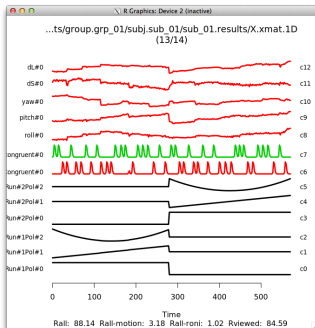
and will allow us to estimate the β -matrix for each participant

The Design Matrix, *cont.*

- ▶ After `uber_subject.py`, the design matrix is output as a `*.xmat.1D` file
- ▶ Multiple versions of this file exist, including
 - ▶ A list of input stimulus timings
(`X.stim.xmat.1D`)
 - ▶ A full design matrix with no motion censoring
(`X.nocensor.xmat.1D`)
 - ▶ A full design matrix with motion censoring
(`X.xmat.1D`)
- ▶ In this class, we'll be working with the `X.xmat.1D` file

To Do: Review the Design Matrix

- ▶ `ExamineXmat` reads in the generated `*.xmat.1D` file to visualize the time series of the design matrix
- ▶ `1dgrayplot` reads in the generated `*.xmat.1D` file to visualize a graph of the design matrix



Viewing First-Level results

- ▶ We need to carry forward the beta coefficients from the first- to second-level
 - ▶ We'll need their index in the created stats file to do that
- ▶ To get the correct index, we can view our first-level results in AFNI

Second-Level Analysis

- ▶ Once we're confident at the first-level, we can move forward to second-level analysis
- ▶ AFNI also provides a GUI to do this!
 - ▶ `uber_ttest.py`

