

Time Series Analysis in **AFNI!**

Outline: 6+ Hours of Edification

- Philosophy (e.g., theory without equations)
- Sample FMRI data
- Theory underlying FMRI analyses: the **HRF**
- “Simple” or “Fixed Shape” regression analysis
 - ★ Theory and Hands-on examples
- “Deconvolution” or “Variable Shape” analysis
 - ★ Theory and Hands-on examples
- Advanced Topics (followed by brain meltdown)

Goals: Conceptual Understanding + Prepare to Try It Yourself

Data Analysis Philosophy

- **Signal** = Measurable response to stimulus
- **Noise** = Components of measurement that interfere with detection of signal
- Statistical detection theory:
 - ★ **Understand** relationship between stimulus & signal
 - ★ Characterize noise statistically
 - ★ Can then devise methods to distinguish noise-only measurements from signal+noise measurements, and assess the methods’ reliability
 - ★ Methods and usefulness depend strongly on the assumptions
 - Some methods are more “robust” against erroneous assumptions than others, but may be less sensitive

FMRI Philosophy: Signals and Noise

- FMRI [Stimulus](#)→[Signal](#) connection and [noise statistics](#) are both complex and poorly characterized
- Result: there is no “**best**” way to analyze FMRI time series data: there are only “**reasonable**” analysis methods
- To deal with data, must make some assumptions about the signal and noise
- Assumptions will be wrong, but must do *something*
- Different kinds of experiments require different kinds of analyses
 - ★ Since signal models and questions you ask about the signal will vary
 - ★ It is important to [understand](#) what is going on, so you can select and evaluate “reasonable” analyses

Meta-method for creating analysis methods

- Write down a mathematical model connecting stimulus (or “activation”) to signal
- Write down a statistical model for the noise
- Combine them to produce an equation for measurements given signal+noise
 - ★ Equation will have unknown parameters, which are to be estimated from the data
 - ★ N.B.: signal may have zero strength (no “activation”)
- Use statistical detection theory to produce an algorithm for processing the measurements to assess signal presence and characteristics
 - ★ e.g., least squares fit of model parameters to data

Time Series Analysis on Voxel Data

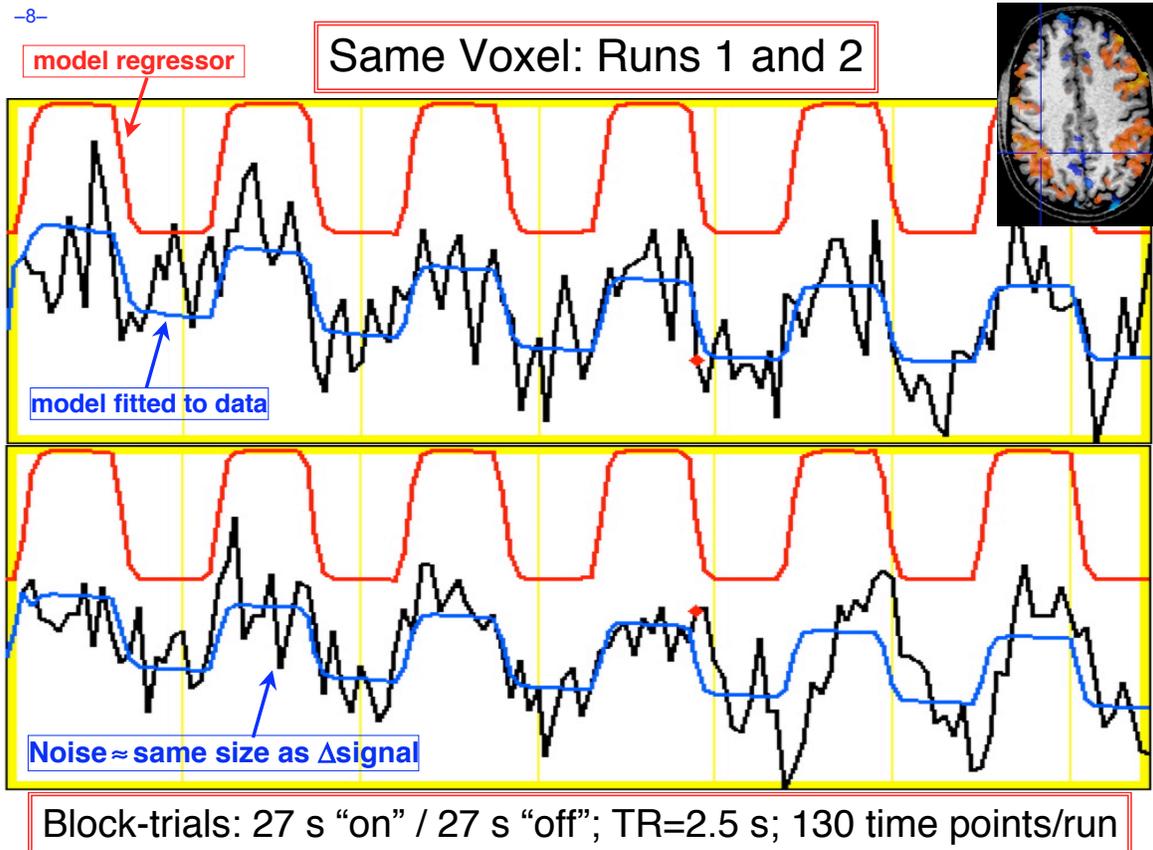
- Most common forms of fMRI analysis involve fitting an activation+BOLD model to each voxel's time series *separately* (AKA “univariate” analysis)
 - ★ Some pre-processing steps do include inter-voxel computations; e.g.,
 - spatial smoothing to reduce noise
 - spatial registration to correct for subject motion
- Result of model fits is a set of parameters at each voxel, estimated from that voxel's data
 - ★ e.g., activation amplitude (β), delay, shape
 - ★ “SPM” = statistical parametric map; e.g., t or F
- Further analysis steps operate on individual SPMs
 - ★ e.g., combining/contrasting data among subjects

Some Features of fMRI Voxel Time Series

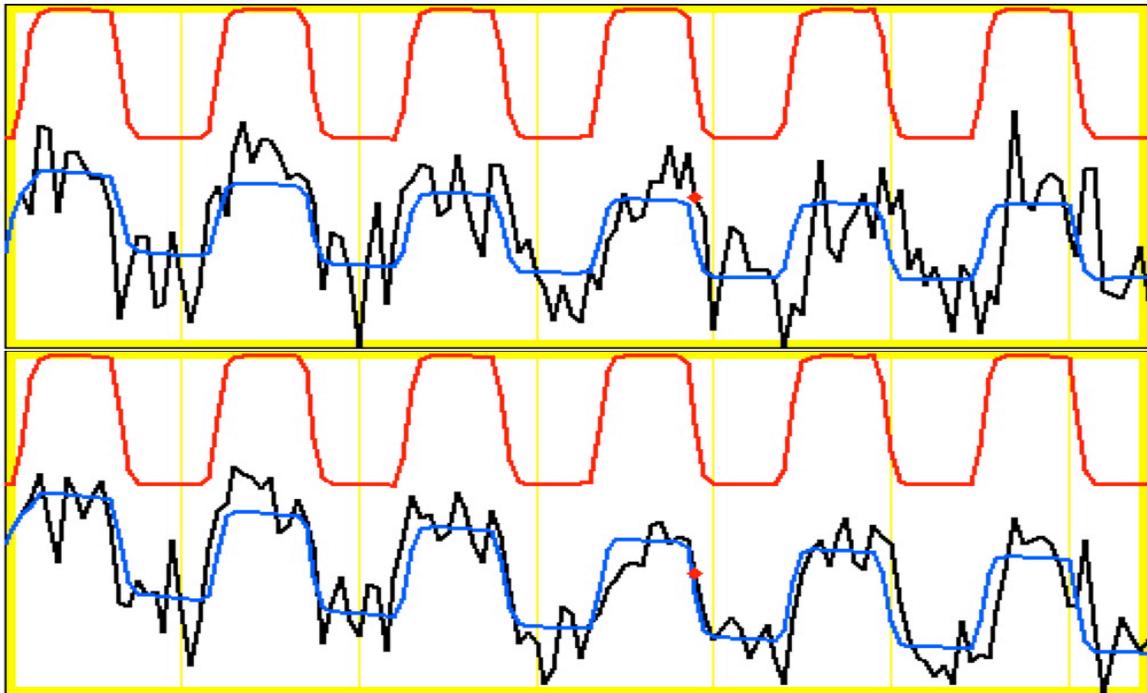
- fMRI only measures *changes* due to neural “activity”
 - ★ Baseline level of signal in a voxel means little or nothing about neural activity
 - ★ Also, baseline level tends to drift around slowly (100 s time scale or so; mostly from small subject motions)
- Therefore, an fMRI experiment must have at least 2 different neural conditions (“tasks” and/or “stimuli”)
 - ★ Then statistically test for differences in the MRI signal level between conditions
 - ★ Many experiments: one condition is “rest”
- Baseline is modeled separately from activation signals, and *baseline model includes “rest” periods*

Some Sample FMRI Data Time Series

- First sample: Block-trial FMRI data
 - ★ “Activation” occurs over a sustained period of time (say, 10 s or longer), usually from more than one stimulation event, in rapid succession
 - ★ BOLD (hemodynamic) response accumulates from multiple close-in-time neural activations and is large
 - ★ BOLD response is often visible in time series
 - ★ Noise magnitude about same as BOLD response
- Next 2 slides: same brain voxel in 3 (of 9) EPI runs
 - ★ **black curve** (noisy) = data
 - ★ **red curve** (above data) = ideal model response
 - ★ **blue curve** (within data) = model fitted to data
 - ★ somatosensory task (finger being rubbed)



Same Voxel: Run 3 and Average of all 9



⇒ Activation amplitude & shape vary among blocks! Why???

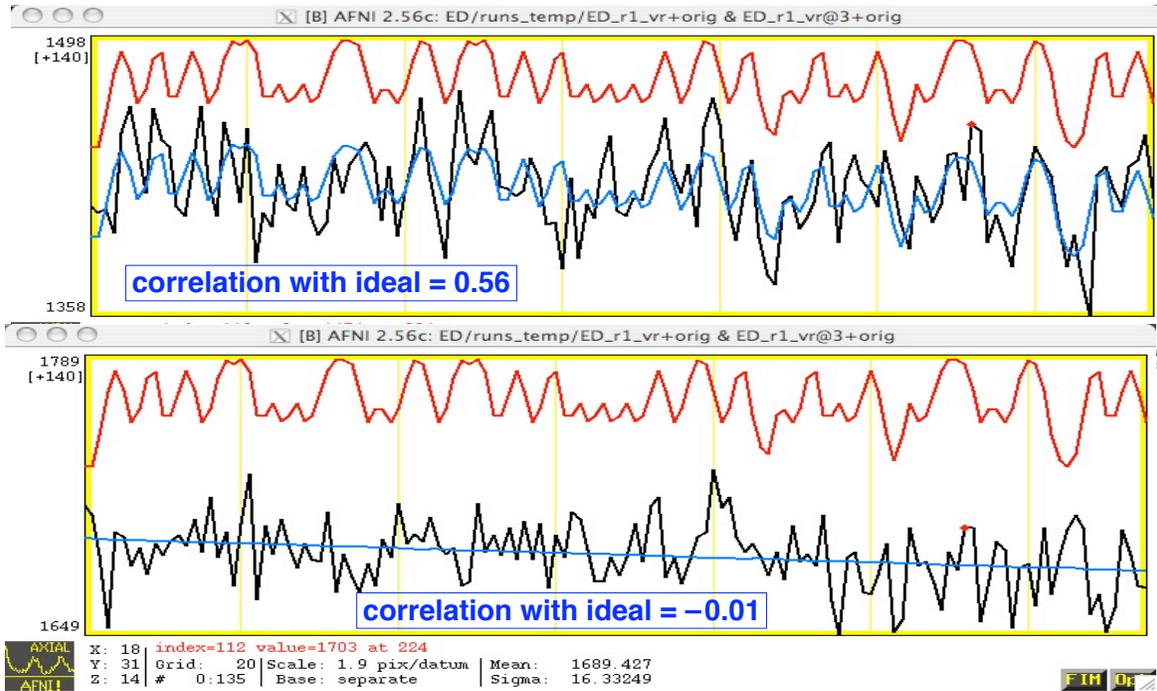
More Sample FMRI Data Time Series

- Second sample: Event-Related FMRI
 - ★ “Activation” occurs in single relatively brief intervals
 - ★ “Events” can be randomly or regularly spaced in time
 - If events are randomly spaced in time, signal model itself looks noise-like (to the pitiful human eye)
 - ★ BOLD response to stimulus tends to be weaker, since fewer nearby-in-time “activations” have overlapping signal changes (hemodynamic responses)
- Next slide: Visual stimulation experiment



“Active” voxel shown in next slide

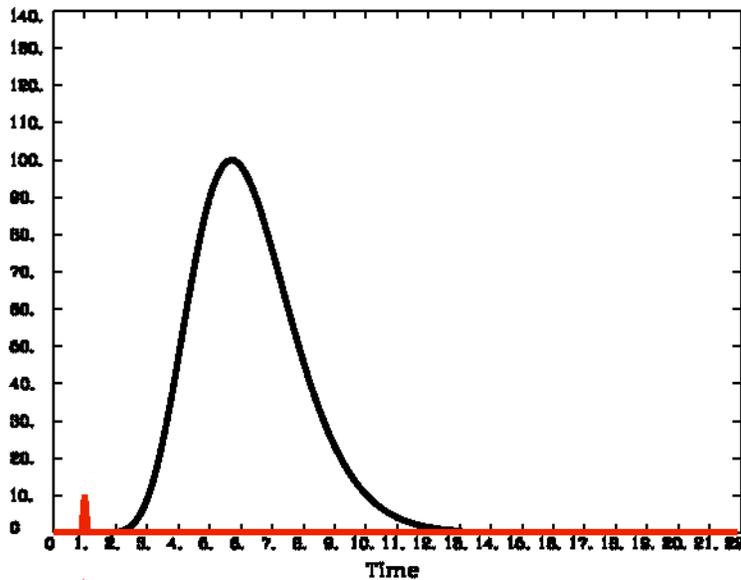
Two Voxel Time Series from Same Run



Lesson: ER-FMRI activation is not obvious via casual inspection

Hemodynamic Response Function (HRF)

- **HRF** is the idealization of measurable FMRI signal change responding to a single activation cycle (up and down) from a stimulus in a voxel



Response to brief activation (< 1 s):

- delay of 1-2 s
- rise time of 4-5 s
- fall time of 4-6 s
- model equation:

$$h(t) \propto t^b e^{-t/c}$$

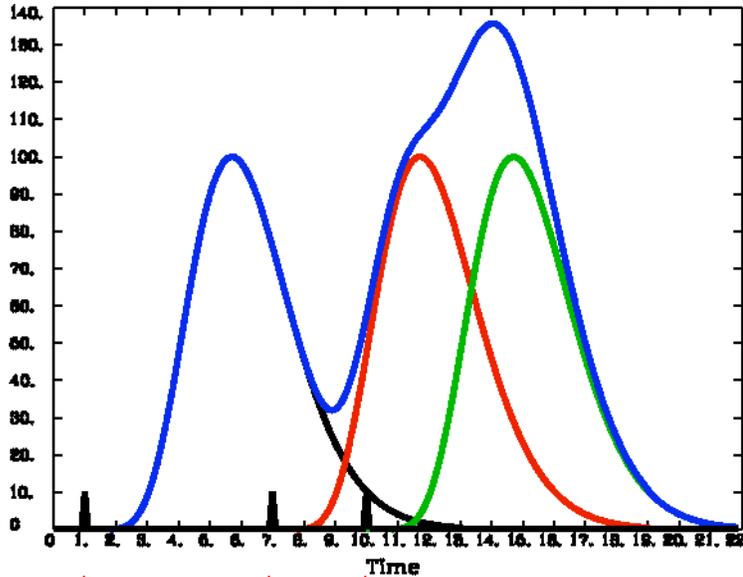
- $h(t)$ is signal change t seconds **after** activation

1 Brief Activation (Event)

Linearity of HRF

- Multiple activation cycles in a voxel, closer in time than duration of HRF:

★ Assume that overlapping responses add



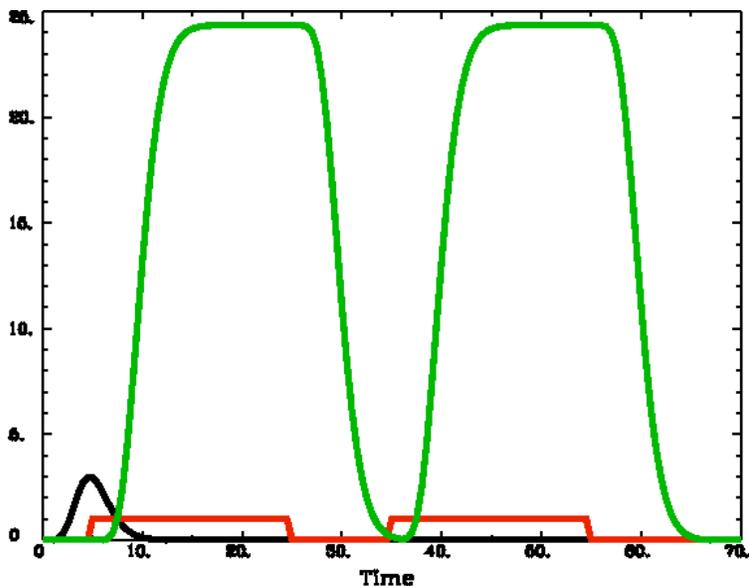
- Linearity is a pretty good assumption
- But not apparently perfect — about 90% correct
- Nevertheless, is widely taken to be true and is the basis for the “general linear model” (GLM) in fMRI analysis

3 Brief Activations

Linearity and Extended Activation

- Extended activation, as in a block-trial experiment:

★ HRF accumulates over its duration (≈ 10 s)

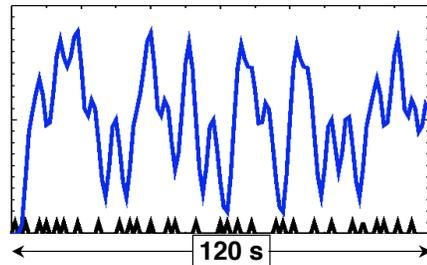
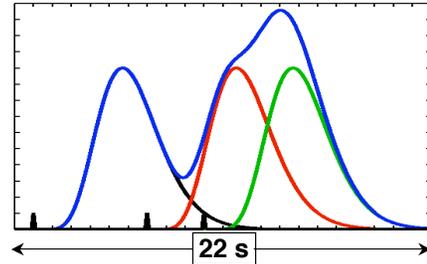


- **Black** curve = response to a single brief stimulus
- **Red** curve = activation intervals
- **Green** curve = summed up HRFs from activations
- Block-trials have larger BOLD signal changes than event-related experiments

2 Long Activations (Blocks)

Convolution Signal Model

- FMRI signal model (in each voxel) is taken as sum of the individual trial HRFs (assumed equal)
 - ★ Stimulus timing is assumed known (or measured)
 - ★ Resulting time series (in blue) are called the *convolution* of the HRF with stimulus timing
 - ★ Finding HRF=“deconvolution”
 - ★ AFNI code = `3dDeconvolve`
 - ★ Convolution models only the FMRI signal **changes** →



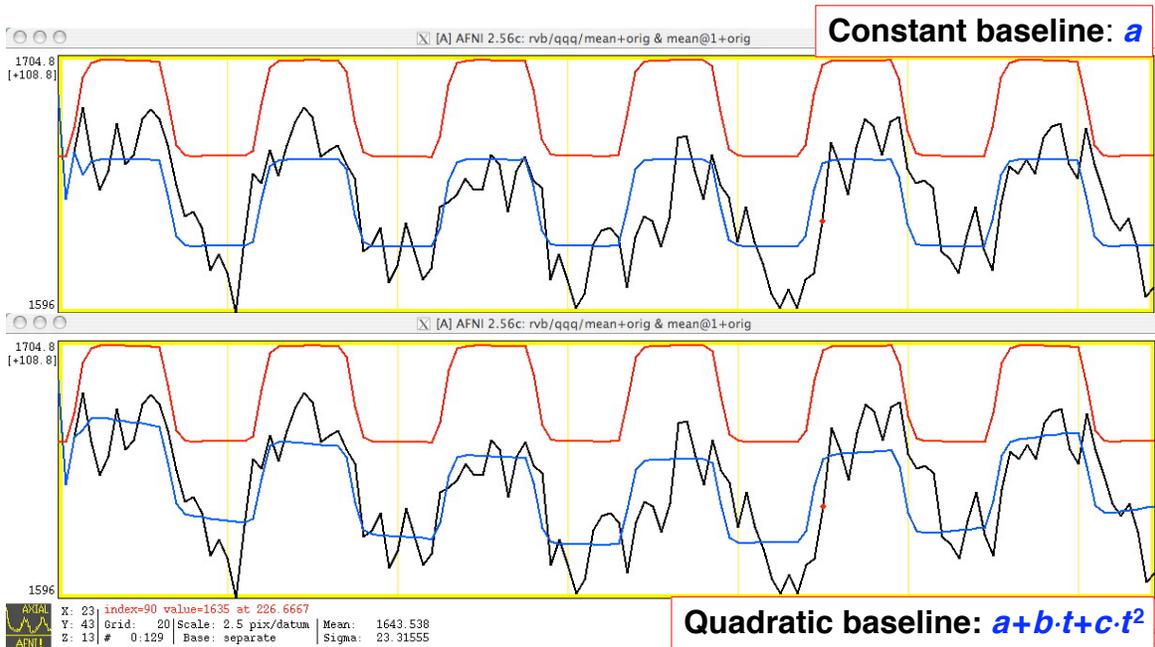
- Real data starts at and returns to a nonzero, slowly drifting baseline

Simple Regression Models

- Assume a fixed shape $h(t)$ for the HRF
 - ★ e.g., $h(t) = t^{8.6} \exp(-t/0.547)$ [MS Cohen, 1997]
 - ★ Convolve with stimulus timing to get ideal response function $r(t) = \sum_{k=1}^K h(t - \tau_k) =$ sum of HRF copies
- Assume a form for the baseline
 - ★ e.g., $a + b \cdot t$ for a constant plus a linear trend
- In each voxel, fit data $Z(t)$ to a curve of the form

$$\underline{Z(t) \approx a + b \cdot t + \beta \cdot r(t)}$$
 ← The signal model!
 - a, b, β are unknown parameters to be calculated in each voxel
 - a, b are “nuisance” parameters
 - β is amplitude of $r(t)$ in data = “how much” BOLD

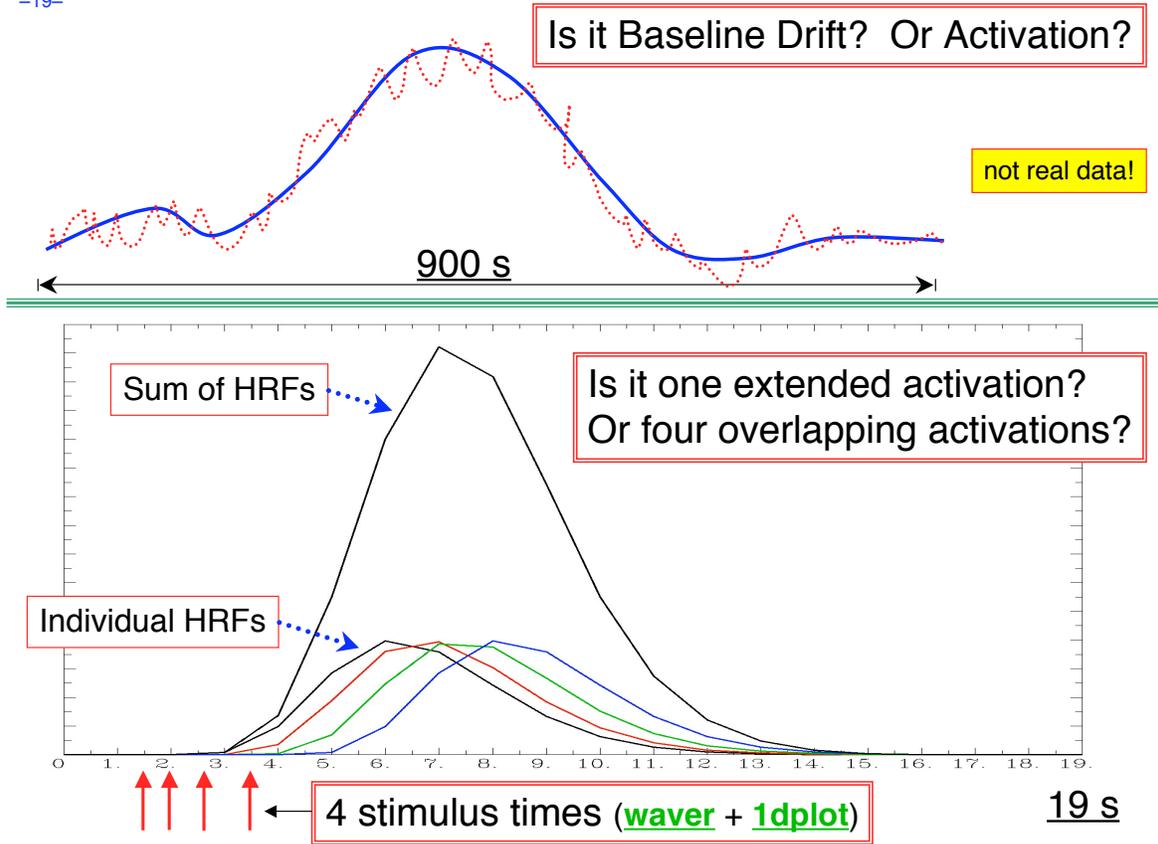
Simple Regression: Example



- Necessary baseline model complexity depends on duration of **continuous** imaging — e.g., 1 parameter per ~150 seconds

Duration of Stimuli - Important Caveats

- Slow baseline drift (time scale 100 s and longer) makes doing fMRI with **long duration** stimuli difficult
 - Learning experiment, where the task is done continuously for ~15 minutes and the subject is scanned to find parts of the brain that adapt during this time interval
 - Pharmaceutical challenge, where the subject is given some psychoactive drug whose action plays out over 10+ minutes (e.g., cocaine, ethanol)
- Multiple very **short duration** stimuli that are also very close in time to each other are very hard to tell apart, since their HRFs will have 90-95% overlap
 - Binocular rivalry, where percept switches ~ 0.5 s



Multiple Stimuli = Multiple Regressors

- Usually have more than one class of stimulus or activation in an experiment
 - ★ e.g., want to see size of “**face activation**” vis-à-vis “**house activation**”; or, “**what**” vs. “**where**” activity
- Need to model each separate class of stimulus with a separate response function $r_1(t)$, $r_2(t)$, $r_3(t)$,
 - ★ Each $r_j(t)$ is based on the stimulus timing for activity in class number j
 - ★ Calculate a β_j amplitude = amount of $r_j(t)$ in voxel data time series $Z(t)$
 - ★ Contrast β s to see which voxels have differential activation levels under different stimulus conditions
 - e.g., statistical test on the question $\beta_1 - \beta_2 = 0$?

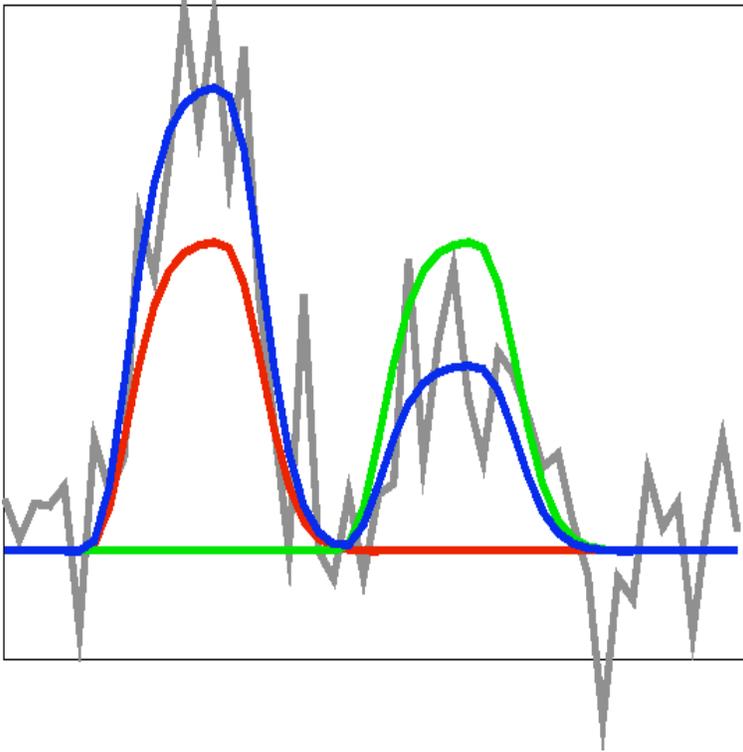
Multiple Stimuli - Important Caveat

- You do ***not*** model the baseline (“control”) condition
 - e.g., “rest”, visual fixation, high-low tone discrimination, or some other simple task
- fMRI can only measure ***changes*** in MR signal levels between tasks
 - So you need some simple-ish task to serve as a reference point
- The baseline model (e.g., $a + b \cdot t$) takes care of the signal level to which the MR signal returns when the “active” tasks are turned off
 - Modeling the reference task explicitly would be redundant (or “collinear”, to anticipate a forthcoming concept)

Multiple Stimuli - Experiment Design

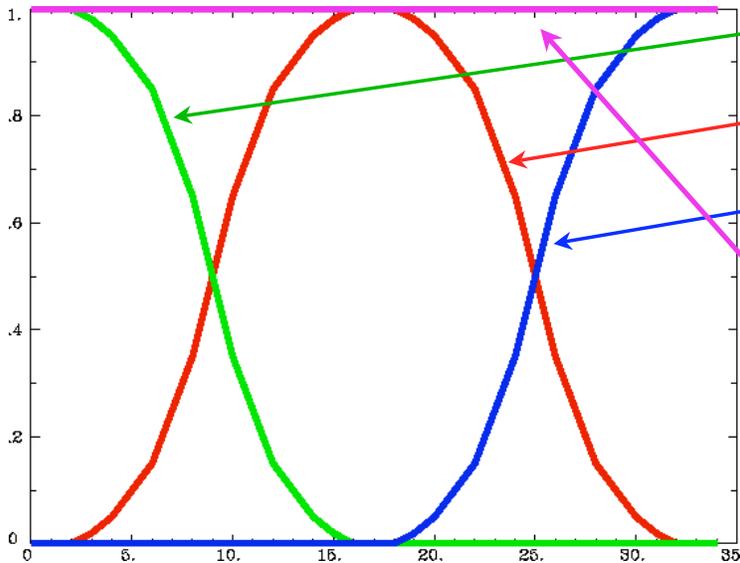
- How many distinct stimuli do you need in each class? Our rough recommendations:
 - Short event-related designs: at least 25 events in each stimulus class (spread across multiple imaging runs) — and more is better
 - Block designs: at least 5 blocks in each stimulus class — 10 would be better
- While we’re on the subject: **How many subjects?**
 - Several independent studies agree that 20-25 subjects in each category are needed for highly reliable results
 - This number is more than has usually been the custom in fMRI-based studies!

Multiple Regressors: Cartoon Animation



- **Red** curve = signal model for class #1
- **Green** curve = signal model for #2
- **Blue** curve = $\beta_1 \cdot \#1 + \beta_2 \cdot \#2$ where β_1 and β_2 vary from 0.1 to 1.7 in the animation
- Goal of regression is to find β_1 and β_2 that make the blue curve best fit the data time series
- **Gray** curve = $1.5 \cdot \#1 + 0.6 \cdot \#2 + \text{noise}$ = simulated data

Multiple Regressors: Collinearity!!

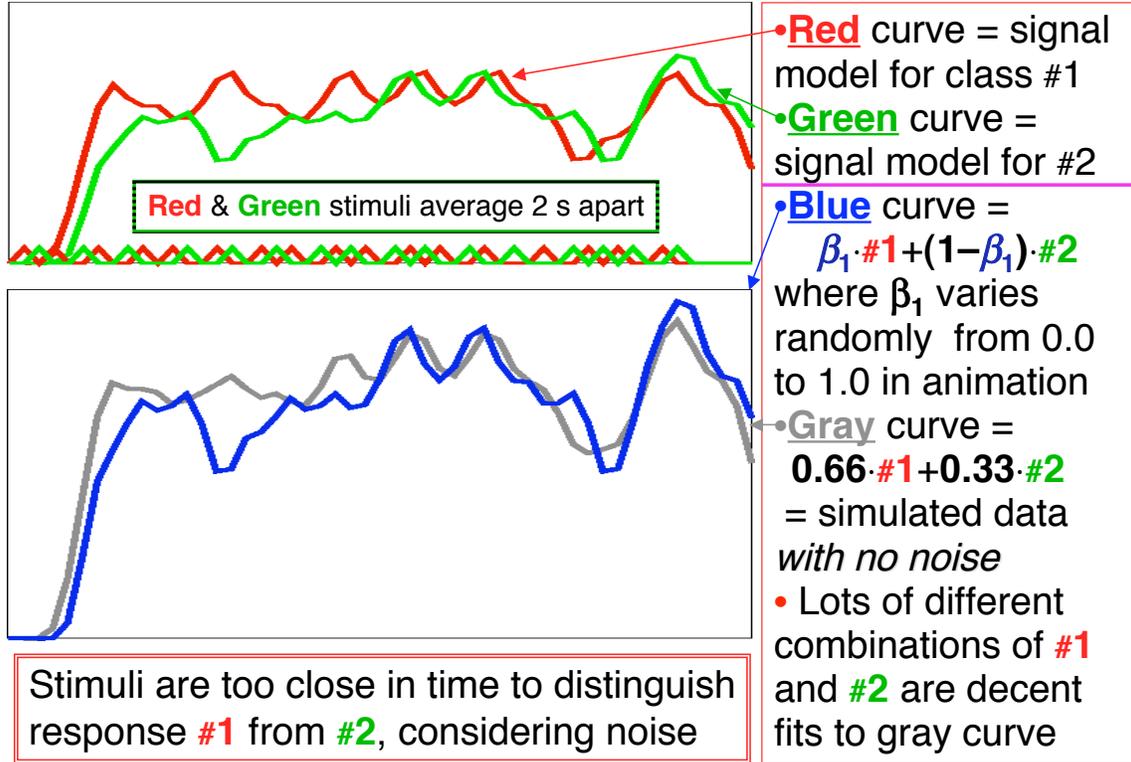


- **Green** curve = signal model for #1
- **Red** curve = signal model for class #2
- **Blue** curve = signal model for #3
- **Purple** curve = $\#1 + \#2 + \#3$ which is exactly = 1
- We cannot — *in principle or in practice* — distinguish sum of 3 signal models from constant baseline!!

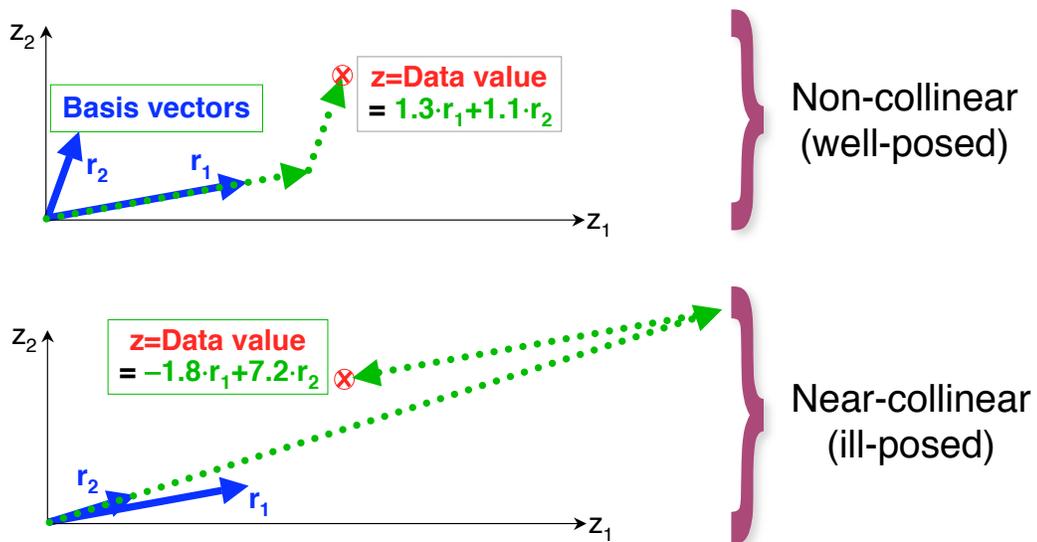
No analysis can distinguish the cases
 $Z(t) = 10 + 5 \cdot \#1$ and
 $Z(t) = 0 + 15 \cdot \#1 + 10 \cdot \#2 + 10 \cdot \#3$
 and an infinity of other possibilities

Collinear designs are **bad bad bad!**

Multiple Regressors: Near Collinearity

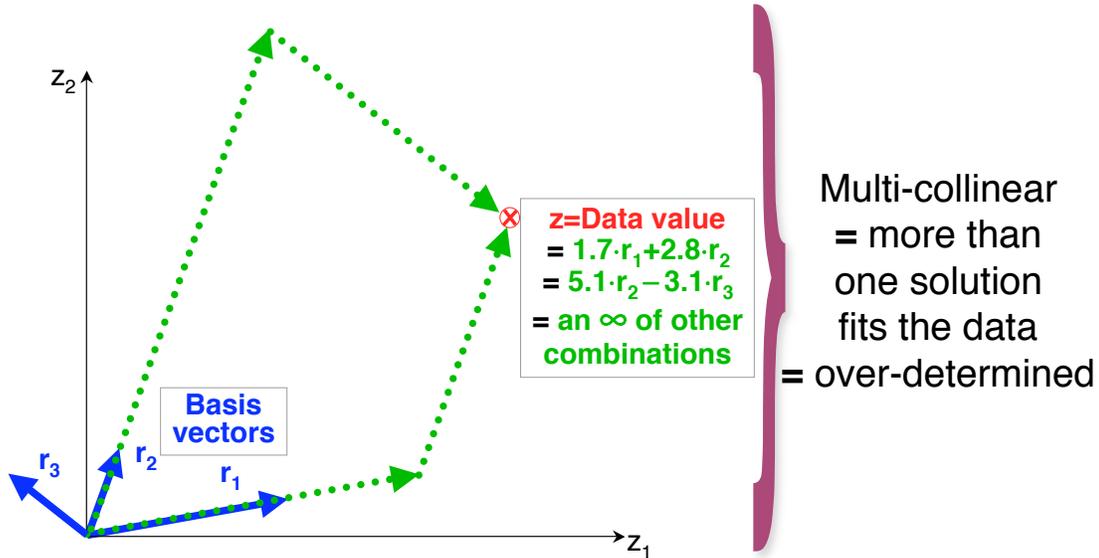


The Geometry of Collinearity - 1



- Trying to fit data as a sum of basis vectors that are nearly parallel doesn't work well: solutions can be huge
- Exactly parallel basis vectors would be impossible:
 - Determinant of matrix to invert would be zero

The Geometry of Collinearity - 2



- Trying to fit data with too many regressors (basis vectors) doesn't work: no unique solution

Equations: Notation

- Will approximately follow notation of manual for the AFNI program **3dDeconvolve**
- Time: continuous in reality, but in steps in the data
 - ★ Functions of continuous time are written like $f(t)$
 - ★ Functions of discrete time expressed like $f(\underbrace{n \cdot TR}_{=t_n})$ where $n=0,1,2,\dots$ and TR =time step
 - ★ Usually use subscript notion f_n as shorthand
 - ★ Collection of numbers assembled in a column is a

vector and is printed in boldface:

$$\left\{ \begin{array}{l} \text{vector of} \\ \text{length } N \end{array} \right\} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{N-1} \end{bmatrix} = \mathbf{f} \quad \begin{bmatrix} A_{00} & A_{01} & \cdots & A_{0,N-1} \\ A_{10} & A_{11} & \cdots & A_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ A_{M-1,0} & A_{M-1,1} & \cdots & A_{M-1,N-1} \end{bmatrix} = \mathbf{A} = \{M \times N \text{ matrix}\}$$

Equations: Single Response Function

- In each voxel, fit data Z_n to a curve of the form

$$Z_n \approx a + b \cdot t_n + \beta \cdot r_n \quad \text{for } n=0,1,\dots,N-1 \quad (N=\# \text{ time pts})$$
- a , b , β are unknown parameters to be calculated in each voxel

$$r_n = \sum_{k=1}^K h(t_n - \tau_k) = \text{sum of HRF copies}$$
- a, b are “nuisance” baseline parameters
- β is amplitude of $r(t)$ in data = “how much” BOLD
- Baseline model should be more complicated for long (> 150 s) continuous imaging runs: ≈ 1 param per 150 s
 - $150 < T < 300$ s: $a+b \cdot t+c \cdot t^2$
 - Longer: $a+b \cdot t+c \cdot t^2 + \lceil T/150 \rceil$ low frequency components
 - 3dDeconvolve uses Legendre polynomials for baseline
 - Often, also include as extra baseline components the estimated subject head movement time series, in order to remove residual contamination from such artifacts (will see example of this later)

Equations: Multiple Response Functions

- In each voxel, fit data Z_n to a curve of the form

$$Z_n \approx [\text{baseline}]_n + \beta_1 \cdot r_n^{(1)} + \beta_2 \cdot r_n^{(2)} + \beta_3 \cdot r_n^{(3)} + \dots$$
- β_j is amplitude in data of $r_n^{(j)} = r_j(t_n)$; i.e., “how much” of j^{th} response function in the data time series
- In simple regression, each $r_j(t)$ is derived directly from stimulus timing **and** user-chosen HRF model
 - In terms of stimulus times:

$$r_n^{(j)} = \sum_{k=1}^{K_j} h_j(t_n - \tau_k^{(j)}) = \text{sum of HRF copies}$$
 - Where $\tau_k^{(j)}$ is the k^{th} stimulus time in the j^{th} stimulus class
 - These times are input using the `-stim_times` option to program **3dDeconvolve**

Equations: Matrix-Vector Form

- Express **known** data vector as a sum of **known** columns with **unknown** coefficients:

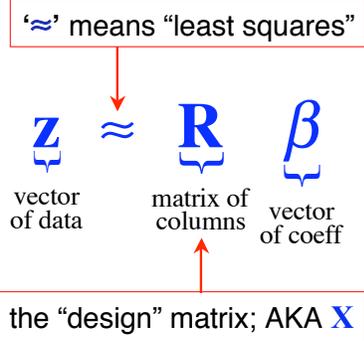
$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \\ z_{N-1} \end{bmatrix} \approx \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \cdot a + \begin{bmatrix} 0 \\ 1 \\ 2 \\ \vdots \\ N-1 \end{bmatrix} \cdot b + \begin{bmatrix} r_0^{(1)} \\ r_1^{(1)} \\ r_2^{(1)} \\ \vdots \\ r_{N-1}^{(1)} \end{bmatrix} \cdot \beta_1 + \begin{bmatrix} r_0^{(2)} \\ r_1^{(2)} \\ r_2^{(2)} \\ \vdots \\ r_{N-1}^{(2)} \end{bmatrix} \cdot \beta_2 + \dots$$

- Const baseline
- Linear trend

or

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \\ z_{N-1} \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & r_0^{(1)} & r_0^{(1)} & \dots \\ 1 & 1 & r_1^{(1)} & r_1^{(1)} & \dots \\ 1 & 2 & r_2^{(1)} & r_2^{(1)} & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ 1 & N-1 & r_{N-1}^{(1)} & r_{N-1}^{(2)} & \dots \end{bmatrix} \begin{bmatrix} a \\ b \\ \beta_1 \\ \beta_2 \\ \vdots \end{bmatrix}$$

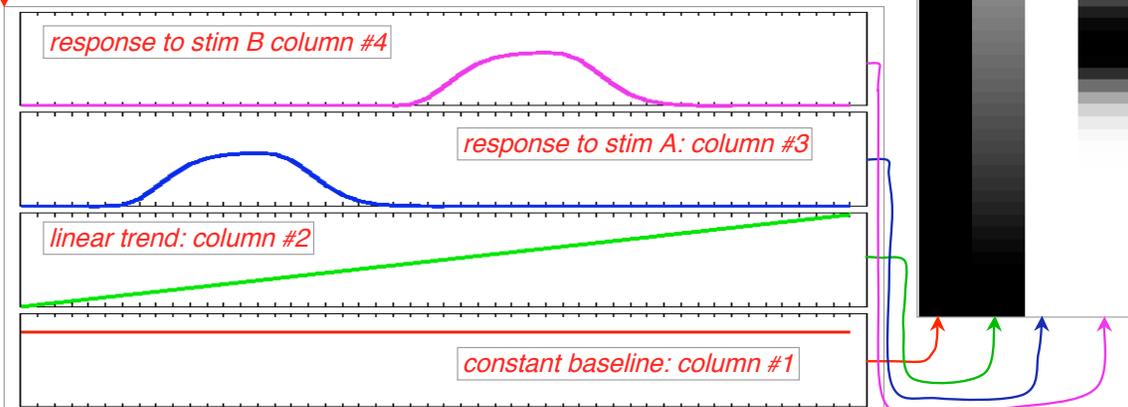
or



z depends on the voxel; **R** doesn't

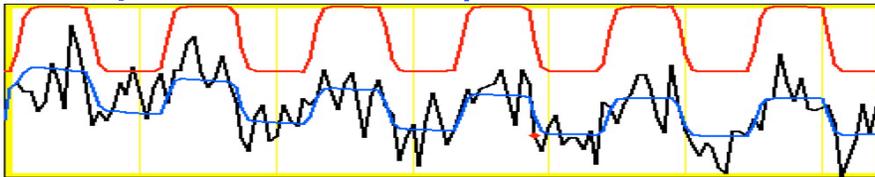
Visualizing the **R** Matrix

- Can graph columns (program **ldplot**)
 - But might have 20-50 columns
- Can plot columns on a grayscale (program **ldgrayplot**)
 - Easier way to show many columns
 - In this plot, darker bars means larger numbers



Solving $\mathbf{z} \approx \mathbf{R}\boldsymbol{\beta}$ for $\boldsymbol{\beta}$

- Number of equations = number of time points
 - ★ 100s per run, but perhaps 1000s per subject
- Number of unknowns usually in range 5–50
- Least squares solution: $\hat{\boldsymbol{\beta}} = [\mathbf{R}^T \mathbf{R}]^{-1} \mathbf{R}^T \mathbf{z}$
 - ★ $\hat{\boldsymbol{\beta}}$ denotes an *estimate* of the true (unknown) $\boldsymbol{\beta}$
 - ★ From $\hat{\boldsymbol{\beta}}$, calculate $\hat{\mathbf{z}} = \mathbf{R}\hat{\boldsymbol{\beta}}$ as the *fitted model*



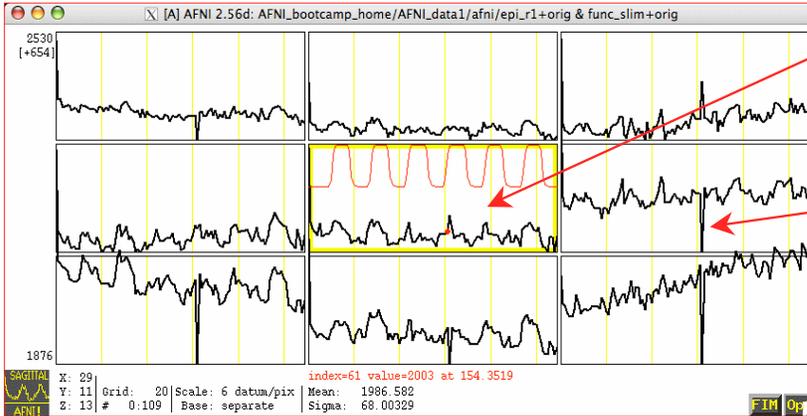
- $\mathbf{z} - \hat{\mathbf{z}}$ is the **residual time series** = noise (we hope)
 - Statistics measure how much each regressor helps reduce residuals
- Collinearity: when matrix $\mathbf{R}^T \mathbf{R}$ can't be inverted
 - ★ Near collinearity: when inverse exists but is huge

Simple Regression: Recapitulation

- Choose HRF model $h(t)$ [AKA *fixed-model regression*]
- Build model responses $r_n(t)$ to each stimulus class
 - ★ Using $h(t)$ and the stimulus timing
- Choose baseline model time series
 - ★ Constant + linear + quadratic (+ movement?)
- Assemble model and baseline time series into the columns of the \mathbf{R} matrix
- For each voxel time series \mathbf{z} , solve $\mathbf{z} \approx \mathbf{R}\boldsymbol{\beta}$ for $\hat{\boldsymbol{\beta}}$
- **Individual subject maps:** Test the coefficients in $\hat{\boldsymbol{\beta}}$ that you care about for statistical significance
- **Group maps:** Transform the coefficients in $\hat{\boldsymbol{\beta}}$ that you care about to Talairach space, and perform statistics on these $\hat{\boldsymbol{\beta}}$ values

Sample Data Analysis: Simple Regression

- Enough theory (for now: more to come later!)
- To look at the data: type `cd AFNI_data1/afni` ; then `afni`
- **Switch Underlay** to dataset `epi_r1`
 - ★ Then **Sagittal Image** and **Graph**
 - ★ **FIM**→**Pick Ideal** ; then click `afni/ideal_r1.1D` ; then **Set**
 - ★ Right-click in image, **Jump to (ijk)**, then `29 11 13`, then **Set**



- Data clearly has activity in sync with reference
 - 20 s blocks
- Data also has a big spike, which is very annoying
 - Subject head movement!

Preparing Data for Analysis

- Six preparatory steps are common:
 - ★ Image registration (AKA realignment): program `3dvolreg`
 - ★ Image smoothing: program `3dmerge`
 - ★ Image masking: program `3dClipLevel` or `3dAutomask`
 - ★ Conversion to percentile: programs `3dTstat` and `3dcalc`
 - ★ Censoring out time points that are bad: program `3dToutcount` or `3dTqual`
 - ★ Catenating multiple imaging runs into 1 big dataset: program `3dTcat`
-
- Not all steps are necessary or desirable in any given case
 - In this first example, will only do registration, since the data obviously needs this correction

Data Analysis Script

- In file `epi_r1_decon`:

```

3dvolreg -base 2 \
        -verb \
        -prefix epi_r1_reg \
        -1Dfile epi_r1_mot.1D \
        epi_r1+orig

```

- `3dvolreg` (3D image registration) will be covered in detail in a later presentation
- filename to get estimated motion parameters

```

3dDeconvolve \
  -input epi_r1_reg+orig \
  -nfirst 2 \
  -num_stimts 1 \
  -stim_times 1 epi_r1_times.1D \
    'BLOCK(20)' \
  -stim_label 1 AllStim \
  -tout \
  -bucket epi_r1_func \
  -fitts epi_r1_fitts \
  -xjpeg epi_r1_Xmat.jpg \
  -x1D epi_r1_Xmat.1D

```

- `3dDeconvolve` = regression code
- Name of input dataset (from `3dvolreg`)
- Index of first sub-brick to process
- Number of input model time series
- Name of input stimulus class timing file (τ 's) and type of HRF model to fit
- Name for results in AFNI menus
- Indicates to output t -statistic for β weights
- Name of output "bucket" dataset (statistics)
- Name of output model fit dataset
- Name of image file to store X (AKA R) matrix
- Name of text (.1D) file to store X matrix

- Type `tcsh epi_r1_decon` ; then wait for programs to run

Text Output of the `epi_r1_decon` script

- `3dvolreg` output

```

++ Program 3dvolreg: AFNI version=AFNI_2005_12_30_0934 [32-bit]
++ Reading input dataset ./epi_r1+orig.BRIK
++ Edging: x=3 y=3 z=1
++ Initializing alignment base
++ Starting final pass on 110 sub-bricks: 0..1..2..3.. *** ..106..107..108..109..
++ CPU time for realignment=8.82 s [=0.0802 s/sub-brick]
++ Min : roll=-0.086 pitch=-0.995 yaw=-0.325 dS=-0.310 dL=-0.010 dP=-0.680
++ Mean: roll=-0.019 pitch=-0.020 yaw=-0.182 dS=+0.106 dL=+0.085 dP=-0.314
++ Max : roll=+0.107 pitch=+0.090 yaw=+0.000 dS=+0.172 dL=+0.204 dP=+0.079
++ Max displacement in automask = 2.05 (mm) at sub-brick 62 } Maximum movement
++ Wrote dataset to disk in ./epi_r1_reg+orig.BRIK

```

- `3dDeconvolve` output

```

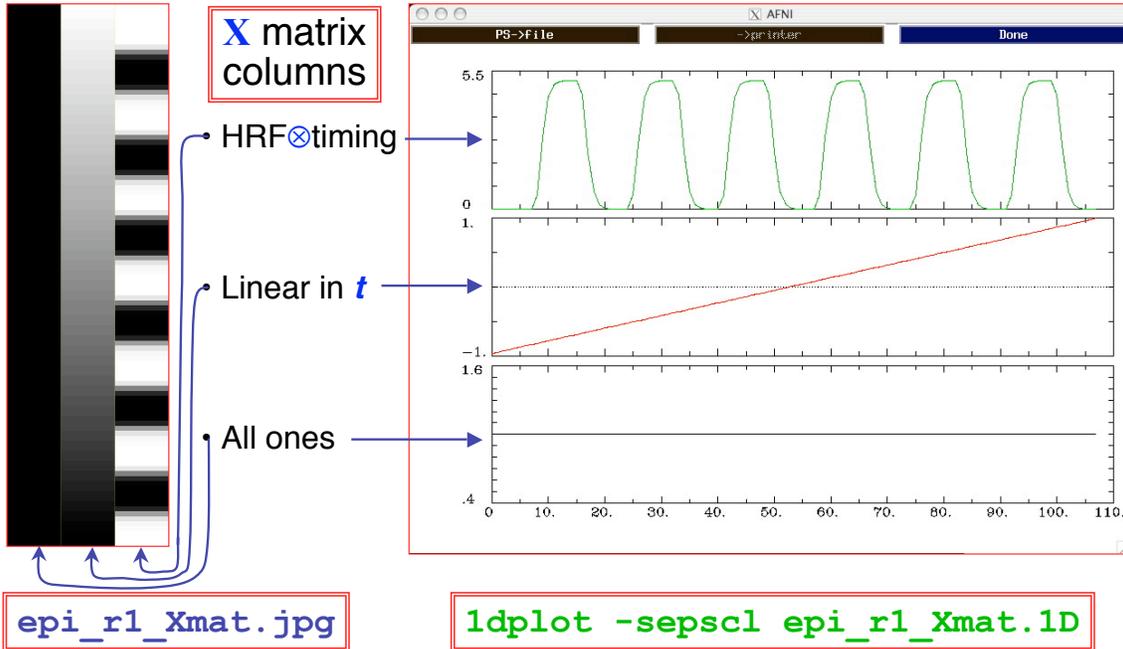
++ Program 3dDeconvolve: AFNI version=AFNI_2006_12_22_0933 [32-bit]
++ -stim_times 1 using local times
++ Wrote matrix image to file epi_r1_Xmat.jpg } Output file indicators
++ Wrote matrix values to file epi_r1_Xmat.1D }
++ (108x3) Matrix condition [X]: 2.44244 } Matrix Quality Control: Good values
++ Matrix inverse average error = 3.97804e-16 }
++ Wrote matrix image to file epi_r1_Xmat_psinv.jpg } Output file indicators
++ Wrote matrix values to file epi_r1_Xmat_psinv.1D }
++ Matrix setup time = 0.76 s
++ Calculations starting; elapsed time=1.247
++ voxel loop:0123456789.0123456789.0123456789.0123456789.0123456789. } Progress meter
++ Calculations finished; elapsed time=2.238
++ Wrote bucket dataset into ./epi_r1_func+orig.BRIK } Output file indicators
++ Wrote 3D+time dataset into ./epi_r1_fitts+orig.BRIK }
++ #Flops=4.18043e+08 Average Dot Product=4.56798

```

- If a program crashes, we'll need to see this text output (at the very least)!

Stimulus Timing: Input and Visualization

`epi_r1_times.1D` = 22.5 65.0 105.0 147.5 190.0 232.5
 = times of start of each **BLOCK (20)**



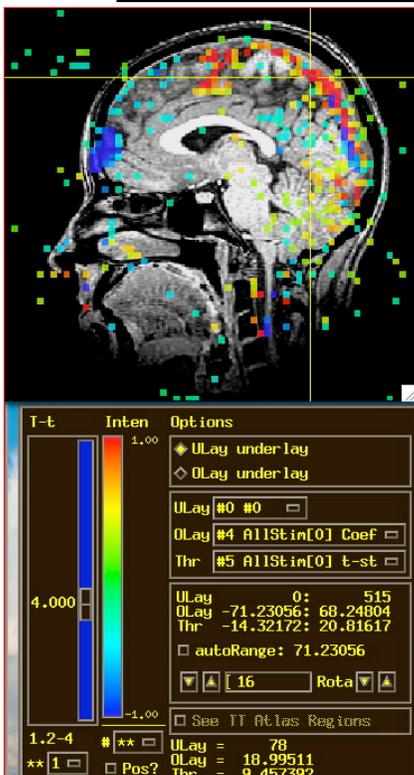
Look at the Activation Map

- Run **afni** to view what we've got
 - ★ **Switch Underlay** to `epi_r1_reg` (output from **3dvolreg**)
 - ★ **Switch Overlay** to `epi_r1_func` (output from **3dDeconvolve**)
 - ★ **Sagittal Image** and **Graph** viewers
 - ★ **FIM** → **Ignore** → **2** to have graph viewer not plot 1st 2 time pts
 - ★ **FIM** → **Pick Ideal** ; pick `epi_r1_ideal.1D` (output from **waver**)
- **Define Overlay** to set up functional coloring
 - **Olay** → **Allstim[0] Coef** (sets coloring to be from model fit β)
 - **Thr** → **Allstim[0] t-s** (sets threshold to be model fit t -statistic)
 - **See Overlay** (otherwise won't see the function!)
 - Play with threshold slider to get a meaningful activation map (e.g., $t=4$ is a decent threshold — more on thresholds later)

More Looking at the Results

- Graph viewer: **Opt**→**Tran 1D**→**Dataset #N** to plot the model fit dataset output by **3dDeconvolve**
 - Will open the control panel for the **Dataset #N** plugin
 - Click first **Input** on ; then choose **Dataset epi_r1_fitts+orig**
 - Also choose **Color dk-blue** to get a pleasing plot
 - Then click on **Set+Close** (to close the plugin panel)
 - Should now see fitted time series in the graph viewer instead of data time series
 - Graph viewer: click **Opt**→**Double Plot**→**Overlay** on to make the fitted time series appear as an overlay curve
 - This tool lets you visualize the quality of the data fit
- Can also now overlay function on MP-RAGE anatomical by using **Switch Underlay** to **anat+orig** dataset
 - Probably won't want to graph the **anat+orig** dataset!

Stimulus Correlated Movement?



- Extensive “activation” (i.e., correlation of data time series with model time series) along top of brain is an indicator of stimulus correlated motion artifact
- Can remain even after image registration, due to errors in alignment process, magnetic field inhomogeneities, etc.
- Can be *partially* removed by using estimated movement history (from **3dvolreg**) as extra baseline model functions

- **3dvolreg** saved the motion parameters estimates into file **epi_r1_mot.1D**
- For fun: **1dplot epi_r1_mot.1D**

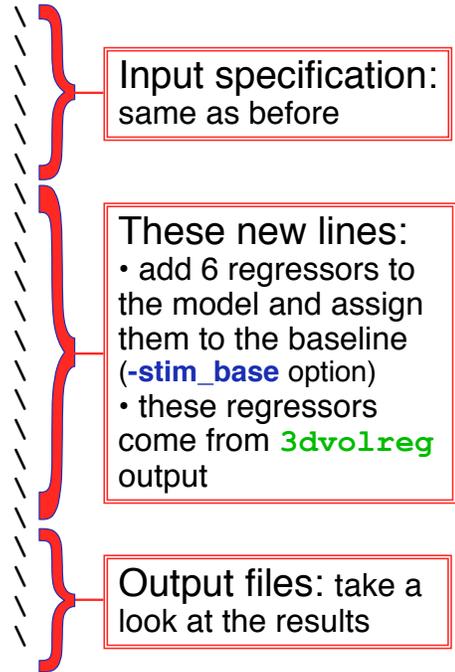
Removing Residual Motion Artifacts

- Script `epi_r1_decon_mot`:

```

3dDeconvolve
-input epi_r1_reg+orig
-nfirst 2
-num_stimts 7
-stim_times 1 epi_r1_times.1D 'BLOCK(20)'
-stim_label 1 AllStim
-stim_file 2 epi_r1_mot.1D'[0]'
-stim_base 2
-stim_file 3 epi_r1_mot.1D'[1]'
-stim_base 3
-stim_file 4 epi_r1_mot.1D'[2]'
-stim_base 4
-stim_file 5 epi_r1_mot.1D'[3]'
-stim_base 5
-stim_file 6 epi_r1_mot.1D'[4]'
-stim_base 6
-stim_file 7 epi_r1_mot.1D'[5]'
-stim_base 7
-tout
-bucket epi_r1_func_mot
-fitts epi_r1_fitts_mot
-xjpeg epi_r1_Xmat_mot.jpg
-x1D epi_r1_Xmat_mot.1D

```

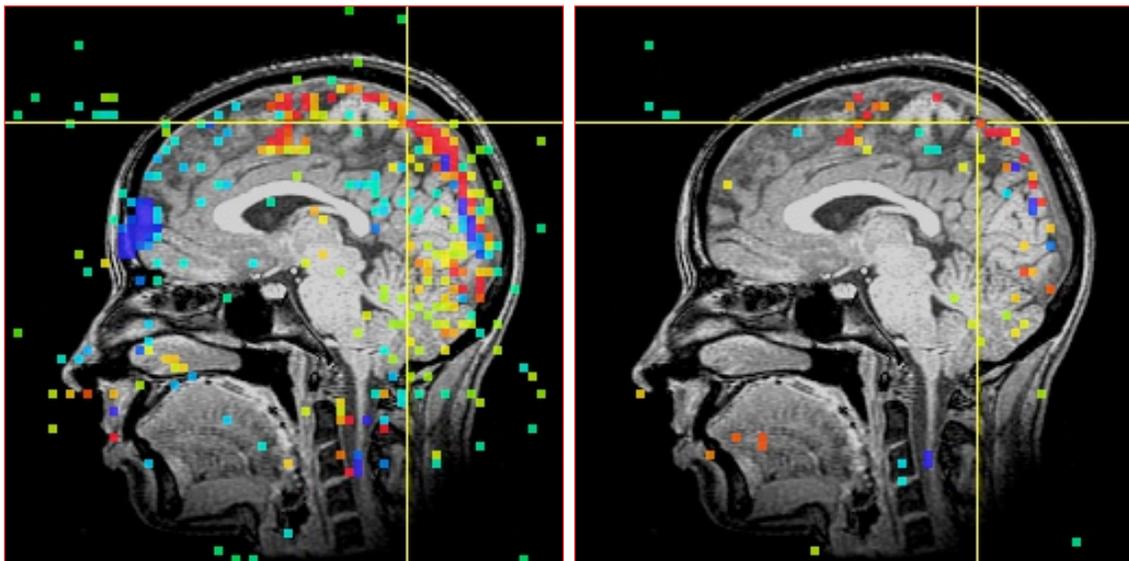


Input specification:
same as before

These new lines:
 • add 6 regressors to the model and assign them to the baseline (-stim_base option)
 • these regressors come from `3dvolreg` output

Output files: take a look at the results

Some Results: Before and After



Before: movement parameters *are not* in baseline model

After: movement parameters *are* in baseline model

t-statistic threshold set to (uncorrected) *p*-value of 10^{-4} in both images
 Before: 105 degrees of freedom ($t=4.044$) | After: $105-6 = 99$ DOF ($t=4.054$)

Setting the Threshold: Principles

- Bad things (i.e., errors):
 - False positives — activations reported that aren't really there \equiv **Type I errors** (i.e., activations from noise-only data)
 - False negatives — non-activations reported where there should be true activations found \equiv **Type II errors**
- Usual approach in statistical testing is to control the probability of a type I error (the " p -value")
- In FMRI, we are making many statistical tests: one per voxel ($\approx 20,000+$) — the result of which is an "activation map":
 - Voxels are colorized if they survive the statistical thresholding process

Important Aside

Setting the Threshold: Bonferroni

- If we set the threshold so there is a 1% chance that any given voxel is declared "active" even if its data is pure noise (FMRI jargon: "uncorrected" p -value is 0.01):
 - And assume each voxel's noise is independent of its neighbors (not really true)
 - With 30,000 voxels to threshold, would expect to get 300 false positives — this may be as many as the true activations! Situation: **Not so good.**
- Bonferroni solution: set threshold (e.g., on t -statistic) so high that uncorrected p -value is $0.05/20000=2.5e-6$
 - Then have only a 5% chance that even a single false positive voxel will be reported
 - **Objection:** will likely lose weak areas of activation

Important Aside

Setting the Threshold: Spatial Clustering

- Cluster-based detection lets us lower the statistical threshold and still control the false positive rate
- **Two** thresholds:
 - **First**: a per-voxel threshold that is somewhat low (so by itself leads to a lot of false positives, scattered around)
 - **Second**: form clusters of spatially contiguous (neighboring) voxels that survive the first threshold, and keep only those clusters above a volume threshold — e.g., we don't keep isolated "active" voxels
- Usually: choose volume threshold, then calculate voxel-wise statistic threshold to get the overall "corrected" p -value you want (typically, corrected $p=0.05$)
 - No easy formulas for this type of dual thresholding, so must use simulation: AFNI program **AlphaSim**

Important Aside

AlphaSim: Clustering Thresholds

- Simulated for brain mask of 18,465 voxels
- Look for smallest cluster with corrected $p < 0.05$

Uncorrected p -value (per voxel)	Cluster Size / Corrected p (uncorrelated)	Cluster Size / Corrected p (correlated 5 mm)
0.0002	2 / 0.001	3 / 0.004
0.0004	2 / 0.008	4 / 0.012
0.0007	2 / 0.026	3 / 0.031
0.0010	3 / 0.001	4 / 0.007
0.0020	3 / 0.003	4 / 0.032
0.0030	3 / 0.008	5 / 0.013
0.0040	3 / 0.018	5 / 0.029
0.0050	3 / 0.030	6 / 0.012
0.0060	4 / 0.003	6 / 0.023
0.0070	4 / 0.004	6 / 0.036
0.0080	4 / 0.006	7 / 0.016
0.0090	4 / 0.010	7 / 0.027
0.0100	4 / 0.015	7 / 0.042

Corresponds to sample data

Can make activation maps for display with cluster editing using **3dmerge** program or in AFNI GUI (new: Sep 2006)

Important Aside

Multiple Stimulus Classes

- The experiment analyzed here in fact is more complicated
 - ★ There are 4 related visual stimulus types
 - **Actions** & **Tools** = active conditions (videos of complex movements and simple tool-like movements)
 - **HighC** & **LowC** = control conditions (moving high and low contrast gratings)
 - 6 blocks of 20 s duration in each condition (with fixation between blocks)
 - ★ One goal is to find areas that are differentially activated between these different types of stimuli
 - That is, differential activation levels between **Actions** and **Tools**, and between **Actions** and **Tools** combined vs. **HighC** and **LowC** combined
 - ★ We have 4 imaging runs, 108 useful time points in each (skipping first 2 in each run) that we will analyze together
 - Already registered and put together into dataset `rall_vr+orig`
 - The first 2 time points in each run have been cut out in this dataset

Regression with Multiple Model Files

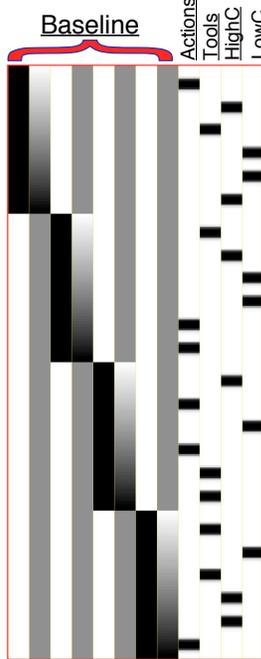
- Script file `rall_decon` does the job:

```
3dDeconvolve -input rall_vr+orig -concat '1D: 0 108 216 324' \
-num_stimts 4 \
-stim_times 1 '1D: 17.5 | 185.0 227.5 | 60.0 142.5 | 227.5' \
               'BLOCK(20,1)' \
-stim_times 2 '1D: 100.0 | 17.5 | 185.0 227.5 | 17.5 100.0' \
               'BLOCK(20,1)' \
-stim_times 3 '1D: 60.0 227.5 | 60.0 | 17.5 | 142.5 185.0' \
               'BLOCK(20,1)' \
-stim_times 4 '1D: 142.5 185.0 | 100.0 142.5 | 100.0 | 60.0' \
               'BLOCK(20,1)' \
-stim_label 1 Actions -stim_label 2 Tools \
-stim_label 3 HighC -stim_label 4 LowC \
-gltsym 'SYM: Actions -Tools' -glt_label 1 AvsT \
-gltsym 'SYM: HighC -LowC' -glt_label 2 HvsL \
-gltsym 'SYM: Actions Tools -HighC -LowC' -glt_label 3 ATvsHL \
-full_first -fout -tout -nobout \
-bucket func_rall -fitts fitts_rall \
-xjpeg xmat_rall.jpg -x1D xmat_rall.1D
```

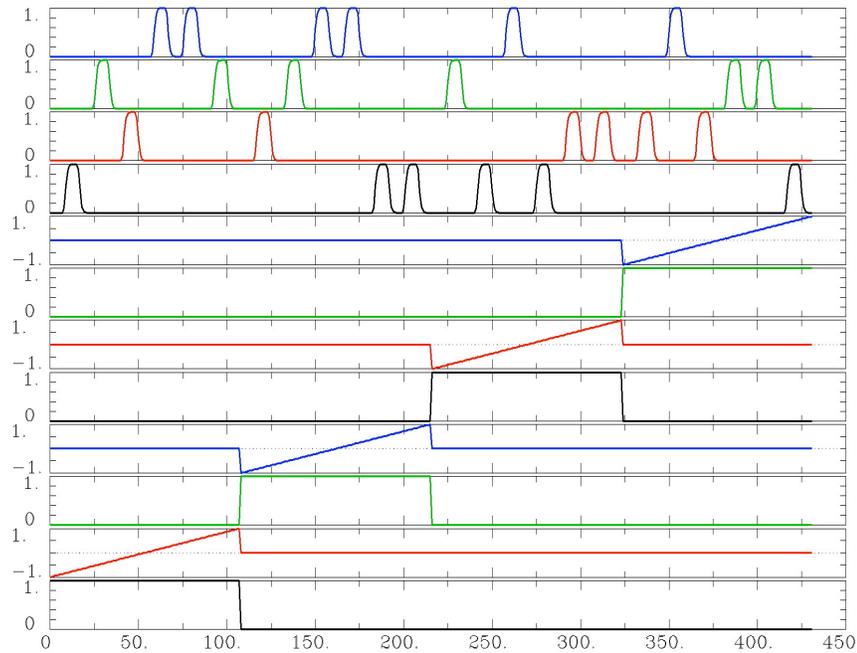
\ ← Run lengths
 \ ← τ's on command line instead of file:
 \ | indicates a new "line" (1 line of stimulus start times per run)
 \ ← generate **Actions-Tools** statistical map
 \ ← specify statistics types to output

- Run this script by typing `tcsh rall_decon` (takes a few minutes)

Regressor Matrix for This Script



via `-xjpeg`



via `-x1D` and `1dplot -sep_scl xmat_rall.1D`

Novel Features of 3dDeconvolve - 1

`-concat '1D: 0 108 216 324'`

- “File” that indicates where distinct imaging runs start inside the input file
 - ★ Numbers are the time indexes inside the file for start of runs
 - ★ In this case, a .1D file put directly on the command line
 - Could also be a filename, if you want to store that data externally

`-num_stimts 4`

- We have 4 stimulus classes, so will need 4 `-stim_times` below

`-stim_times 1`

`'1D: 17.5 | 185.0 227.5 | 60.0 142.5 | 227.5'`
`'BLOCK (20,1)'`

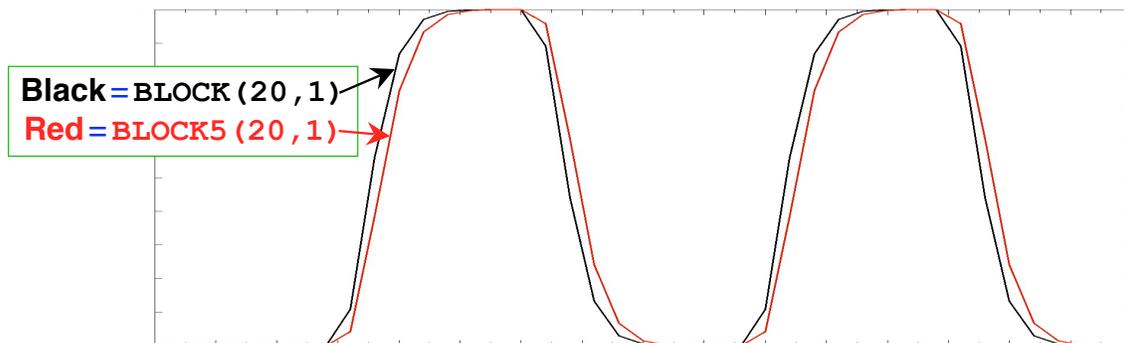
- “File” with 4 lines, each line specifying the start time in seconds for the stimuli within the corresponding imaging run, with the time measured relative to the start of the imaging run itself
- HRF for each block stimulus is now specified to go to maximum value of 1 (compare to graphs on previous slide)
 - ★ This feature is useful when converting FMRI response magnitude to be in units of percent of the mean baseline

Aside: the 'BLOCK ()' HRF Model

- **BLOCK (L)** is convolution of square wave of duration **L** with “gamma variate function” $t^4 e^{-t} / [4^4 e^{-4}]$ (peak value=1 at $t=4$):

$$h(t) = \int_0^{\min(t,L)} s^4 e^{-s} / [4^4 e^{-4}] ds$$

- “Hidden” option: **BLOCK5** replaces “4” with “5” in the above
 - Slightly more delayed rise and fall times
- **BLOCK (L, 1)** makes peak amplitude of *block* response = 1



Novel Features of 3dDeconvolve - 2

```
-gltsym 'SYM: Actions -Tools' -glt_label 1 AvsT
```

- **GLTs** are **General Linear Tests**
- **3dDeconvolve** provides test statistics for each regressor and stimulus class separately, but if you want to test combinations or contrasts of the β weights in each voxel, you need the **-gltsym** option
- Example above tests the difference between the β weights for the **Actions** and the **Tools** responses
 - ★ Starting with **SYM:** means symbolic input is on command line
 - Otherwise inputs will be read from a file
 - ★ Symbolic names for each stimulus class are taken from **-stim_label** options
 - ★ Stimulus label can be preceded by **+** or **-** to indicate sign to use in combination of β weights
- Goal is to test a linear combination of the β weights
 - Tests if $\beta_{\text{Actions}} - \beta_{\text{Tools}} \neq 0$
 - e.g., does **Actions** get a bigger response than **Tools** ?
- Quiz: what would 'SYM: Actions +Tools' test? It would test if $\beta_{\text{Actions}} + \beta_{\text{Tools}} \neq 0$

Novel Features of 3dDeconvolve - 3

```
-gltsym 'SYM: Actions Tools -HighC -LowC'
-glt_label 1 ATvsHL
```

- Goal is to test if $(\beta_{\text{Actions}} + \beta_{\text{Tool}}) - (\beta_{\text{HighC}} + \beta_{\text{LowC}}) \neq 0$
 - Regions where this statistic is significant have different amounts of BOLD signal change in the more complex activity viewing tasks versus the grating viewing tasks
 - This is a way to factor out primary visual cortex, and see what areas are differentially more (or less?) responsive to complicated movements
- `-glt_label 3 ATvsHL` option is used to attach a meaningful label to the resulting statistics sub-bricks
 - ★ Output includes the ordered summation of the β weights and the associated statistical parameters (t - and/or F -statistics)

Novel Features of 3dDeconvolve - 4

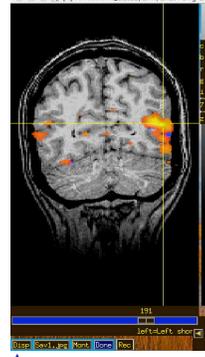
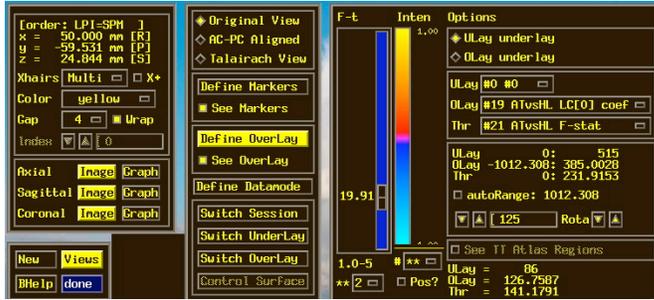
`-full_first` = put **full model** statistic first in output `-bucket` dataset, not last (for convenience)

`-fout -tout` = output both F - and t -statistics for each stimulus class and regressor (but not the statistics for the baseline regressors)

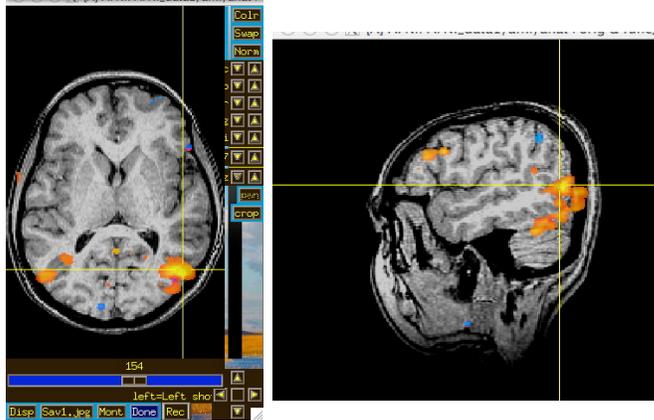
`-nobout`

- The full model statistic is an F -statistic that shows how well the sum of all 4 input model time series fits voxel time series data
 - ★ Compared to how well *just* the baseline model time series fit the data times (in this example, have 8 baseline regressors)
- The individual stimulus classes also will get individual F - and/or t -statistics indicating the significance of their individual incremental contributions to the data time series fit
 - ★ F_{Actions} tells if model **{Actions+HighC+LowC+Tools+baseline}** explains more of the data variability than model **{HighC+LowC+Tools+baseline}** — with **Actions** omitted

Results of **rall_decon** Script



```
# 0 Full F-stat
# 1 Actions[0] Coef
# 2 Actions[0] t-st
# 3 Actions F-stat
# 4 Tools[0] Coef
# 5 Tools[0] t-st
# 6 Tools F-stat
# 7 HighC[0] Coef
# 8 HighC[0] t-st
# 9 HighC F-stat
#10 LowC[0] Coef
#11 LowC[0] t-st
#12 LowC F-stat
#13 AvsT LC[0] coef
#14 AvsT LC[0] t-st
#15 AvsT F-stat
#16 HvsL LC[0] coef
#17 HvsL LC[0] t-st
#18 HvsL F-stat
#19 ATvsHL LC[0] coef
#20 ATvsHL LC[0] t-st
#21 ATvsHL F-stat
```



- Menu showing labels from **3dDeconvolve** run
- Images showing results from third GLT contrast: **ATvsHL**
- Play with these results yourself

Statistics from 3dDeconvolve

- An *F*-statistic measures significance of how much a model component (stimulus class) reduced the variance (sum of squares) of data time series residual
 - ★ After all the other model components were given their chance to reduce the variance
 - ★ **Residuals** \equiv data - model fit = errors = **-errts**
 - ★ A *t*-statistic sub-brick measures impact of one coefficient (of course, **BLOCK** has only one coefficient)
- Full *F* measures how much the all signal regressors combined reduced the variance over just the baseline regressors (**sub-brick #0**)
- Individual partial-model *F*s measures how much each individual signal regressor reduced data variance over the full model with that regressor excluded (**sub-bricks #3, #6, #9, and #12**)
- The **Coef** sub-bricks are the β weights (e.g., **#1, #4, #7, #10**) for the individual regressors
- Also present: GLT coefficients (**LC**) and statistics

```
# 0 Full F-stat
# 1 Actions[0] Coef
# 2 Actions[0] t-st
# 3 Actions F-stat
# 4 Tools[0] Coef
# 5 Tools[0] t-st
# 6 Tools F-stat
# 7 HighC[0] Coef
# 8 HighC[0] t-st
# 9 HighC F-stat
#10 LowC[0] Coef
#11 LowC[0] t-st
#12 LowC F-stat
#13 AvsT LC[0] coef
#14 AvsT LC[0] t-st
#15 AvsT F-stat
#16 HvsL LC[0] coef
#17 HvsL LC[0] t-st
#18 HvsL F-stat
#19 ATvsHL LC[0] coef
#20 ATvsHL LC[0] t-st
#21 ATvsHL F-stat
```

Group Analysis: carried out on β or **LC** coefs from single-subject analyses

Deconvolution Signal Models

- Simple or Fixed-shape regression (previous):
 - ★ We fixed the shape of the HRF — amplitude varies
 - ★ Used `-stim_times` to generate the signal model from the stimulus timing
 - ★ Found the amplitude of the signal model in each voxel — solution to the set of linear equations = β weights
- Deconvolution or Variable-shape regression (now):
 - ★ We allow the shape of the HRF to vary in each voxel, for each stimulus class
 - ★ Appropriate when you don't want to over-constrain the solution by assuming an HRF shape
 - ★ **Caveat:** need to have enough time points during the HRF in order to resolve its shape

Deconvolution: Pros & Cons (+ & -)

- + Letting HRF shape varies allows for subject and regional variability in hemodynamics
- + Can test HRF estimate for different shapes (e.g., are later time points more “active” than earlier?)
- Need to estimate more parameters for each stimulus class than a fixed-shape model (e.g., 4-15 vs. 1 parameter = amplitude of HRF)
- Which means you need more data to get the same statistical power (assuming that the fixed-shape model you would otherwise use was in fact “correct”)
- Freedom to get any shape in HRF results can give weird shapes that are difficult to interpret

Expressing HRF via Regression Unknowns

- The tool for expressing an unknown function as a finite set of numbers that can be fit via linear regression is an **expansion in basis functions**

$$h(t) = \beta_0 \psi_0(t) + \beta_1 \psi_1(t) + \beta_2 \psi_2(t) + \dots = \sum_{q=0}^{q=p} \beta_q \psi_q(t)$$

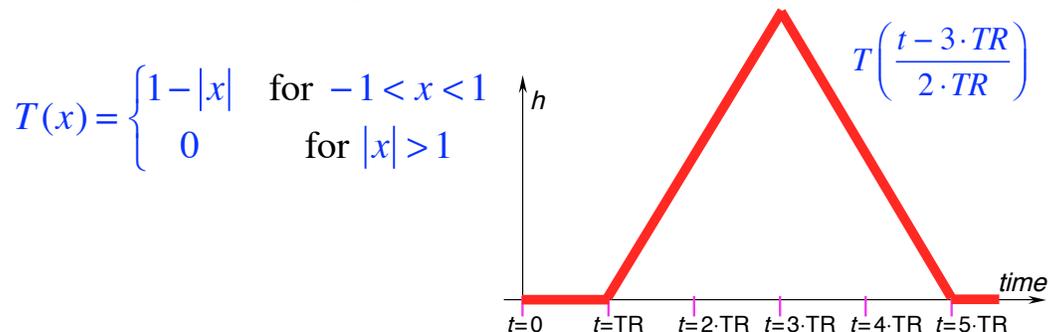
- ★ The basis functions $\psi_q(t)$ & expansion order p are known
 - Larger $p \Rightarrow$ more complex shapes & more parameters
- ★ The unknowns to be found (in each voxel) comprises the set of weights β_q for each $\psi_q(t)$
- β weights appear only by multiplying known values, and HRF only appears in signal model by linear convolution (addition) with known stimulus timing
 - Resulting signal model still solvable by linear regression

3dDeconvolve with “Tent Functions”

- Need to describe HRF shape and magnitude with a finite number of parameters
 - ★ And allow for calculation of $h(t)$ at any arbitrary point in time after the stimulus times:

$$r_n = \sum_{k=1}^K h(t_n - \tau_k) = \text{sum of HRF copies}$$

- Simplest set of such functions are **tent functions**
 - ★ Also known as “piecewise linear B-splines”

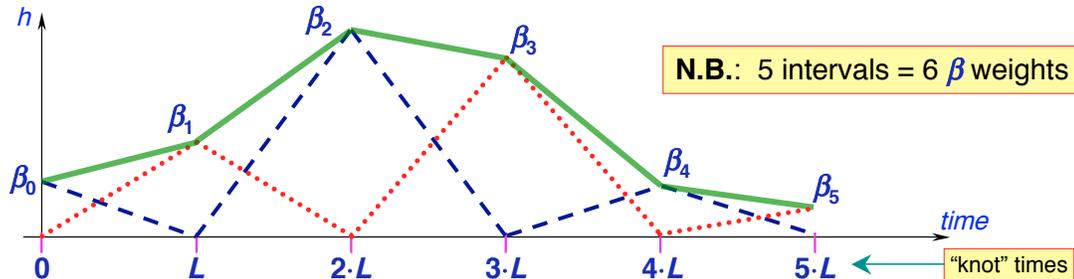


Tent Functions = Linear Interpolation

A

- Expansion of HRF in a set of spaced-apart tent functions is the same as linear interpolation between “knots”

$$h(t) = \beta_0 \cdot T\left(\frac{t}{L}\right) + \beta_1 \cdot T\left(\frac{t-L}{L}\right) + \beta_2 \cdot T\left(\frac{t-2 \cdot L}{L}\right) + \beta_3 \cdot T\left(\frac{t-3 \cdot L}{L}\right) + \dots$$



- Tent function parameters are also easily interpreted as function values (e.g., β_2 = response at time $t = 2 \cdot L$ after stim)
- User must decide on relationship of tent function grid spacing L and time grid spacing TR (usually would choose $L \geq TR$)
- In **3dDeconvolve**: specify duration of HRF and number of β parameters (details shown in a few slides)

Tent Functions: Average Signal Change

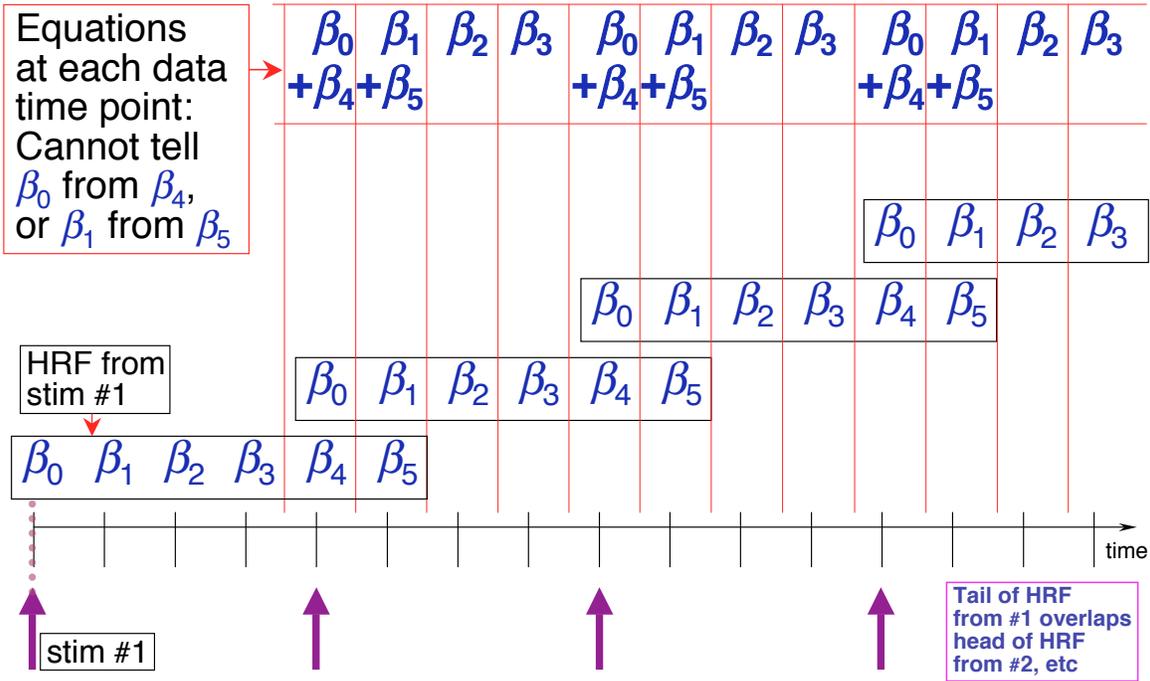
- For input to group analysis, usually want to compute average signal change
 - ★ Over entire duration of HRF (usual)
 - ★ Over a sub-interval of the HRF duration (sometimes)
- In previous slide, with 6 β weights, average signal change is

$$\frac{1}{2} \beta_0 + \beta_1 + \beta_2 + \beta_3 + \beta_4 + \frac{1}{2} \beta_5$$

- First and last β weights are scaled by half since they only affect half as much of the duration of the response
- In practice, may want to use $0 \cdot \beta_0$ since immediate post-stimulus response is not neuro-hemodynamically relevant
- All β weights (for each stimulus class) are output into the “bucket” dataset produced by **3dDeconvolve**
- Can then be combined into a single number using **3dcalc**

Deconvolution and Collinearity

- Regular stimulus timing can lead to collinearity!



Deconvolution Example - The Data

- `cd AFNI_data2`
 - ★ data is in `ED/` subdirectory (10 runs of 136 images each; TR=2 s)
 - ★ script = `s1.afni_proc_command` (in `AFNI_data2/` directory)
 - stimuli timing and GLT contrast files in `misc_files/`
 - ★ this script just runs program `afni_proc.py` that generates a shell script with all AFNI commands for analysis
 - Run by typing `tcsh s1.afni_proc_command`; then copy/paste `tcsh -x proc.ED.8.glt |& tee output.proc.ED.8.glt`
- Event-related study from Mike Beauchamp
 - ★ 10 runs with four classes of stimuli (short videos)
 - Tools moving (e.g., a hammer pounding) - **ToolMovie**
 - People moving (e.g., jumping jacks) - **HumanMovie**
 - Points outlining tools moving (no objects, just points) - **ToolPoint**
 - Points outlining people moving - **HumanPoint**
 - ★ Goal: find brain area that distinguishes natural motions (**HumanMovie** and **HumanPoint**) from simpler rigid motions (**ToolMovie** and **ToolPoint**)

Text output from programs goes to screen and file

Master Script for Data Analysis

afni_proc.py

```

-dsets ED/ED_r??+orig.HEAD
-subj_id ED.8.glt
-copy_anat ED/EDspgr
-tcat_remove_first_trs 2
-volreg_align_to first
-regress_stim_times misc_files/stim_times.*.1D
-regress_stim_labels ToolMovie HumanMovie
                    ToolPoint HumanPoint
-regress_basis 'TENT(0,14,8)'
-regress_opts_3dD
-gltsym ../misc_files/glt1.txt -glt_label 1 FullF
-gltsym ../misc_files/glt2.txt -glt_label 2 HvsT
-gltsym ../misc_files/glt3.txt -glt_label 3 MvsP
-gltsym ../misc_files/glt4.txt -glt_label 4 HMvsHP
-gltsym ../misc_files/glt5.txt -glt_label 5 TMvsTP
-gltsym ../misc_files/glt6.txt -glt_label 6 HPvsTP
-gltsym ../misc_files/glt7.txt -glt_label 7 HMvsTM

```

\<- Master script program
 \<- 10 input datasets
 \<- Set output filenames
 \<- Copy anat to output dir
 \<- Discard first 2 TRs
 \<- Where to align all EPIs
 \<- Stimulus timing files (4)
 \<- Stimulus labels
 \
 \<- HRF model
 \<- Specifies that next lines are options to be passed to **3dDeconvolve** directly (in this case, the GLTs we want computed)

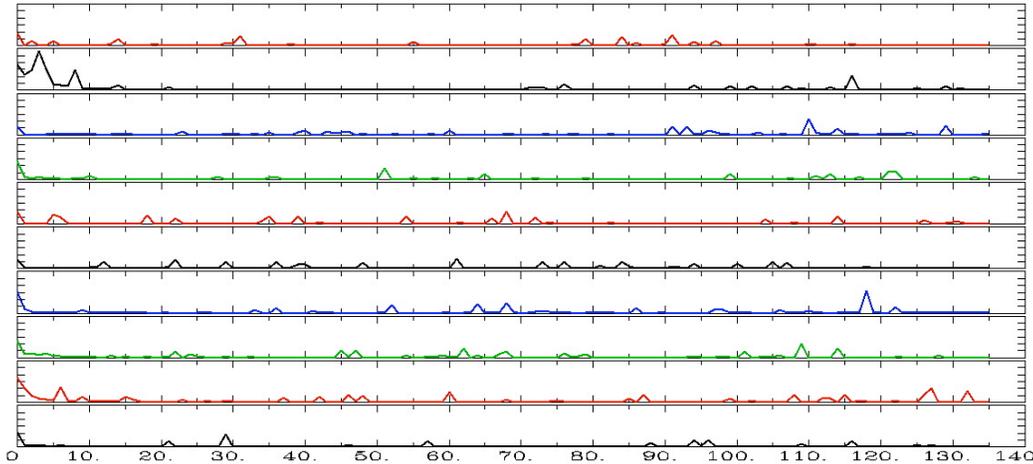
This script generates file **proc.ED.8.glt** (180 lines), which contains all the AFNI commands to produce analysis results into directory **ED.8.glt.results/** (148 files)

Script for Deconvolution - Outline

- Copy datasets into output directory for processing
- Examine each imaging run for outliers: **3dToutcount**
- Time shift each run's slices to a common origin: **3dTshift**
- Registration of each imaging run: **3dvolreg**
- Smooth each volume in space (136 sub-bricks per run): **3dmerge**
- Create a brain mask: **3dAutomask** and **3dcalc**
- Rescale each voxel time series in each imaging run so that its average through time is 100: **3dTstat** and **3dcalc**
 - ★ If baseline is 100, then a β_q of 5 (say) indicates a 5% signal change in that voxel at tent function knot # q after stimulus
 - ★ Biophysics: believe % signal change is relevant physiological parameter
- Catenate all imaging runs together into one big dataset (1360 time points): **3dTcat**
 - ★ This dataset is useful for plotting **-fitts** output from **3dDeconvolve** and visually examining time series fitting
- Compute HRFs and statistics: **3dDeconvolve**

Script - 3dToutcount

```
# set list of runs
set runs = (`count -digits 2 1 10`)
# run 3dToutcount for each run
foreach run ( $runs )
  3dToutcount -automask pb00.$subj.r$run.tcat+orig > outcount_r$run.1D
end
```



Via `1dplot outcount_r???.1D` ←
3dToutcount searches for “outliers” in data time series;
 You may want to examine noticeable runs & time points

Script - 3dTshift

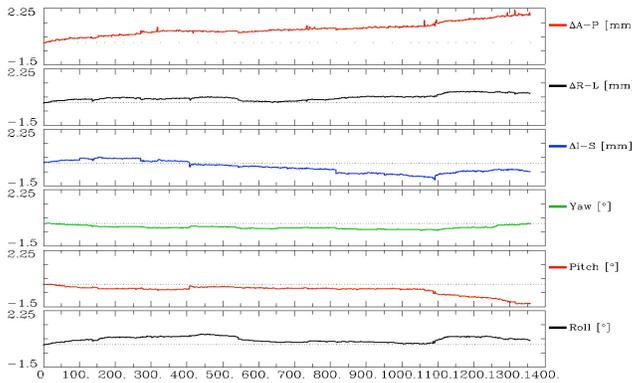
```
# run 3dTshift for each run
foreach run ( $runs )
  3dTshift -tzero 0 -quintic -prefix pb01.$subj.r$run.tshift \
    pb00.$subj.r$run.tcat+orig
end
```

- Produces new datasets where each time series has been shifted to have the same time origin
- `-tzero 0` means that all data time series are interpolated to match the time offset of the first slice
 - Which is what the slice timing files usually refer to
 - Quintic (5th order) polynomial interpolation is used
- **3dDeconvolve** will be run on time-shifted datasets
 - This is mostly important for Event-Related fMRI studies, where the response to the stimulus is briefer than for Block designs
 - (Because the stimulus is briefer)
 - Being a little off in the stimulus timing in a Block design is not likely to matter much

Script - 3dvolreg

```
# align each dset to the base volume
foreach run ( $runs )
  3dvolreg -verbose -zpad 1 -base pb01.$subj.r01.tshift+orig'[0]' \
    -1Dfile dfile.r$run.1D -prefix pb02.$subj.r$run.volreg \
    pb01.$subj.r$run.tshift+orig
end
```

- Produces new datasets where each volume (one time point) has been aligned (registered) to the #0 time point in the #1 dataset
- Movement parameters are saved into files `dfile.r$run.1D`
 - Will be used as extra regressors in `3dDeconvolve` to reduce motion artifacts



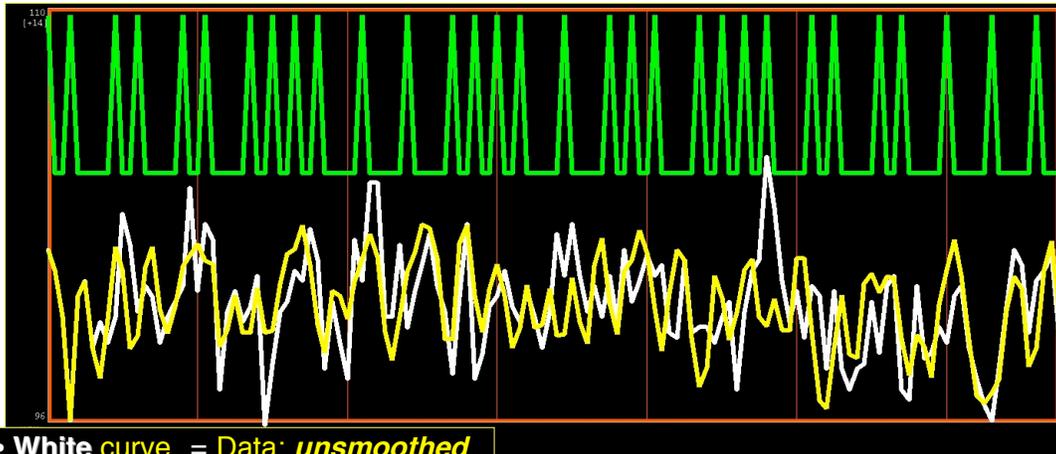
```
1dplot -volreg dfile.rall.1D
```

- Shows movement parameters for all runs (1360 time points) in degrees and millimeters
- Important to look at this graph!
- Excessive movement can make an imaging run useless — `3dvolreg` won't be able to compensate
 - Pay attention to scale of movements: more than about 2 voxel sizes in a short time interval is usually bad

Script - 3dmerge

```
# blur each volume
foreach run ( $runs )
  3dmerge -1blur_fwhm 4 -doall -prefix pb03.$subj.r$run.blur \
    pb02.$subj.r$run.volreg+orig
end
```

- **Why Blur?** Reduce noise by averaging neighboring voxels time series

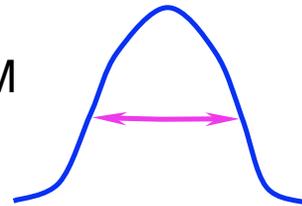
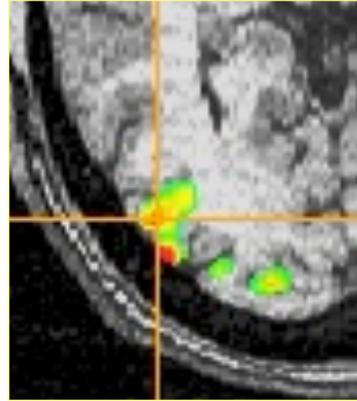


- **White** curve = Data: *unsmoothed*
- **Yellow** curve = Model fit ($R^2 = 0.50$)
- **Green** curve = Stimulus timing

This is an extremely good fit for ER FMRI data!

Why Blur? - 2

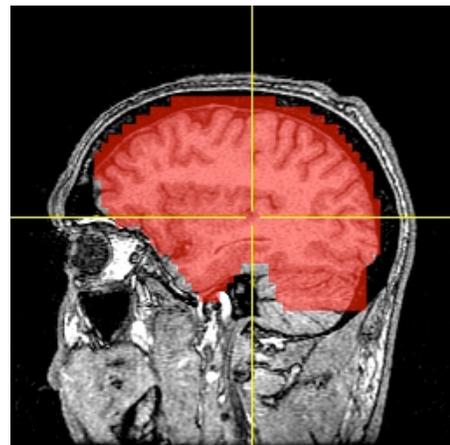
- fMRI activations are (usually) blob-ish (several voxels across)
- Averaging neighbors will also reduce the fiendish multiple comparisons problem
 - ★ Number of independent “resels” will be smaller than number of voxels (*e.g.*, 2000 vs. 20000)
- Why not just acquire at lower resolution?
 - ★ To avoid averaging across brain/non-brain interfaces
 - ★ To project onto surface models
- Amount to blur is specified as FWHM (Full Width at Half Maximum) of spatial averaging filter (4 mm in script)



Script - 3dAutomask

```
# create 'full_mask' dataset (union mask)
foreach run ( $runs )
  3dAutomask -dilate 1 -prefix rm.mask_r$run pb03.$subj.r$run.blur+orig
end
# get mean and compare it to 0 for taking 'union'
3dMean -datum short -prefix rm.mean rm.mask*.HEAD
3dcalc -a rm.mean+orig -expr 'ispositive(a-0)' -prefix full_mask.$subj
```

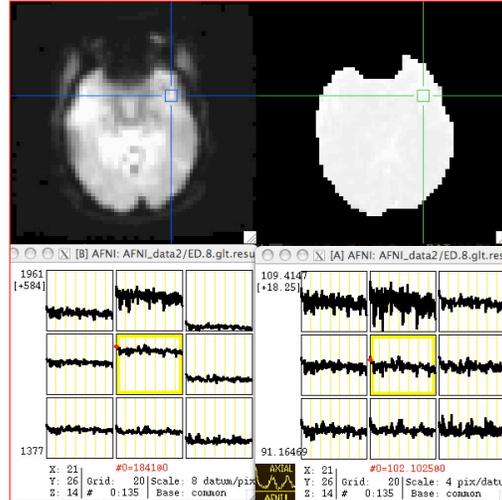
- **3dAutomask** creates a mask of contiguous high-intensity voxels (with some hole-filling) from each imaging run separately
- **3dMean** and **3dcalc** are used to create a mask that is the union of all the individual run masks
- **3dDeconvolve** analysis will be limited to voxels in this mask



Script - Scaling

```
# scale each voxel time series to have a mean of 100
# (subject to maximum value of 200)
foreach run ( $runs )
  3dTstat -prefix rm.mean_r$run pb03.$subj.r$run.blur+orig
  3dcalc -a pb03.$subj.r$run.blur+orig -b rm.mean_r$run+orig \
    -c full_mask.$subj+orig \
    -expr 'c * min(200, a/b*100)' -prefix pb04.$subj.r$run.scale
end
```

- **3dTstat** calculates the mean (through time) of each voxel's time series data
- For voxels in the mask, each data point is scaled (multiplied) using **3dcalc** so that it's time series will have mean=100
- If an HRF regressor has max amplitude = 1, then its β coefficient will represent the percent signal change (from the mean) due to that part of the signal model
- Scaled images are very boring
 - No spatial contrast by design!
 - Graphs have common baseline now



Script - 3dDeconvolve

```
3dDeconvolve -input pb04.$subj.r??.scale+orig.HEAD -polort 2 \
  -mask full_mask.$subj+orig -basis_normall 1 -num_stimts 10 \
  -stim_times 1 stimuli/stim_times.01.1D 'TENT(0,14,8)' \
  -stim_label 1 ToolMovie \
  -stim_times 2 stimuli/stim_times.02.1D 'TENT(0,14,8)' \
  -stim_label 2 HumanMovie \
  -stim_times 3 stimuli/stim_times.03.1D 'TENT(0,14,8)' \
  -stim_label 3 ToolPoint \
  -stim_times 4 stimuli/stim_times.04.1D 'TENT(0,14,8)' \
  -stim_label 4 HumanPoint \
  -stim_file 5 dfile.rall.1D'[0]' -stim_base 5 -stim_label 5 roll \
  -stim_file 6 dfile.rall.1D'[1]' -stim_base 6 -stim_label 6 pitch \
  -stim_file 7 dfile.rall.1D'[2]' -stim_base 7 -stim_label 7 yaw \
  -stim_file 8 dfile.rall.1D'[3]' -stim_base 8 -stim_label 8 dS \
  -stim_file 9 dfile.rall.1D'[4]' -stim_base 9 -stim_label 9 dL \
  -stim_file 10 dfile.rall.1D'[5]' -stim_base 10 -stim_label 10 dP \
  -iresp 1 iresp_ToolMovie.$subj -iresp 2 iresp_HumanMovie.$subj \
  -iresp 3 iresp_ToolPoint.$subj -iresp 4 iresp_HumanPoint.$subj \
  -gltsym ../misc_files/glt1.txt -glt_label 1 FullF \
  -gltsym ../misc_files/glt2.txt -glt_label 2 HvsT \
  -gltsym ../misc_files/glt3.txt -glt_label 3 MvsP \
  -gltsym ../misc_files/glt4.txt -glt_label 4 HMvsHP \
  -gltsym ../misc_files/glt5.txt -glt_label 5 TMvsTP \
  -gltsym ../misc_files/glt6.txt -glt_label 6 HPvsTP \
  -gltsym ../misc_files/glt7.txt -glt_label 7 HMvsTM \
  -fout -tout -full_first -x1D Xmat.1D -fitts fitts.$subj -bucket stats.$subj
```

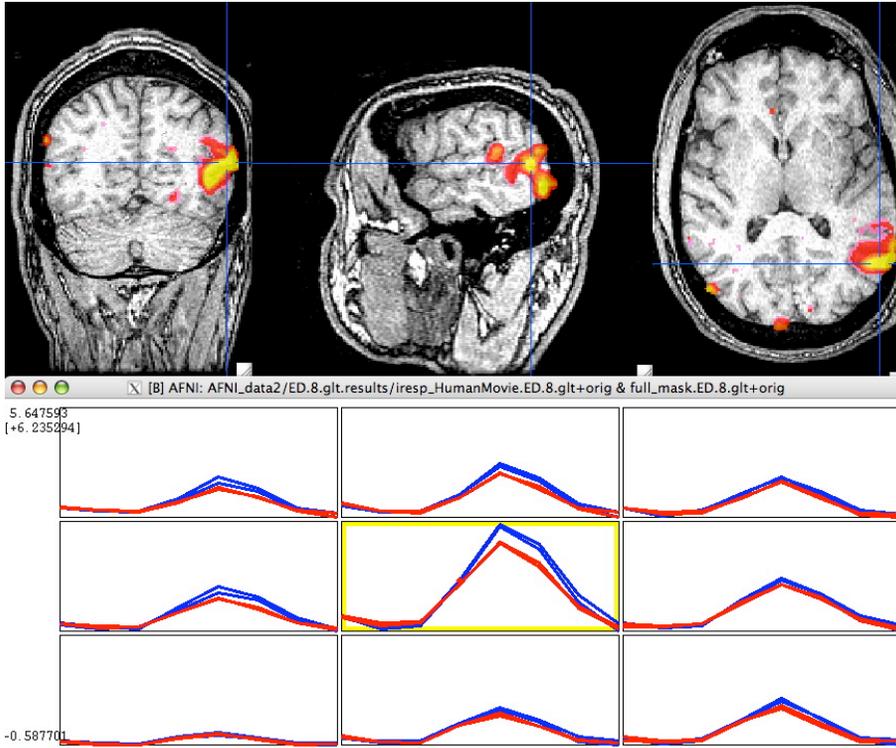
4 stim types

motion params

HRF outputs

GLTs

Results: Humans vs. Tools



- Color overlay: **HvsT** GLT contrast
- **Blue** (upper) graphs: Human HRFs
- **Red** (lower) graphs: Tool HRFs

Script - X Matrix



Via `lgrayplot -sep Xmat.1D`

Script - Random Comments

- `-polort 2`
 - ★ Sets baseline (detrending) to use quadratic polynomials—in each run
- `-mask full_mask.$subj+orig`
 - ★ Process only the voxels that are nonzero in this mask dataset
- `-basis_normall 1`
 - ★ Make sure that the basis functions used in the HRF expansion all have maximum magnitude=1
- `-stim_times 1 stimuli/stim_times.01.1D`
`'TENT(0,14,8)'`
 - `-stim_label 1 ToolMovie`
 - ★ The HRF model for the **ToolMovie** stimuli starts at 0 s after each stimulus, lasts for 14 s, and has 8 basis tent functions
 - Which have knots spaced $14/(8-1)=2$ s apart)
- `-iresp 1 iresp_ToolMovie.$subj`
 - ★ The HRF model for the **ToolMovie** stimuli is output into dataset `iresp_ToolMovie.ED.8.glt+orig`

Script - GLTs

- `-gltsym ../misc_files/glt2.txt -glt_label 2 HvsT`
 - ★ File `../misc_files/glt2.txt` contains 1 line of text:
 - `-ToolMovie +HumanMovie -ToolPoint +HumanPoint`
 - This is the “Humans vs. Tools” **HvsT** contrast shown on Results slide
- This GLT means to take all 8 β coefficients for each stimulus class and combine them as ordered:

$$LC = -\beta_0^{TM} - \dots - \beta_7^{TM} + \beta_0^{HM} + \dots + \beta_7^{HM} - \beta_0^{TP} - \dots - \beta_7^{TP} + \beta_0^{HP} + \dots + \beta_7^{HP}$$
 - This test is looking at the integrated (summed) response to the “Human” stimuli and subtracting it from the integrated response to the “Tool” stimuli
- Combining subsets of the β weights is also possible with `-gltsym`:
 - `+HumanMovie[2..6] -HumanPoint[2..6]`
 - This GLT would add up just the #2,3,4,5, & 6 β weights for one type of stimulus and subtract the sum of the #2,3,4,5, & 6 β weights for another type of stimulus
 - And also produce *F*- and *t*-statistics for this linear combination

Script - Multi-Row GLTs

- GLTs presented up to now have had one row
 - ★ Testing if some linear combination of β weights is nonzero; test statistic is t or F ($F=t^2$ when testing a single number)
 - ★ Testing if the X matrix columns, when added together to form one column as specified by the GLT (+ and -), explain a significant fraction of the data time series
- Can also do a single test to see if several different combinations of β weights are **all** zero

```
-gltsym ../misc_files/glt1.txt
-glt_label 1 FullF
```

4 rows

```
+ToolMovie
+HumanMovie
+ToolPoint
+HumanPoint
```

- ★ Tests if **any** of the stimulus classes have nonzero integrated HRF (each name means “add up those β weights”) : $DOF= (4,1292)$
- ★ Different than the default “Full F-stat” produced by **3dDeconvolve**, which tests if any of the **individual β** weights are nonzero: $DOF= (32,1292)$

Other Fun 3dDeconvolve Options

- **-input1D** = used to process a single time series, rather than a dataset full of time series
 - ★ e.g., test out a stimulus timing sequence on sample data
 - ★ **-nodata** option can be used to check for collinearity
- **-censor** = used to turn off processing for some time points
 - ★ for time points that are “bad” (e.g., too much movement)
- **-sresp** = output standard deviation of HRF estimates
 - ★ can then plot error bands around HRF in AFNI graph viewer
- **-errts** = output residuals (difference between fitted model and data)
 - ★ for statistical analysis of time series noise
- **-TR_times dt** = output **-iresp** and **-sresp** HRF results with time step **dt** (instead of input dataset TR)
 - ★ Can be used to make HRF graphs look better
- **-jobs N** = run with independent threads — **N** of them
 - ★ extra speed, if you have a dual-CPU system (or more)!

Two Possible Formats for `-stim_times`

- A single column of numbers
 - ★ One stimulus time per row
 - ★ Times are relative to first image in dataset being at $t=0$
 - ★ May not be simplest to use if multiple runs are catenated
- One row for each run within a catenated dataset
 - ★ Each time in j^{th} row is relative to start of run $\#j$ being $t=0$
 - ★ If some run has NO stimuli in the given class, just put a single “*” in that row as a filler
 - Different numbers of stim per run are OK
 - At least one row must have more than 1 time (so that this type of timing file can be told from the other)
- Two methods are available because of users’ diverse needs
 - ★ **N.B.:** if you chop first few images off the start of each run, the inputs to `-stim_times` must be adjusted accordingly
 - Can be done with program `1deval`

```
4.7
9.6
11.8
19.4
```

```
4.7 9.6 11.8 19.4
*
8.3 10.6
```

Technical Points about 3dDeconvolve

- See <http://afni.nimh.nih.gov/doc/misc/3dDeconvolveSummer2004/>
- Equation solver: Gaussian elimination to compute **R** matrix pseudo-inverse uses SVD (like principal components), not Gaussian elimination
 - ★ Advantage: smaller sensitivity to computational errors
 - ★ “Condition number” and “inverse error” values are printed at program startup, as measures of accuracy of pseudo-inverse
 - ★ Condition number < 1000 is good
 - ★ Inverse error < 1.0e-10 is good
- `3dDeconvolve_f` program can be used to compute in single precision (7 decimal places) rather than double precision (16)
 - ★ For better speed, but with lower numerical accuracy
 - ★ Best to do at least one run **both** ways to check if results differ significantly (SVD solver *should* be safe, but caution is good)

Technical Points - 2

- `-polort` option uses Legendre polynomials instead of simple $1, t, t^2, t^3, \dots$ basis functions (more numerical accuracy)
 - ★ Usually you want to use the `-nobout` option to suppress output of the baseline parameters and their statistics
 - ★ If you **do** want to keep the baseline parameters and use them, you need to know the details of how the Legendre polynomials are actually used in `3dDeconvolve`
- `-stim_times` has some other basis function options for the HRF model besides `BLOCK` and `TENT`
 - ★ Most of these are experimental, and should only be used by someone who understands
- `-stim_file` is used when you want to input a column into the **X** matrix directly
 - ★ Motion parameters (as in previous examples)
 - ★ If you create a stimulus+response model outside `3dDeconvolve` (e.g., using program `waver`)

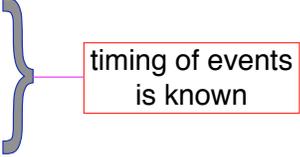
Technical Points - 3

- `3dDeconvolve` also checks for duplicate `-stim_file` and `-stim_times` names and for duplicate **X** matrix columns, and prints warnings
 - ★ These are not fatal errors
 - If the same regressor is given twice, each copy will only get half the amplitude (" β weight") in the solution
- All-zero regressors are allowed
 - ★ Will get zero weight in the solution
 - A warning message will be printed to the terminal
 - ★ Example: task where subject makes a choice for each stimulus (e.g., male or female face?)
 - You want to analyze correct and incorrect trials a separate cases
 - What if a subject makes no mistakes? Hmm...

Upgrades – *Planned* or *Dreamed of*

- “Area under curve” addition to `-gltsym` to allow testing of pieces of HRF models from `-stim_times`
- Slice- and/or voxel-dependent regressors
 - ★ For physiological noise cancellation, etc.
- Floating point output format
 - ★ Currently is shorts + scale factor (can lose precision)
- Estimate temporal correlation structure of residual time series, then use that information to re-do the regression analysis (to improve/correct the statistics)
 - ★ Could do the same with spatial correlation?

Advanced Topics in Regression

- Can have activations with multiple phases that are not always in the same time relationship to each other; e.g.:
 - a) subject gets cue #1
 - b) variable waiting time (“hold”)
 - c) subject gets cue #2, emits response
 - which depends on both cue #1 and #2
- ★ Cannot treat this as one event with one HRF, since the different waiting times will result in different overlaps in separate responses from cue #1 and cue #2
- ★ Solution is multiple HRFs: separate HRF (fixed shape or deconvolution) for cue #1 times and for cue #2 times
 - Must have significant variability in inter-cue waiting times, or will get a nearly-collinear model
 - impossible to tell tail end of HRF #1 from the start of HRF #2, if always locked together in same temporal relationship
 - How much variability is “significant”? Good question.

Even More Complicated Case

- Solving a visually presented puzzle:
 - a) subject sees puzzle
 - b) subject cogitates a while
 - c) subject responds with solution
- } timing of events is measured
- The problem is that we expect some voxels to be significant in phase (b) as well as phases (a) and/or (c)
 - Variable length of phase (b) means that shape for its response varies between trials
 - ★ Which is contrary to the whole idea of averaging trials together to get decent statistics (which is basically what linear regression amounts to, in an elaborate sort of way)
 - Could assume response **amplitude** in phase (b) is constant across trials, and response **duration** varies directly with time between phases (a) and (c)
 - ★ Need three HRFs
 - ★ Difficult to generate (b) HRF in **3dDeconvolve** (use **waver**?)

Regression with ABI - 1

- **ABI = Auxiliary Behavioral Information**
 - ★ Extra data measured about each response to a stimulus
 - ★ Reaction time; Galvanic skin response; Pain level perception; Emotional valence (happy or sad or angry face?)
- Want to include this information in data analysis, to see if some brain activations vary in amplitude with the ABI
- Discrete levels (2 or maybe 3) of ABI:
 - ★ Separate the stimuli into sub-classes that are determined by the ABI (“on” and “off”, maybe?)
 - ★ Use a GLT to test if there is a difference between the fMRI responses in the sub-classes

```
3dDeconvolve ... \
-stim_times 1 regressor_on.1D 'BLOCK(2,1)' -stim_label 1 'On' \
-stim_times 2 regressor_off.1D 'BLOCK(2,1)' -stim_label 2 'Off' \
-gltsym 'SYM: +On | +Off' -glt_label 1 'On+Off' \
-gltsym 'SYM: +On -Off' -glt_label 2 'On-Off' ...
```

- “On+Off” tests for any activation in either the “on” or “off” conditions
- “On-Off” tests for differences in activation between “on” and “off” conditions
- Can use 3dcalc to threshold on **both** statistics at once to find a **conjunction**

Regression with ABI - 2

- Continuous ABI levels
 - ★ Want to find active voxels whose activation level also depends on the ABI measurement
- Need to make 2 separate regressors
 - ★ One to find the mean FMRI response (the usual `-stim_times` input)
 - ★ One to find the variations in the FMRI response as the ABI data varies
- The second regressor should have the form

$$r(t) = \sum_{k=1}^K h(t - \tau_k) \cdot (a_k - \bar{a})$$

- ★ Where a_k = amplitude of k^{th} ABI value, and \bar{a} is the average ABI amplitude
- Response (β) for first regressor is standard activation map
- β for second regressor is activation map of places whose BOLD response is proportional to fluctuations in ABI data
 - ★ Unfortunately, at this time, `3dDeconvolve` does not contain any way to make the second regressor (maybe in the future?)
 - ★ Can be done with the `waver` program

Noise Issues

- “Noise” in FMRI is caused by several factors, not completely characterized
 - ★ MR thermal noise (well understood, unremovable)
 - ★ Cardiac and respiratory cycles (partly understood)
 - In principle, could measure these sources of noise separately and then try to regress them out
 - ➔ RETROICOR program underway (R Birn & M Smith of FIM/NIMH)
 - ★ Scanner fluctuations (e.g., thermal drift of hardware)
 - ★ Small subject head movements (10-100 mm)
 - ★ Very low frequency fluctuations (periods longer than 100 s)
- Data analysis should try to remove what can be removed and allow for the statistical effects of what can't be removed
 - ★ “Serial correlation” in the noise time series affects the t - and F -statistics calculated by `3dDeconvolve`
 - ★ At present, nothing is done to correct for this effect (by us)

Nonlinear Regression

- Linear models aren't the only possibility
 - ★ e.g., could try to fit HRF of the form $h(t) = a \cdot t^b \cdot e^{-t/c}$
 - ★ Unknowns b and c appear nonlinearly in this formula
- Program **3dNLFit** can do nonlinear regression (including nonlinear deconvolution)
 - ★ User must provide a C function that computes the model time series, given a set of parameters (e.g., a, b, c)
 - We could help you develop this C model function
 - Several sample model functions in the AFNI source code distribution
 - ★ Program then drives this C function repeatedly, searching for the set of parameters that best fit each voxel
 - ★ Has been used to fit pharmacological wash-in/wash-out models (difference of two exponentials) to fMRI data acquired during pharmacological challenges
 - e.g., injection of nicotine, cocaine, ethanol, etc.
 - these are difficult experiments to do **and** to analyze