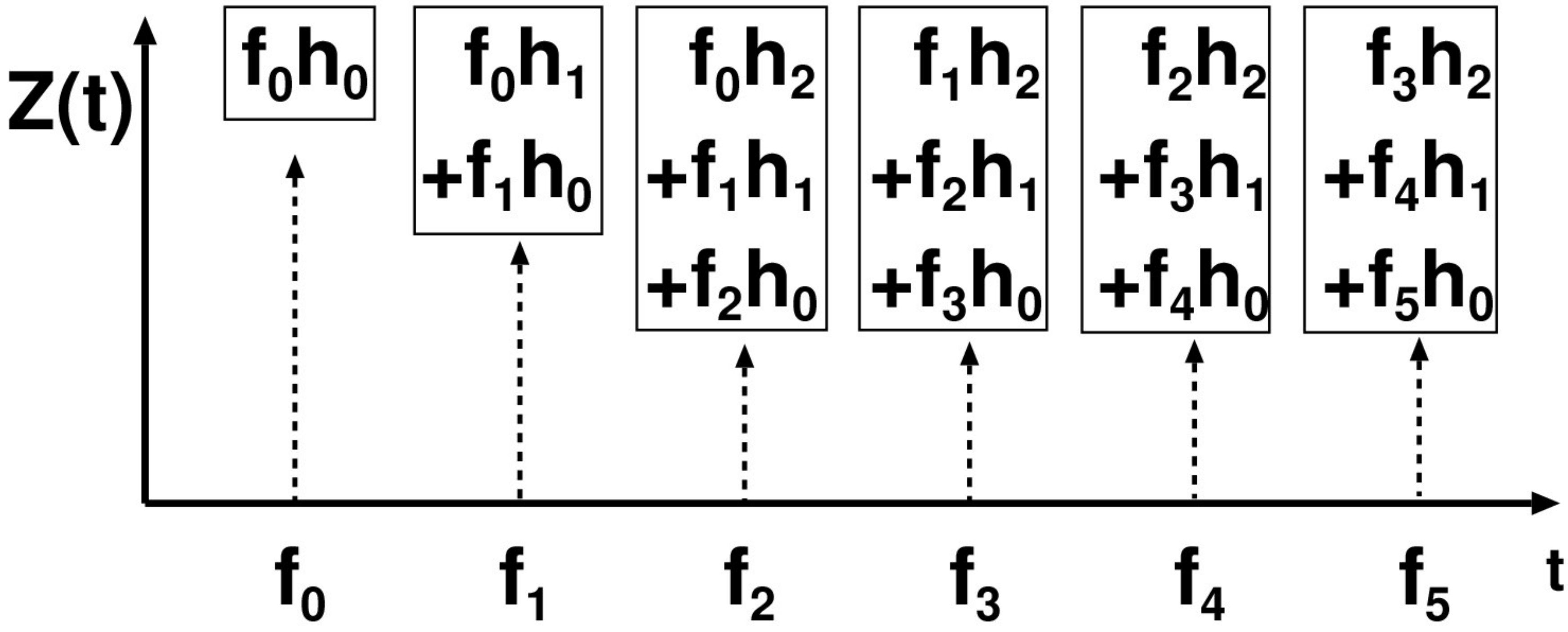
Data Analysis: Deconvolution Models

- Convolution signal model:

$$Z(t) = \underbrace{\beta_0 + \beta_1 \cdot t}_{\text{baseline model}} + \alpha \cdot \sum_{m=0}^{p-1} \underbrace{f(t - m\Delta t)}_{\substack{\text{Stimulus} \\ \text{function} \\ \text{(0 or 1?)}}} \cdot \underbrace{h(m\Delta t)}_{\substack{\text{hemodynamic} \\ \text{response} \\ \text{function}}} + \underbrace{\varepsilon(t)}_{\text{noise}}$$

- Deconvolution is computing $f(t)$ and/or $h(t)$ from data $Z(t)$
  - ◇ Most common use in FMRI is computing each voxel's HRF $h(t)$, assuming we know the (common) input function $f(t)$
  - ↪ Then compute various statistics about estimated $h(t)$'s:
    - ▷ Is it significantly different from zero (activation)?
    - ▷ Is the early part or the late part bigger?
  - ◇ Can also assume $h(t)$ and try to find $f(t)$
  - ↪ Might be useful with complex continuous stimuli (e.g., a video), to see which parts of the stimulus elicited a significantly increased activation in what parts of the brain
  - ◇ Can also try to find both $f(t)$ and $h(t)$ simultaneously: "blind deconvolution"
  - ↪ Must put some constraints on $f(t)$, $h(t)$ to get anywhere with this

- How deconvolution solves for $h(t)$, given data $Z(t)$ and stimulus $f(t)$
  - ◇ Assemble equations for each Z_n data value [here, assume max lag is 2]



$$
\begin{aligned}
Z_0 &= 1 \cdot \beta_0 + 0 \cdot \beta_1 + f_0 \cdot h_0 \\
Z_1 &= 1 \cdot \beta_0 + 1 \cdot \beta_1 + f_1 \cdot h_0 + f_0 \cdot h_1 \\
Z_2 &= 1 \cdot \beta_0 + 2 \cdot \beta_1 + f_2 \cdot h_0 + f_1 \cdot h_1 + f_0 \cdot h_2 \\
Z_3 &= 1 \cdot \beta_0 + 3 \cdot \beta_1 + f_3 \cdot h_0 + f_2 \cdot h_1 + f_1 \cdot h_2
\end{aligned}
$$

  - ◇ Solve linear equations for unknowns $\{\beta_0, \beta_1, h_0, h_1, h_2\}$
  - ◇ Then compute various statistics about these estimates

- Variations and generalizations of the above model:
  - ◇ Stimulus does not occur on the Δt time grid:

$$Z(t) = \beta_0 + \beta_1 \cdot t + \sum_{s=1}^{N_s} h(t - \tau_s) + \varepsilon(t)$$

    where the s^{th} stimulus occurs at time τ_s, for $s = 1, 2, \ldots, N_s$

    $\hookrightarrow$ Have replaced $f(t)$ with known stimulus times

    $\hookrightarrow$ Goal is to find $h(t)$

    $\hookrightarrow$ Question for the astute: what happened to α?

  - ◇ Stimulus has two (or more) phases, which may occur at different times (e.g., presentation and response phases):

$$Z(t) = \beta_0 + \beta_1 \cdot t + \sum_{s=1}^{N_s} \left[h_1(t - \tau_s) + h_2(t - (\tau_s + \delta_s) \right] + \varepsilon(t)$$

    where the first phase of the s^{th} stimulus occurs at time τ_s and the second phase at time δ_s later

    $\hookrightarrow$ Goal is to find $h_1(t)$ and $h_2(t)$ separately

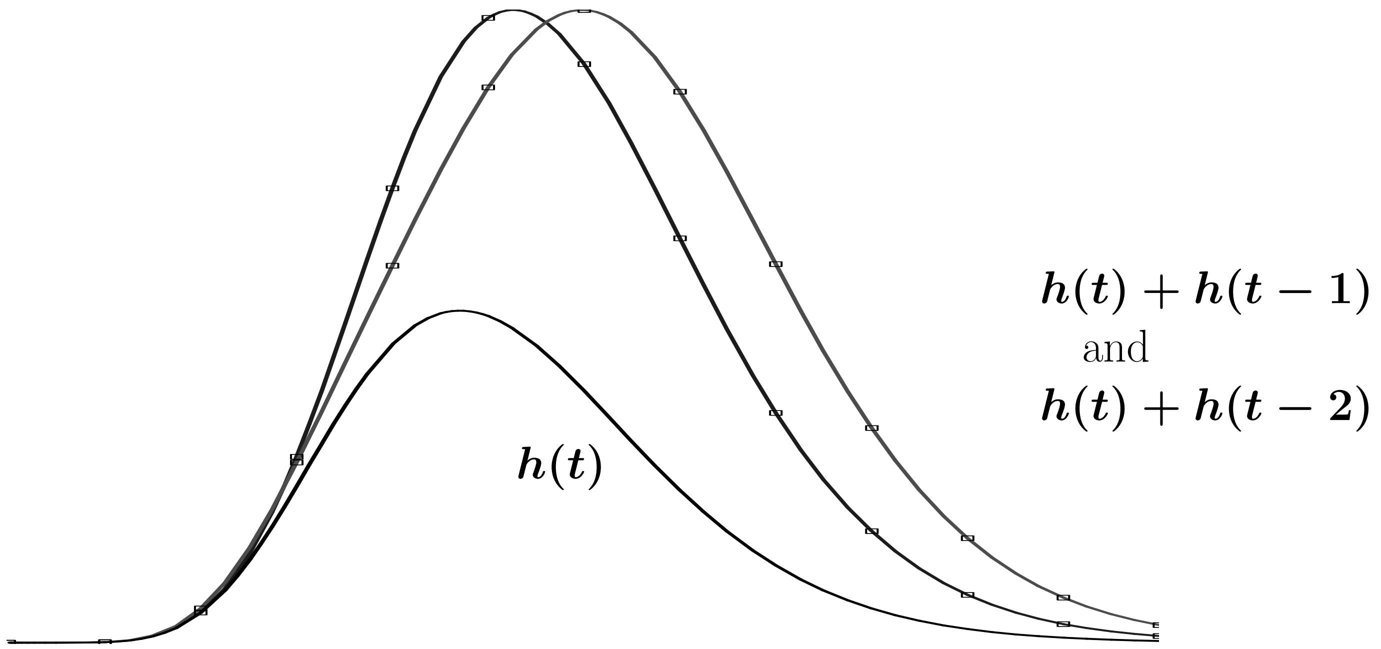
    $\hookrightarrow$ Delay time δ_s must vary ("jitter") to make this feasible

      - ▷ Otherwise, a single HRF $h(t) = h_1(t) + h_2(t)$ is indistinguishable from this model

◇ There are two (or more) types of stimuli:

$$Z(t) = \beta_0 + \beta_1 \cdot t + \sum_{s=1}^{N_s} h_1(t - \tau_s) + \sum_{q=1}^{N_q} h_2(t - \mu_q) + \varepsilon(t)$$

where there are N_s stimuli of the first class (at times $\tau_1, \tau_2, \ldots$) and N_q stimuli of the second class (at times $\mu_1, \mu_2, \ldots$)



$\hookrightarrow$ Problem is to get enough data to distinguish between $h(t) + h(t-1)$ and $h(t) + h(t-2)$, for example

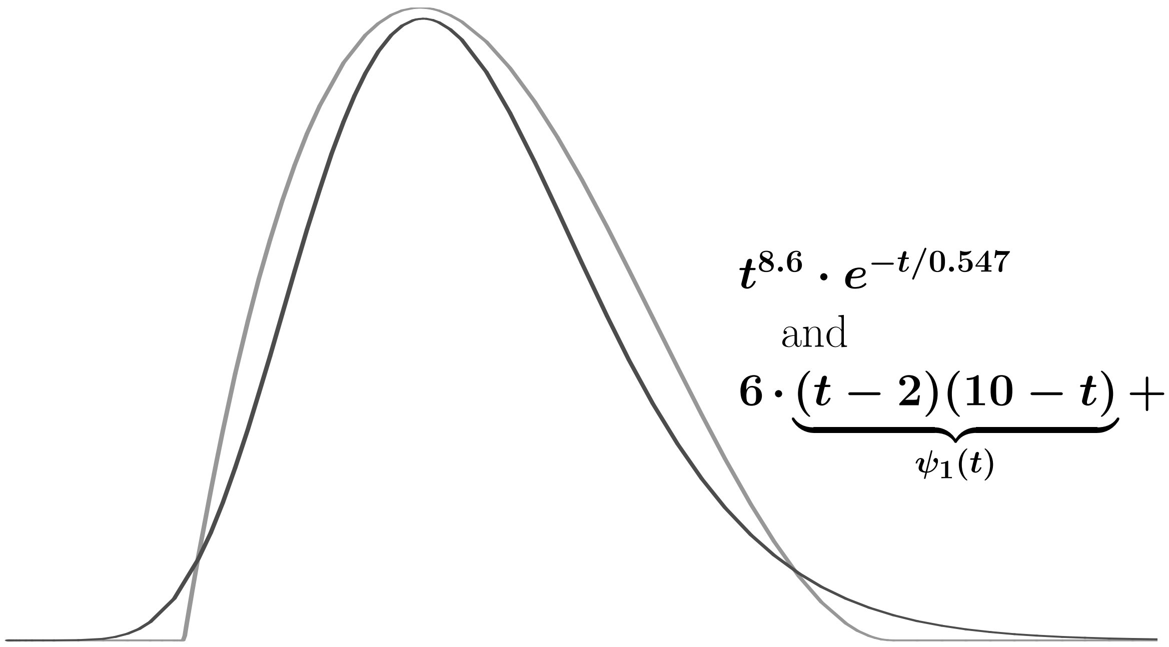
◇ Constraints on HRF functions:

$\hookrightarrow$ Can either try to find $\boldsymbol{h(t)}$ in each voxel separately, or try to find a common HRF that works everywhere (e.g., component analyses)

$\hookrightarrow$ Can let $\boldsymbol{h(t)}$ be arbitrary function, or can limit it to make HRF be more "reasonable" and/or more "manageable"

▷ Linear constraint: $\boldsymbol{h(t) = \sum_{a=0}^{N_a} \lambda_a \cdot \psi_a(t)}$

where each $\boldsymbol{\psi_a(t)}$ is a fixed "basis" function (which constrains shape of $\boldsymbol{h(t)}$) and the unknown amplitudes $\boldsymbol{\lambda_a}$ are to be determined from data



$\boldsymbol{t^{8.6} \cdot e^{-t/0.547}}$

and

$$\boldsymbol{6 \cdot \underbrace{(t-2)(10-t)}_{\psi_1(t)} + 1.5 \cdot \underbrace{(t-2)(10-t)(6-t)}_{\psi_2(t)}}$$

These two functions would be hard to tell apart without a lot of data!

▷ Nonlinear constraint:

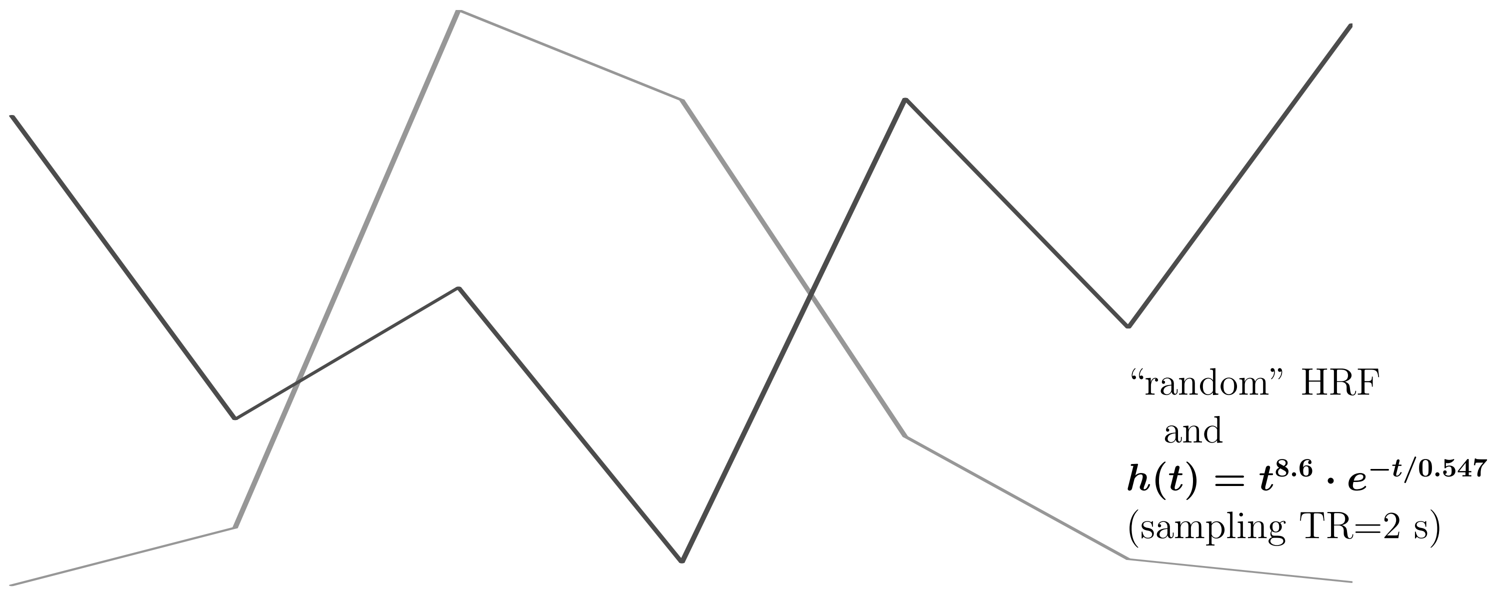
$$h(t) = \begin{cases} 0 & t \leq t_0 \\ A \cdot (t - t_0)^r \cdot e^{-(t-t_0)/b} & t > t_0 \end{cases}$$

where the unknowns are A (amplitude), t_0 (time delay), r (rise exponent), and b (decay time)

- "Gamma variate" or "gamma density" model (<u>not</u> "gamma function")
- Peak response is at $t = t_0 + b \cdot r$; FWHM $\approx 2.4 \cdot b^{1/2} \cdot r$

▷ "Reasonability" from constraints:

- If TR=2 s and stimulus lasts 3 s $\Rightarrow$ 6–7 time points for $h(t)$
- Arbitrary $h(t)$ at these points could give something weird:



- Constraints can suppress "unreasonable" responses
- But such responses may be symptoms of problems you need to find

- AFNI programs for deconvolution analysis:
  - ◇ `3dDeconvolve` will perform linear least squares to fit time series models to each voxel separately
    - ↪ Models can have fixed $h(t)$ (one lag), or can have multiple lags to perform deconvolution
    - ↪ Can also use to analyze a single time series, as a test
  - ◇ `3dNLfim` will perform nonlinear least squares to fit time series models to each voxel separately
    - ↪ Requires writing a C function to evaluate nonlinear model
    - ↪ Example model for gamma variate fitting is in AFNI distribution
    - ↪ This program is very slow
  - ◇ Both are command line programs: they read in datasets, compute for a while, and write out new datasets, which you then load back into AFNI for display/exploration
  - ◇ Both programs also have an interactive plugin which can be used to fit data in AFNI graph viewer
    - ↪ Useful for playing with model and determining if it useful/complete
    - ↪ Can be quite fun to overlay fitted responses on data graphs!
  - ◇ `3dConvolve` is a program for generating 3D+time datasets from a convolution model

Using 3dDeconvolve

- Written and maintained by Doug Ward of the Biophysics Research Institute, Medical College of Wisconsin, Milwaukee
- Master documentation: `3dDeconvolve.ps` or `3dDeconvolve.pdf`, available at AFNI Web site documentation pages:

  PDF format ⇒ `http://afni.nimh.nih.gov/afni/docpdf/`
  PostScript ⇒ `http://afni.nimh.nih.gov/afni/docps/`

  - ◇ Refer to this manual for more math, all input options, and many examples
  - ◇ 3dConvolve and the deconvolution plugin are also documented therein
- Ostensibly, 3dDeconvolve is a "command line" program, but in practice, there are so many inputs on the command line that you actually have to put the command into a script file, then execute the file
  - ◇ This also gives you a record of what you did, so you can do it again
  - ◇ To execute a command (or list of commands) in a file: `source scriptfilename`
  - ◇ A command "line" is a single logical line, but can be split across many physical lines in the script file:
    - ↪ This is done by putting a backslash "\" at the end of each physical line but the last one

- ↪ Don't use the forward slash "/" for this!
- ↪ Don't put any blanks or other characters after the "\", or the logical command line will end right there (which is bad)
- ↪ Example (with the options and "/" characters made to line up):

```
3dDeconvolve -input fred+orig      \
             -num_stimts 1         \
             -stim_file 1 elvis.1D \
             -stim_label 1 Elvis   \
             -stim_minlag 1 1      \
             -stim_maxlag 1 5      \
             -bucket Ethel         \
             -fout -tout           \
             -fitts fredfit
```

- ↪ In this format, it is relatively easy to read and edit the script file
- ↪ Recommended text editor for "newbies" to Unix: `nedit`

- Setting up 3dDeconvolve for deconvolution analysis:
  - ◇ Simplest case:
    - ↪ Stimulus events take place on Δt grid
    - ↪ Will allow arbitrary HRF to stimulus in each voxel
  - ◇ There are 3 types of input files:
    - ↪ AFNI formatted 3D+time datasets
    - ↪ 1D files, representing time series on the Δt grid
      - ▷ Stored as a single ASCII number per line
    - ↪ Matrix files, used to control generation of analysis results
      - ▷ Stored as a 2D layout of ASCII numbers in a text file
    - ↪ Examples: a time series of length 5, and a 2×6 matrix

```
1
0
1
0
1
```

```
0  -1  -1  1   0   0
0  -1   0  0  -1  -1
```

  - ◇ User must divide stimulus events into classes
    - ↪ Need a 0/1 time series series file for each class, indicating when the stimuli for that class occur
    - ↪ Each class k will get its own HRF $h_k(t)$, for $k = 1, 2, \ldots$

- Important command line options for 3dDeconvolve:
  - ◇ Format of the descriptions below:
    `-option` *arguments*
    - ↪ The string "`-option`" specifies the option, and must be typed as shown
    - ↪ If an option has arguments (most of them do), their names are given in italics following the option name
    - ↪ When you actually use an option, the arguments will be replaced with file-names, numbers, etc., as appropriate
  - ◇ `-input` *fname*
    fname specifies the input 3D+time AFNI dataset (e.g., `fred+orig`)
  - ◇ `-num_stimts` *num*
    This option specifies how many classes of stimuli are present; it is required. There is no built-in upper limit on *num*.
  - ◇ `-stim_file` *k sname*
    This option specifies the input time series for the $\boldsymbol{k}^{\text{th}}$ stimulus class
    - ↪ *k* should be from 1 to *num* (from `-num_stimts`)
    - ↪ *sname* is the name of the file to be read in
    - ↪ For event-related analyses, *sname* would usually be a time series consisting of 0s and 1s
    - ↪ This input corresponds to the function $\boldsymbol{f_k(t)}$

◇ `-stim_label` *k slabel*

This option specifies the label that will be attached to the output that is relevant to the $\boldsymbol{k}^{\text{th}}$ stimulus file

↪ Makes it easier to interpret the output file

↪ *slabel* should be enclosed in `'quotes'` if it contains "special" characters such as: blank, `*[]{};`

◇ `-stim_minlag` *k m*

This option specifies that the minimum lag to be used for the $\boldsymbol{k}^{\text{th}}$ stimulus file is the number *m*

↪ If this option is not present, then $\boldsymbol{m = 0}$

◇ `-stim_maxlag` *k n*

This option specifies that the maximum lag to be used for the $\boldsymbol{k}^{\text{th}}$ stimulus file is the number *n* ($\boldsymbol{n \geq m}$ is required)

↪ If this option is not present, then $\boldsymbol{n = 0}$

↪ The response to the $\boldsymbol{k}^{\text{th}}$ stimulus is $\boldsymbol{r_k(t) = \sum_{q=m}^{n} f_k(t - q\Delta t) \cdot h_k(q\Delta t)}$

One goal of the program is to compute the set $\boldsymbol{\{h_k(q\Delta t) : q = m \ldots n\}}$

↪ The default case $\boldsymbol{m = n = 0}$ corresponds to simple linear regression

▷ Then $\boldsymbol{h_k(0)}$ is the amplitude of $\boldsymbol{f_k(t)}$ in the data

◇ `-iresp` *k iprefix*

This option specifies that the $\boldsymbol{k}^{\text{th}}$ HRF function $\boldsymbol{h_k(t)}$ is to be saved in an AFNI dataset with prefix name given by the string *iprefix*

↪ This dataset is useful if you want to graph the HRF results

◇ `-sresp` *k sprefix*

This option specifies that the standard deviation of the $\boldsymbol{k}^{\text{th}}$ HRF function $\boldsymbol{h_k(t)}$ should be saved in an AFNI dataset with prefix name given by the string *sprefix*

↪ This dataset lets you visually inspect the confidence you should have in $\boldsymbol{h_k(t)}$

◇ `-fitts` *fprefix*

This option specifies that the fitted model should be written to an AFNI 3D+time dataset with prefix name given by the string *fprefix*

↪ Using the `Dataset#2` plugin and 1D Transform, and the `Double Plot` graphing option, you can use this to overlay the fitted time series model on each voxel's actual data

↪ Another way to make this type of graph is with the Deconvolution plugin

◇ `-bucket` *bprefix*

This option specifies that the statistical output should be written to an AFNI "bucket" dataset with prefix name *bprefix* — you almost surely want to use this option!

↪ The bucket output contains multiple sub-bricks, with various statistical parameters; it provides a convenient way to gather all the diverse possible outputs into one place

↪ The sub-bricks are labeled via `-stim_label`, and can be used within AFNI as a statistical threshold and/or to generate colored overlays

↪ Additional options are needed to specify which statistics go into this dataset:

- ▷ `-fout` specifies that the $\boldsymbol{F}$-statistics for the full model (with all stimulus functions) and for each individual partial model (with one stimulus function at a time) be included in the bucket dataset
  - Full $\boldsymbol{F}$ measures significance of overall model
  - Partial $\boldsymbol{F}$ measures significance of each component of model
- ▷ `-rout` specifies that the $\boldsymbol{R}^2$-statistics for the full and partial models be included in the bucket dataset (these are generalizations of the correlation coefficient, and are equivalent to the $\boldsymbol{F}$-statistics <u>if</u> the Gaussian white noise model is correct)
- ▷ `-tout` specifies that the $\boldsymbol{t}$-statistics for each regression parameter ($\boldsymbol{h_k(q\Delta t)}$ for all $\boldsymbol{k}$ and $\boldsymbol{q}$) be saved into bucket dataset sub-bricks

◇ General Linear Tests (GLTs):

↪ These are used to perform tests on linear combinations of regression parameters ($h_k(q\Delta t)$ for all k and q, plus the baseline parameters)

↪ The resulting F-statistics are added to the output bucket dataset

↪ To specify a test, you input a matrix that gives the coefficient of the linear combinations you want to test against zero

↪ In most cases, this matrix will have only 0, 1, and -1 as entries (0=ignore, 1=add, -1=subtract)

↪ To specify the test, you must know the order of the regression parameters in the output

- ▷ Baseline parameters come first (usually, 2 of them: β_0, β_1)
- ▷ $h_1(q\Delta t)$ for $q = m_1 \ldots n_1$ comes next
- ▷ $h_2(q\Delta t)$ for $q = m_2 \ldots n_2$ comes next, etc.
- ▷ Example: 2 stimulus classes, 4 lags each ⇒ parameter vector is

$$\{\ \beta_0\ \beta_1\ h_1(0)\ h_1(\Delta t)\ h_1(2\Delta t)\ h_1(3\Delta t)\ h_2(0)\ h_2(\Delta t)\ h_2(2\Delta t)\ h_2(3\Delta t)\ \}$$

- ▷ To test if $h_1(\Delta t)$ is different from $h_2(\Delta t)$ (i.e., if $h_1(\Delta t) - h_2(\Delta t) \neq 0$), the matrix is

  $[\,0\ 0\ 0\ 1\ 0\ 0\ 0\ \ -1\ 0\ 0\,]$

- ▷ `-glt` *s gltname*

  Indicates to do a GLT with s rows, reading the matrix from file *gltname*

  • Example above: $s = 1$; matrix file contains `0 0 0 1 0 0 0 -1 0 0`

- To test if $h_1(t) = h_2(t)$ for all t computed, we need four input lines:

```
0 0 1 0 0 0 -1  0  0  0
0 0 0 1 0 0  0 -1  0  0
0 0 0 0 1 0  0  0 -1  0
0 0 0 0 0 1  0  0  0 -1
```

- The result from this is an F-statistic
- ANOVA type analyses can be carried out with `-glt`
- `-glt_label` *k glabel*

  This option attaches the label string *glabel* to the output for the k^{th} GLT (in order on the command "line")

◇ Other things you can do:

↪ `-censor` *cname*

cname is a 1D time series file specifying which points to keep (input=1) and which to delete (input=0) from the analysis (default=keep all points)

↪ `-concat` *rname*

3dDeconvolve can deal with 3D+time input datasets that are catenated from multiple imaging runs (via program 3dTcat)

- ▷ To deal properly with the discontinuity across runs, you must specify the starting point in the input dataset for each imaging run
- ▷ *rname* is the name of a 1D time series file whose $\boldsymbol{j}^{\text{th}}$ entry is the time index for the start of the $\boldsymbol{j}^{\text{th}}$ run within the input dataset
- ▷ Note that each run will get a separate $\boldsymbol{\beta}_0$ and $\boldsymbol{\beta}_1$, which must be allowed for when setting up `-glt` matrices

↪ `-mask` *mname*

mname is the name of a 3D dataset that can be used to mask off unwanted regions from analysis; voxels where the mask dataset is 0 will not be analyzed by 3dDeconvolve

- ▷ A mask dataset might be created using program 3dClipLevel

↪ `-polort` *pnum*

pnum sets the polynomial order of the baseline model; the default is 1; useful values would be from 0 to 3