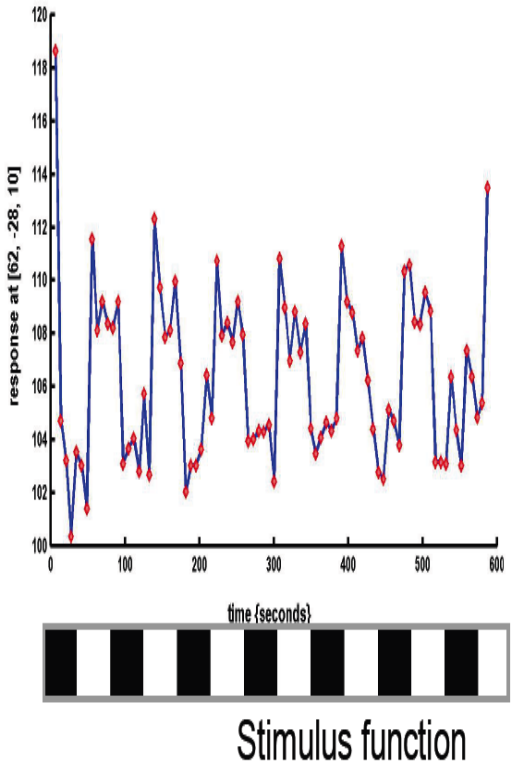
General Linear Model (GLM)

Lecture Notes

Ahmet Ademoglu, *PhD*
Bogazici University
Institute of Biomedical Engineering

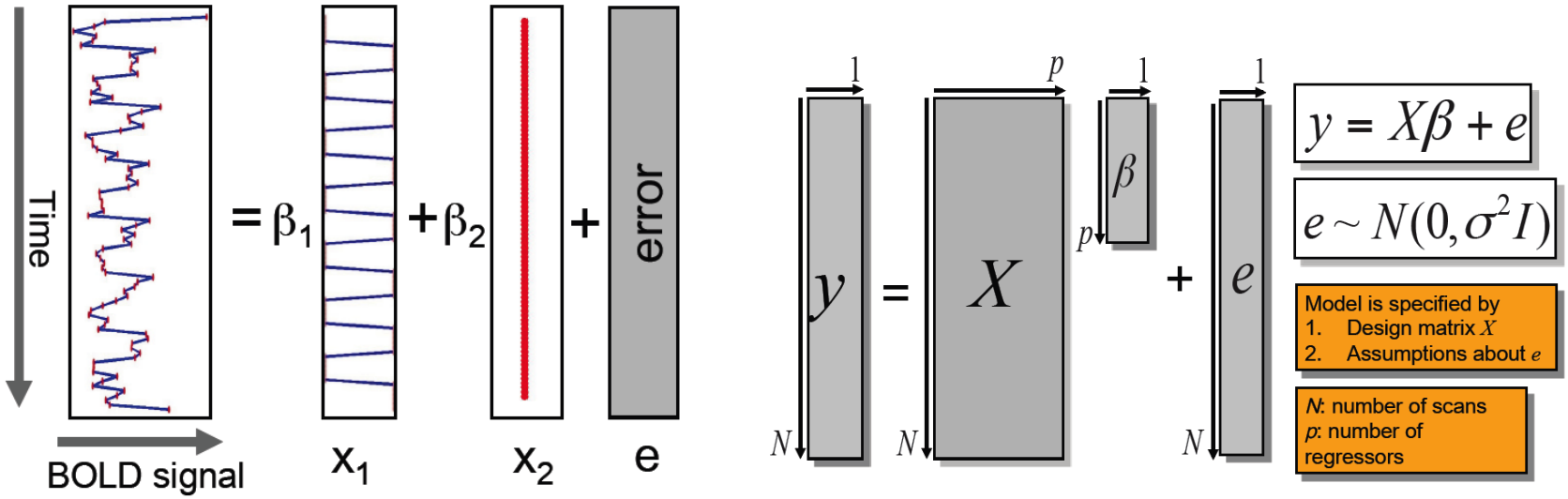
Some concepts and illustrations in this lecture are adapted from the textbook,
Statistical Parametric Mapping: The Analysis of Functional Brain Images, Editors: K. Friston, J. Ashburner, S. Kiebel, T. Nichols and W. Penny, *Academic Press*, 2006.

A simple fMRI Experiment



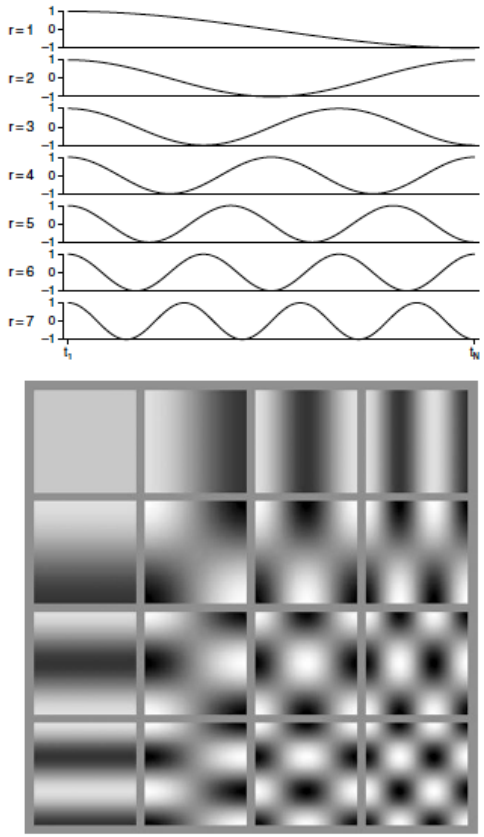
- One Session
- Passive word listening versus rest
- 7 cycles of rest and listening
- Blocks of 6 scans with 7 sec TR

Single voxel regression model



$$Y = x_1\beta_1 + x_2\beta_2 + e, \qquad e \sim \mathcal{N}(0, \sigma^2)$$

Discrete Cosine Transform

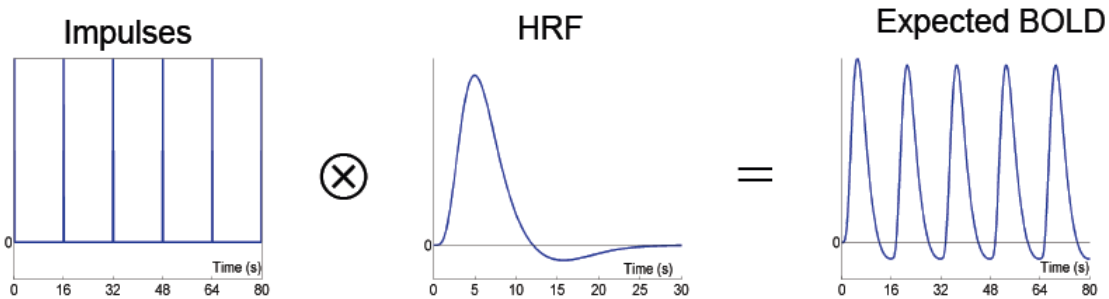


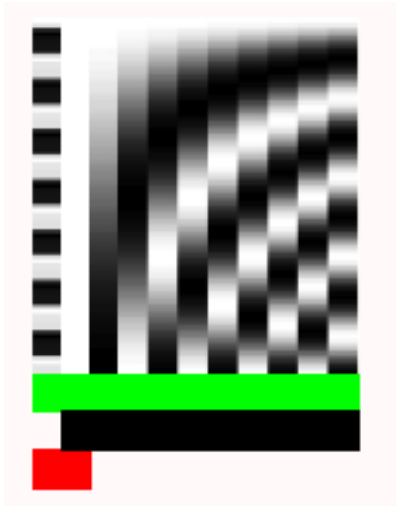
1-D Discrete Cosine Basis functions w_k are

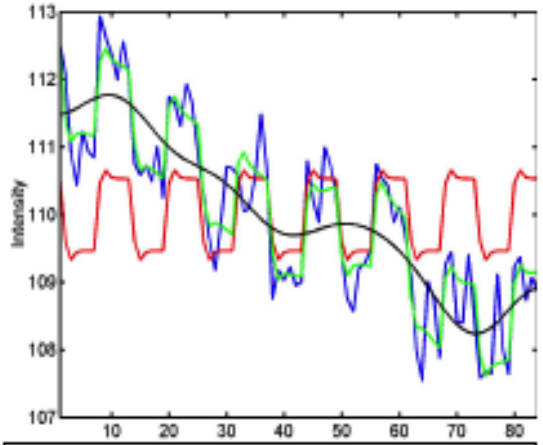
$$w_k(n) = \sqrt{\frac{2}{N}} \sum_{n=1}^{N} \cos\left(\frac{\pi}{4N}(2n-1)(2k-1)\right)$$

2-D Discrete Cosine Basis functions $W_{k,l}$ are obtained by outer product of 1-D basis functions w_k andw_l.

GLM Temporal Basis Functions Approach



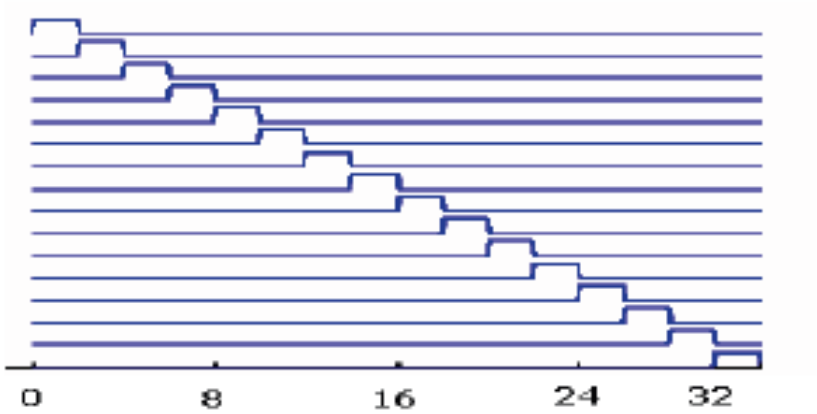


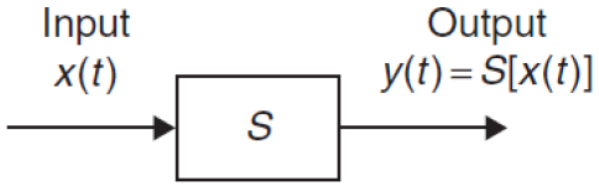


blue = data
black = mean + low-frequency drift
green = predicted response, taking into account low-frequency drift
red = predicted response (with low-frequency drift explained away)

$$y(t) = u(t) * h(t)$$
$$h(t) = \sum b_i f_i(t)$$
$$y(t) = \sum b_i (u(t) * f_i(t))$$

GLM FIR Approach



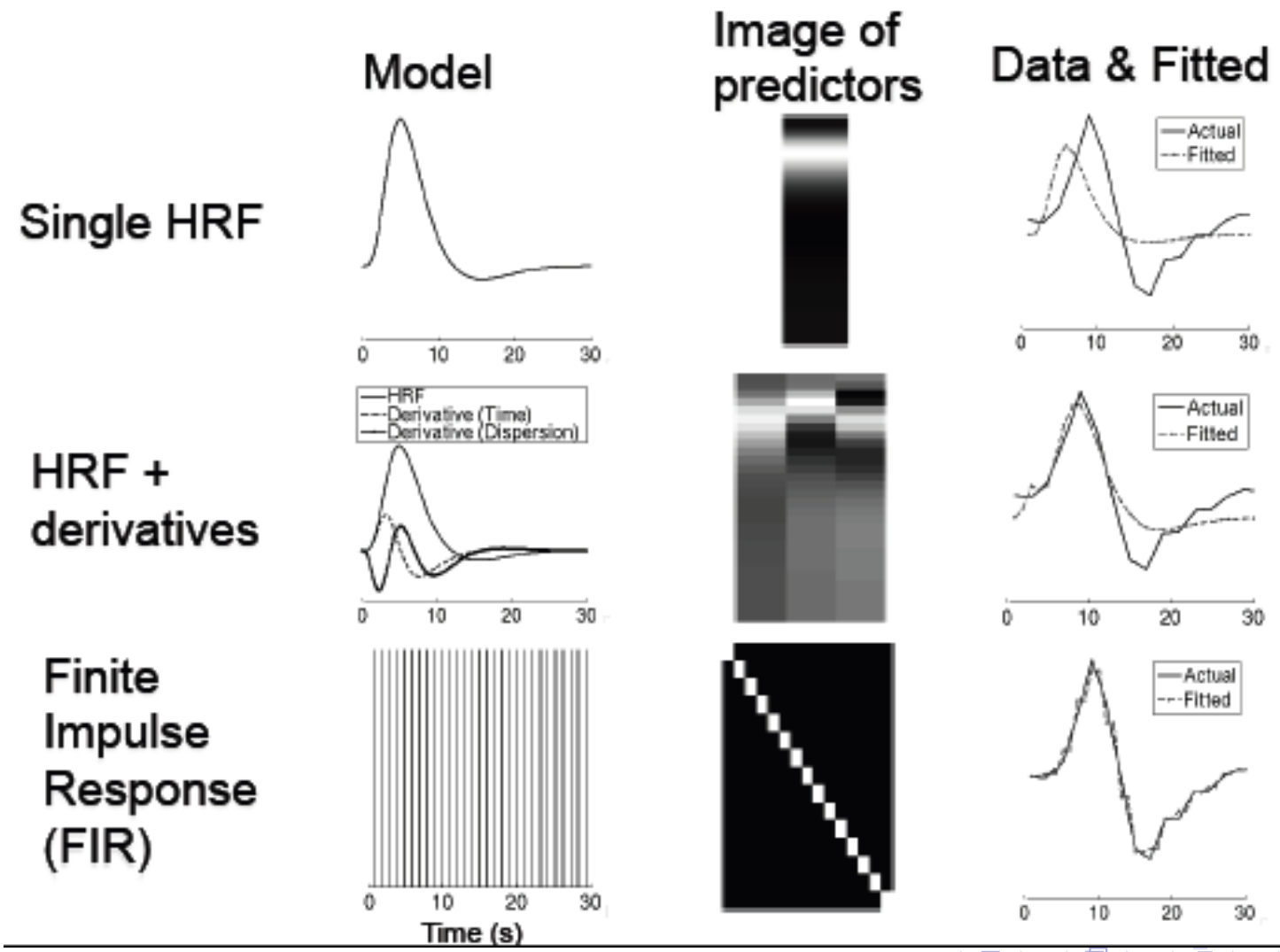


$\delta(t)$ $h(t)$

$$y[n] = \sum_k \delta[n-k]h[k]$$

The GLM parameters **b** estimate an HRF $h[n]$ of arbitrary shape for each event type in each voxel of the brain

Basis sets


Model
Image of predictors
Data & Fitted
Single HRF
HRF + derivatives
Finite Impulse Response (FIR)
HRF
Derivative (Time)
Derivative (Dispersion)
Actual
Fitted
0
10
20
30
Time (s)

Least Squares Solution

The General Linear Problem

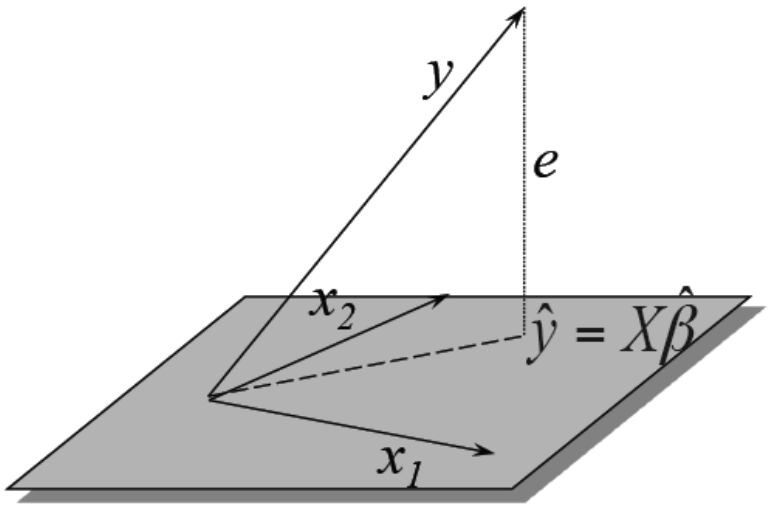
$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{e}$$

Minimize $||\mathbf{e}||^2 = ||(\mathbf{y} - \mathbf{X}\mathbf{b})||^2$

$$\frac{\partial}{\partial \mathbf{b}}||\mathbf{e}||^2 = 0 = \frac{\partial}{\partial \mathbf{b}}\mathbf{b}^T\mathbf{X}^T\mathbf{X}\mathbf{b} + \frac{\partial}{\partial \mathbf{b}}\mathbf{b}^T\mathbf{X}^T\mathbf{y} + \frac{\partial}{\partial \mathbf{b}}\mathbf{y}^T\mathbf{X}\mathbf{b}$$

$$\mathbf{b} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

A Geometric Interpretation



Design space
defined by X

P is a Projection Matrix, $\mathbf{P} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$
R is a Residual Forming Matrix, $\mathbf{R} = \mathbf{I} - \mathbf{X}^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$

Maximum Likelihood Estimation

If $\mathbf{e}$ is a sequence of correlated samples of a time series, having a covariance $\mathbf{V}$, then $\mathbf{e} \sim \mathcal{N}(0, \mathbf{V})$.

Bayes Rule

$$p(\mathbf{b}|\mathbf{y}) = p(\mathbf{y}|\mathbf{b})p(\mathbf{b})/p(\mathbf{y})$$

If $p(\mathbf{b})$ is uniform which implies we have no prior information about $p(\mathbf{b})$ then maximum likelihood estimation is performed;

$$p(\mathbf{b}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{b})$$

Maximizing log likelihood, $\frac{\partial}{\partial \mathbf{b}} \log p(\mathbf{y}|\mathbf{b})$ yields,

$$\frac{\partial}{\partial \mathbf{b}}(\mathbf{y} - \mathbf{X}\mathbf{b})^T \mathbf{V}^{-1}(\mathbf{y} - \mathbf{X}\mathbf{b}) = 0$$

from which we get weighted least squares estimation of parameters

$$\hat{\mathbf{b}} = (\mathbf{X}^T \mathbf{V}^{-1} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{V}^{-1} \mathbf{y}$$

Linear Autoregressive Models

A linear autoregressive discrete random process $x[n]$ with order p can be represented as

$$x[n] = a_1 x[n-1] + a_2 x[n-2] + \ldots + a_p x[n-p] + z[n]$$

where $z[n]$ is a multivariate white noise process with zero mean and a diagonal covariance matrix $\mathbf{V}$.
a_i are called linear the autoregressive parameters.

Autoregressive model for colored noise

AR(1) model for colored noise $w[n] = aw[n-1] + z[n]$ where $z[n] \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$$\begin{bmatrix} w[0] \\ w[1] \\ w[2] \\ \vdots \\ w[N-1] \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ a & 0 & 0 & 0 & \dots & 0 \\ 0 & a & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & a & \dots & 0 \end{bmatrix} \begin{bmatrix} w[0] \\ w[1] \\ w[2] \\ \vdots \\ w[N-1] \end{bmatrix} + \begin{bmatrix} z[0] \\ z[1] \\ z[2] \\ \vdots \\ z[N-1] \end{bmatrix}$$

$$\mathbf{W} = \mathbf{AW} + \mathbf{Z} \longrightarrow \mathbf{W}(\mathbf{I} - \mathbf{A}) = \mathbf{Z}$$

$$Cov[\mathbf{W}] = E[\mathbf{W}\mathbf{W}^T] = E[(\mathbf{I} - \mathbf{A})^{-1}\mathbf{Z}\mathbf{Z}^T(\mathbf{I} - \mathbf{A})^{-T}] =$$
$$(\mathbf{I} - \mathbf{A})^{-1}E[\mathbf{Z}\mathbf{Z}^T](\mathbf{I} - \mathbf{A})^{-T} = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{I}(\mathbf{I} - \mathbf{A})^{-T}$$

$$Cov[\mathbf{W}] = (\mathbf{I} - \mathbf{A})^{-1}(\mathbf{I} - \mathbf{A})^{-T}$$

Generation of a Colored Process

Therefore, a colored process $\mathbf{W}$ can be generated by prefiltering a white process $z[n]$ by $(\mathbf{I}-\mathbf{A})^{-1}$ *i.e.*

$$\mathbf{W} = (\mathbf{I}-\mathbf{A})^{-1}\mathbf{Z}$$

MATLAB SCRIPT FOR COLORED NOISE GENERATION

```
N = 100;
Z = randn(N,1); % Generate Normally distributed noise process
Z = (Z-mean(Z))/std(Z); % Make the noise process 0 mean and unit variance
a = [ 0.8]; % AR Coefficient with  model x(n) = ax(n-1) + z(n)
I_A = eye(N,N);
I_A = I_A - full(spdiags(ones(N,1)*a,-1:-1:-length(a),N,N)) ; % I-A
I_A_1 = pinv(I_A) ; % Coloring filter [I-A]^{-1}
W = I_A_1 * V;
```

The General Linear Problem with Colored Error Covariance

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{e}$$

If we have an error covariance matrix $\mathbf{V}$, we can prewhiten the measurements $\mathbf{y}$ by using a filter $\mathbf{F}$.

$$E[\mathbf{y}] = E[\mathbf{X}\mathbf{b}] + E[\mathbf{e}] = \mathbf{X}\mathbf{b}$$

$$Cov[\mathbf{y}] = Cov[\mathbf{X}\mathbf{b}] + Cov[\mathbf{e}] = \mathbf{V}$$

If $\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \mathbf{V})$ then $\mathbf{y} \sim \mathcal{N}(\mathbf{X}\mathbf{b}, \mathbf{V})$.
If we choose $\mathbf{F} = \mathbf{V}^{-1/2}$ and apply it to the GLM equation

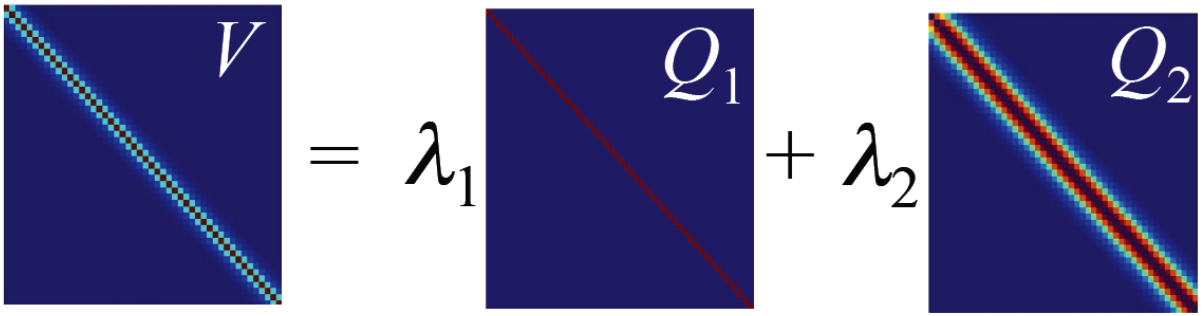
$$\mathbf{F}\mathbf{y} = \mathbf{F}\mathbf{X}\mathbf{b} + \mathbf{F}\mathbf{e}$$

$$\bar{\mathbf{y}} = \bar{\mathbf{X}}\mathbf{b} + \bar{\mathbf{e}}$$

$$Cov[\bar{\mathbf{e}}] = E[\mathbf{F}\mathbf{e}\mathbf{e}^T\mathbf{F}^T] = \mathbf{F}E[\mathbf{e}\mathbf{e}^T]\mathbf{F}^T = \mathbf{V}^{-1/2}\mathbf{V}^{1/2}\mathbf{V}^{T/2}\mathbf{V}^{T/2} = \mathbf{I}$$

$$\hat{\mathbf{b}} = (\bar{\mathbf{X}}^T\bar{\mathbf{X}})^{-1}\mathbf{X}^T\bar{\mathbf{y}}$$

GLM with Unknown Covariance $\mathbf{V}$: ReML Estimation



The log likelihood becomes

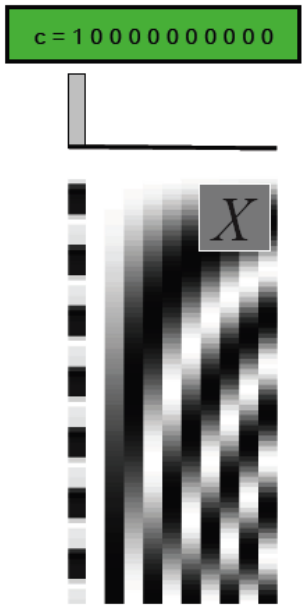
$$-\frac{1}{2}\log\mathbf{V} - \log(\mathbf{X}^T\mathbf{V}\mathbf{X}) - \frac{1}{2}(\mathbf{y}-\mathbf{X}\hat{\mathbf{b}})^T\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\hat{\mathbf{b}})$$

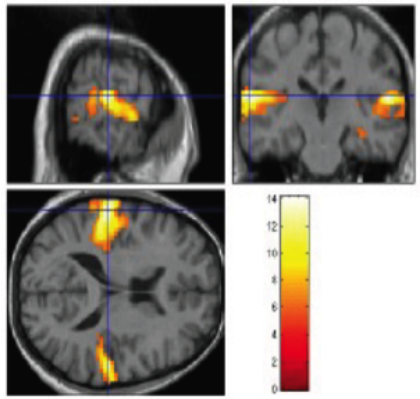
and REML estimates hyperparameters λ_i which form the error covariance matrix $\mathbf{V} = \sum_i \mathbf{Q}_i\lambda_i$ and reconstructs the parameters

$$\hat{\mathbf{b}} = (\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{V}^{-1}\mathbf{y}$$

with $\hat{\mathbf{b}} \sim \mathcal{N}(\mathbf{b}, (\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X})^{-1})$

Contrasts





Q. Any Activation during listening?

A. No activation

The Null Hypothesis:

$$\mathbf{c}^T\mathbf{b} = 0$$

$$t_\nu = \frac{\mathbf{c}^T\hat{\mathbf{b}}}{std(\mathbf{c}^T\hat{\mathbf{b}})}$$

$$= \frac{\mathbf{c}^T\hat{\mathbf{b}}}{\sqrt{\mathbf{c}^T(\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{c}}}$$

Effective degrees of freedom ν determined by Satterthwaite approximation: $\nu = \frac{trace(\mathbf{RV})^2}{trace(\mathbf{RVRV})}$

MATLAB TUTORIAL 1

Modeling HRF variation by its temporal and dispersive derivatives

```
% Generate a canonical HRF with Repetition time T =1
[hrf,p] = spm_hrf(1);
p1= p;
% Changing the hrf positive peak latency from 6 to 8
p1(1) = 8;
% Changing the hrf dispersion latency from 1 to 2
p1(3) = 2;
% Generate a varied HRF with Repetition time T =1 and with new p
[hrf1] = spm_hrf(1,p1);
% Construct the design matrix with canonical HRF and with its latency and dispersion derivatives
dl = 0.001;
dhrf_dt = (hrf-spm_hrf(1,p-[dl 0 0 0 0 0 0]))/dl;
ds = 0.001;
dhrf_ds = (hrf-spm_hrf(1,p-[0 0 ds 0 0 0 0]))/ds;
% Construct the design matrix with canonical HRF and its latency & dispersion derivative
X = [ hrf   dhrf_dt dhrf_ds ];
% determine the beta values for varied HRF
beta = X\hrf1;
% Reconstruct the varied HRF using the canonical HRF and its derivatives and plot them
plot([X*beta hrf1 hrf]);grid;
```

MATLAB TUTORIAL 2

Determining the HRF function as an FIR filter

```
Determining the HRF function as an FIR ^^Llter
% Generate 40 point random interval time axis ranging between [8 24] sec
t=randi([8 24],40,1);
% Convert this time interval axis into absolute time axis
% Generate a pulse train located at these time points and keep its size
s=zeros(max(cumsum(t)),1);
s(cumsum(t)) =1 ;
stem(s);pause;close;
% Generate an event related fmri time series using the canonical HRF
% convolved with pulse train and white noise added
y=conv(hrf,s);
y1=y + randn(size(y))*0.05;
plot([y y1]);pause;close;
% Construct the design matrix X using the convolution matrix function in matlab
X=convmtx(s,length(hrf));
% Apply GLM to determine the FIR parameters and plot it along with the
% original HRF convolved with the pulse train which is supposed to be the FIR
beta = inv(X'*X)*X'*y1;
plot([hrf beta]);grid;pause;close;
```

MATLAB TUTORIAL 3

A simple GLM

```
[hrf,p] = spm_hrf(1);
x= zeros(100,1);
x([11:20 41:60 71:90]) = 1;
y = convmtx (x,length(hrf))*hrf;
y = y(1:length(x),1:end);
y1 = y + randn(size(y))*0.4;
plot([x y y1+0*sin([0:length(y1)-1]'*0.01)]);axis([0 100 -2 6]); pause ; close;
X = [ones(size(y1)) y-mean(y)] ;
plot([X]);axis([0 100 -2 6]);
beta = inv(X'*X)*X'*y1;
c= [0 1]';
R = eye(length(y1),length(y1)) - X*inv(X'*X)*X';
sigma = std (R*y1);
T = c'*beta / sqrt((c'*inv(X'*X)*c)*sigma);
V = sigma^2*eye(length(y1),length(y1));
nu = (trace (R*V))^2/trace(R*V*R*V);
p = 1 - tcdf(T,nu)
```

MATLAB TUTORIAL 4

Modeling the drift using DCT5

```
N=100;
xd=[0:N-1]'; % Drift Signal
w = ones(1,N)/sqrt(N); % DC component
d=sqrt (2/N)*cos([1:N-1]'*[1:2:2*N-1]*pi/(2*N)); % higher frequency components
d=[w;d];
y = dct(xd); % estimating the DCT coefficients
k = 6; % Number of DCT components to model the drift
plot([ xd d(1:k,:)'*y(1:k)  ]);pause;close;
[hrf,p] = spm_hrf(1);
x= zeros(N,1);
x([11:20 41:60 71:90]) = 1;
y = convmtx (x,length(hrf))*hrf;
y = y(1:length(x),1:end);
y1 = y + randn(size(y))*0.4 ; % Add noise to BOLD
y2 =  y1 + 0.04*xd; % Add a drift to BOLD
plot([0.04*xd x y y1 y2]); pause ; close;
% Design Matrix
X = [   y-mean(y) ones(size(y))   d(2:k,:)'  ] ;
Xinv = pinv(X'*X);
beta = Xinv*X'*y2;
c= [1 zeros(1,k)]';
R = eye(length(y2),length(y2)) - X*Xinv*X';
plot([x y1 y2    y X(:,1:2)*beta(1:2)  ]); grid;pause ; close;
```