3dREMLfit - Math Notes
Serial Correlation in AFNI

RW Cox / summer 2008

also see 3dREMLfit -help and 3dREMLfit.ppt

What is REML? Why do we care?

- Variance / correlation estimation!
- Linear regression gives us fit parameters (βs) $\Rightarrow$ HRF
- To assess statistics, need variance estimates

- Ordinary Least Squares (OLSQ)

$$\underline{y} = \underline{\underline{X}}\,\underline{\beta} + \sigma\,\underline{\varepsilon}$$

where $\underline{\varepsilon}$ is Gaussian "white" noise

$$E[\underline{\varepsilon}\,\underline{\varepsilon}^T] = \underline{\underline{I}}_n$$

= uncorrelated

$\underline{y}$	$\underline{\underline{X}}$	$\underline{\beta}$	$\sigma\,\underline{\varepsilon}$
n-vector of data	n x m vector of model	m-vector of fit params	noise level
scanner	thinking	TBD	TBD

$$\Rightarrow_{\text{OLSQ}} \quad \hat{\beta} = \underbrace{\left(\underline{\underline{X}}^T\underline{\underline{X}}\right)^{-1}\underline{\underline{X}}^T}_{\text{pseudo-inverse of } \underline{\underline{X}}}\,\underline{y} \quad \Rightarrow \quad \hat{\underline{y}} = \underline{\underline{X}}\,\hat{\beta} = \text{fitted model time series}$$

$$\Rightarrow \hat{\sigma}^2 = \|\underline{y} - \hat{\underline{y}}\|^2/(n-m) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2/(n-m)$$

= variance estimate

- Even if noise model is incorrect, so that $E[\underline{\varepsilon}\,\underline{\varepsilon}^T] = \underline{\underline{R}} \neq \underline{\underline{I}}_n$, the estimator $\hat{\beta} = \left(\underline{\underline{X}}^T\underline{\underline{X}}\right)^{-1}\underline{\underline{X}}^T\underline{y}$ is consistent (un-biased)
- But the noise variance estimator $\hat{\sigma}^2$ is biased if $\underline{\underline{R}} \neq \underline{\underline{I}}_n$
- N.B.: $\underline{\underline{R}}$ has diagonal entries $R_{ii} = 1$; R_{ij} = correlation of noise at time #i with #j

- **Maximum Likelihood (ML) Estimation**
  - Suppose $\underline{\underline{R}} = \underline{\underline{R}}(\underline{\theta})$ $\underline{\theta}$ = some (small) set of unknown parameters giving the structure of $\underline{\underline{R}}$'s entries

e.g.

$$\underline{\underline{R}} = \begin{bmatrix} 1 & a & a^2 & a^3 & \cdots \\ a & 1 & a & a^2 & \cdots \\ a^2 & a & 1 & a & \\ a^3 & a^2 & a & 1 & \\ \vdots & & & & \end{bmatrix}$$

where $|a| < 1$ $\{\underline{\theta} = a\}$

[Note: Toeplitz structure = constant down diagonals]

↔ This is the AR(1) model for $\underline{\underline{R}}$

  - Gaussian likelihood of data $\underline{y}$, given this model:

$$L(\underline{\beta}, \sigma^2, \underline{\theta}) = \underbrace{\frac{1}{(2\pi)^{n/2}}}_{\text{ignore this!}} \frac{1}{\det[\underline{\underline{R}}]^{1/2}} \frac{1}{(\sigma^2)^{n/2}} e^{-\frac{1}{2\sigma^2}[\underline{y}-\underline{\underline{X}}\underline{\beta}]^T \underline{\underline{R}}^{-1}[\underline{y}-\underline{\underline{X}}\underline{\beta}]}$$

$$-2\log L = \ell = \ell_{ML} = \log\det[\underline{\underline{R}}] + n\cdot\log(\sigma^2) + \frac{1}{\sigma^2}[\underline{y}-\underline{\underline{X}}\underline{\beta}]^T \underline{\underline{R}}^{-1}[\underline{y}-\underline{\underline{X}}\underline{\beta}]$$

  - maximize L ⇒ minimize $\ell_{ML}(\underline{\beta}, \sigma^2, \underline{\theta})$ over all parameters at once

  - $\underline{\beta}$ is easy: $\hat{\underline{\beta}} = (\underline{\underline{X}}^T \underline{\underline{R}}^{-1} \underline{\underline{X}})^{-1} \underline{\underline{X}}^T \underline{\underline{R}}^{-1} \underline{y}$ given $\underline{\underline{R}}(\underline{\theta})$

  - σ^2 is easy: $\hat{\sigma}^2 = \frac{1}{n}[\underline{y}-\underline{\underline{X}}\hat{\underline{\beta}}]^T \underline{\underline{R}}^{-1}[\underline{y}-\underline{\underline{X}}\hat{\underline{\beta}}]$ given $\underline{\underline{R}}(\underline{\theta})$

  - $\underline{\underline{R}}(\underline{\theta})$ isn't so easy: have to do nonlinear optimization in general case
  - for AR(p) model, there are closed form algorithms for getting the p values in $\underline{\theta}$, without a general purpose optimizer being required (but these aren't ML-optimal).

- Problem with ML estimation of $\hat{\sigma}^2$: it is <u>biased</u>
- It's value, averaged over all possible noise instances, is not the true σ^2. (unlike the case for $\hat{\beta}$)
- Problem: combined estimation of linear ($\hat{\beta}$) and quadratic ($\hat{\sigma}^2$) parameters in L: no one knows how to exactly eliminate bias in $\hat{\sigma}^2$. (In general case, that is, where $\underline{\underline{R}}$ is unknown).

REML (1970s)

- ML optimizes L over $\underline{\beta}, \sigma^2, \underline{\theta}$ all at once
- REML = Restricted (or Residual) ML: only optimizes over space of residuals, after the model $(\underline{\underline{X}}\,\hat{\underline{\beta}})$ is removed from the data $\Rightarrow$ <u>Variance estimate is less biased</u>

$$\ell_{REML} = \log\det\left[\underline{\underline{R}}(\underline{\theta})\right] + \log\det\left[\underline{\underline{X}}^T\underline{\underline{R}}(\underline{\theta})^{-1}\underline{\underline{X}}\right] \quad \left\{\text{these depend on } \underline{\theta}, \text{ but not on the data } \underline{y}\right.$$

$$+ (n-m)\log\left(\underline{y}^T\underline{\underline{P}}\underline{y}\right) - \underbrace{\log\det\left[\underline{\underline{X}}^T\underline{\underline{X}}\right]}_{\text{indep of } \underline{\theta}}$$

where the matrix $\underline{\underline{P}}$ is

$$\underline{\underline{P}} = \underline{\underline{R}}^{-1} - \underline{\underline{R}}^{-1}\underline{\underline{X}}\left(\underline{\underline{X}}^T\underline{\underline{R}}^{-1}\underline{\underline{X}}\right)^{-1}\underline{\underline{X}}^T\underline{\underline{R}}^{-1} \qquad \text{and } \underbrace{\underline{\underline{P}} = \underline{\underline{P}}^T}_{\text{symmetric}} \quad \underline{\underline{P}} = \underline{\underline{P}}\,\underline{\underline{R}}\,\underline{\underline{P}}$$

= prewhitening projection matrix onto space $\perp$ to columns of $\underline{\underline{X}}$

- ℓ_{REML} is a function of $\underline{\theta}$, to be minimized. Doing this efficiently is the key. The matrices are big ($n \approx 1000$ $m \approx 100$ typically).

REML & ARMA(1,1)

In solving system $\underline{y} = \underline{X}\underline{\beta} + \underline{\eta}$ $\quad \underline{y} \in \mathbb{R}^n$ $\quad \underline{X} \in \mathbb{R}^{n \times m}$ $\quad \underline{\beta} \in \mathbb{R}^m$

with $\underline{\eta} \sim N(\underline{0}, \sigma^2 \underline{R}(\underline{\theta}))$ $\quad (n > m)$

and σ^2 unknown, $\underline{R}(\underline{\theta})$ = correlation matrix with unknown $\underline{\theta}$

the REML method estimates $\underline{\theta}$ by minimizing (independent of $\underline{\beta}$)

$$\ell(\underline{\theta}) = (n-m)\log\left[\underline{y}^T \underline{P} \underline{y}\right] + \log\det\left[\underline{R}(\underline{\theta})\right] + \log\det\left[\underline{X}^T \underline{R}(\underline{\theta})^{-1} \underline{X}\right] - \log\det\left[\underline{X}^T \underline{X}\right]$$

(last term independent of $\underline{\theta}$, ignored since we're not varying $\underline{X}$)

where $\underline{P} = \underline{P}^T = \underline{P}\,\underline{R}\,\underline{P}$

$$= \underline{R}^{-1} - \underline{R}^{-1}\underline{X}\left[\underline{X}^T \underline{R}^{-1} \underline{X}\right]^{-1} \underline{X}^T \underline{R}^{-1}$$

Once $\hat{\underline{\theta}}$ is estimated, then $\hat{\sigma}^2 = \underline{y}^T \underline{P}(\hat{\underline{\theta}})\,\underline{y}\,/(n-m)$ is the REML estimate of variance, and $\hat{\underline{\beta}} = \left[\underline{X}^T \underline{R}^{-1}(\hat{\underline{\theta}}) \underline{X}\right]^{-1} \underline{X}^T \underline{R}^{-1} \underline{y}$ is the regression estimate.

N.B. $\underline{y}^T \underline{P} \underline{y}$ = norm of prewhitened residuals

Computations: Cholesky decompose $\underline{R}(\underline{\theta}) = \underline{C}^T \underline{C}$ and $\underline{X}^T \underline{R}^{-1} \underline{X} = \underline{D}^T \underline{D}$

[$\underline{C}, \underline{D}$ are upper triangular; $\underline{C}$ is $n \times n$ and $\underline{D}$ is $m \times m$]

Then $\underline{y}^T \underline{P} \underline{y} = \underline{y}^T \underline{P}^T \underline{C}^T \underline{C} \underline{P} \underline{y}$ (since $\underline{P} = \underline{P}^T \underline{R} \underline{P}$)

$$= \left\| \underline{C}\,\underline{P}\,\underline{y} \right\|^2$$

$$\underline{P} = \underline{C}^{-1}\underline{C}^{-T} - \underline{C}^{-1}\underline{C}^{-T}\underline{X}\left[\underline{D}^T\underline{D}\right]^{-1}\underline{X}^T\underline{C}^{-1}\underline{C}^{-T}$$

$$\Rightarrow \underline{C}\,\underline{P} = \underline{C}^{-T} - \underline{C}^{-T}\underline{X}\left[\underline{D}^T\underline{D}\right]^{-1}\underline{X}^T\underline{C}^{-1}\underline{C}^{-T} \Longrightarrow$$

Compute $\underline{C}\,\underline{P}\,\underline{y}$:

(n)	solve	$\underline{C}^T \underline{b}_1 = \underline{y}$	$\left[\underline{b}_1 = \underline{C}^{-T}\underline{y}\right]$
(n)	solve	$\underline{C}\,\underline{b}_2 = \underline{b}_1$	
* (m)	multiply	$\underline{b}_3 = \underline{X}^T \underline{b}_2$	
(m)	solve	$\underline{D}^T \underline{b}_4 = \underline{b}_3$	
(m)	solve	$\underline{D}\,\underline{b}_5 = \underline{b}_4$	$(= \hat{\underline{\beta}})$
* (n)	multiply	$\underline{b}_6 = \underline{X}\,\underline{b}_5$	(= fitted model)
(n)	solve	$\underline{C}^T \underline{b}_7 = \underline{b}_6$	(= prewhitened model)
(n) $\Rightarrow$		$\underline{C}\,\underline{P}\,\underline{y} = \underline{b}_1 - \underline{b}_7$	(= prewhitened resid)

Note: $\underline{C}^{-T}$ is "prewhitening matrix" since

$$E\left[\left(\underline{C}^{-T}\underline{\eta}\right)\left(\underline{C}^{-T}\underline{\eta}\right)^T\right] = \underline{C}^{-T}\underline{R}\,\underline{C}^{-1}$$

$$= \underline{C}^{-T}\underline{C}^T\underline{C}\,\underline{C}^{-1} = \underline{I}_n$$

Note: the terms $\log\det[\underline{\underline{R}}(\theta)] + \log\det[\underline{\underline{X}}^T\underline{\underline{R}}(\theta)^{-1}\underline{\underline{X}}]$ are like a "penalty" favoring some values of $\underline{\theta}$ and penalizing others. As it happens, at least for the ARMA(1,1) model, the $a=b=0$ case is the most penalized; that is, these terms favor correlations $\neq \underline{\underline{I}}_n$.

$$\left(\underline{\underline{X}}^T\underline{\underline{R}}^{-1}\underline{\underline{X}}\right) = \left[\underline{\underline{X}}^T\underline{\underline{C}}^{-1}\underline{\underline{C}}^{-T}\underline{\underline{X}}\right] = \left[\underline{\underline{C}}^{-T}\underline{\underline{X}}\right]^T\left[\underline{\underline{C}}^{-T}\underline{\underline{X}}\right]$$

$\therefore$ $\underline{\underline{D}}$ can be computed as the "R" upper triangular factor directly from $\underline{\underline{C}}^{-T}\underline{\underline{X}}$ using the QR algorithm (no need to compute $\underline{\underline{X}}^T\underline{\underline{R}}^{-1}\underline{\underline{X}}$)

and $$\log\det\left[\underline{\underline{X}}^T\underline{\underline{R}}^{-1}\underline{\underline{X}}\right] = \log\det\left[\underline{\underline{D}}^T\underline{\underline{D}}\right] = 2\log\det\left[\underline{\underline{D}}\right] = 2\sum_{i=1}^{m}\log D_{ii}$$
(D_{ii}: diagonal elements of $\underline{\underline{D}}$)

similarly,
$$\log\det[R] = 2\sum_{i=1}^{n}\log C_{ii}$$ (C_{ii}: diagonal elements of $\underline{\underline{C}}$ matrix)

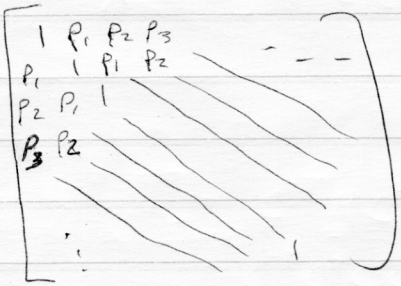
So, the above outlines the REML $\ell(\underline{\theta})$ calculation given the model $\underline{\underline{R}}(\underline{\theta})$, $\underline{\underline{X}}$, and the data $\underline{y}$. Normally, $\underline{\underline{X}}^T\underline{\underline{R}}^{-1}\underline{\underline{X}}$ and so $\underline{\underline{D}}$ will be (nearly) full matrices, but $\underline{\underline{R}}$ (and so $\underline{\underline{C}}$) will be banded, since points far apart are not expected to be correlated. Since n is (usually) much bigger than m, we don't want to form a full $n\times n$ R matrix; we want to use the sparse structure to save space & time.

ARMA(1,1): $\eta_n = u_n + b u_{n-1} + a\eta_{n-1}$ $\quad$ a = AR parameter $\in(-1,1)$
$u_i \sim N(0,1)$ i.i.d. $\quad$ b = MA parameter $\in(-1,1)$

We find that $\rho_k = \dfrac{\langle \eta_n \eta_{n-k}\rangle}{\langle \eta_n^2\rangle} = 1$ for $k=0$ (duh)

$$= \frac{(b+a)(1+ab)\,a^{k-1}}{1+2ab+b^2} \qquad k \geq 1$$

So $\underline{\underline{R}} =$



= Toeplitz
(not allowing for censoring and multiple runs)

It's simplest to define $\lambda = \rho_1 = \frac{(b+a)(1+ab)}{1+2ab+b^2}$ and then $\rho_k = \begin{cases} 1 & k=0 \\ \lambda \rho^{k-1} & k \geq 1 \end{cases}$ [where $\rho = a$]

In fact, a reasonable model for FMRI noise is AR(1) + white noise, which would have a covariance matrix like

$$\sigma_A^2 \begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 & \cdots \\ & \ddots & & & \end{bmatrix} + \sigma_W^2 \begin{bmatrix} 1 & 0 & 0 & & \\ 0 & 1 & & & \\ & & \ddots & & \end{bmatrix}$$

and so a correlation matrix of

$$R = \begin{bmatrix} 1 & \delta\rho & \delta\rho^2 & \delta\rho^3 & \cdots \\ & \ddots & & & \end{bmatrix}$$

where $\delta = \frac{\sigma_A^2}{\sigma_A^2 + \sigma_W^2} \in [0,1]$.

= fraction of noise that is AR(1).

This is ARMA(1,1) with $a = \rho$ and $\lambda = \delta\rho$, so $0 \leq \lambda \leq \rho$.

We'll limit ourselves to $\rho = 0, 0.1, 0.2, \ldots, 0.8$. Note that $0.8^{18} \approx 0.02$, so the $\underline{\underline{R}}$ matrix is pretty much confined to the near region of the main diagonal. We will use $\delta = 0.1, 0.2, \ldots, 1.0$, so we'll have 80 cases of $\underline{\theta} = [\rho, \delta]^T$ plus 1 more case for white noise $[\rho = 0, \delta = 0]$, with $\underline{\underline{R}} = \underline{\underline{I}}$.

obsolete!

a little out of date

Since $\underline{\underline{R}}$ will be banded, $\underline{\underline{C}}$ will be banded. Special purpose matrix functions have been written to deal with such matrices, see remla.c.

(For a pure AR(1) model, $\underline{\underline{C}}$ and $\underline{\underline{C}}^{-1}$ is very simple, but that's a special case.)

(Note: $\underline{\underline{C}}^{-1}$ and $\underline{\underline{D}}^{-1}$ are never formed; since $\underline{\underline{C}}$ and $\underline{\underline{D}}$ are upper triangular)

In actuality, $\underline{\underline{R}}$ needs to allow for censoring and inter-run gaps. Thus, we compute

for $i > j$: $R_{ij} = \lambda \rho^{\tau(i)-\tau(j)}$ $\quad [\lambda = \rho_1$, as before$]$

where $\tau(i)$ = "true" time index before censoring,
and where the p^{th} run starts at $\tau = 10000p$ (so that $\tau(i)-\tau(j)$ is very large between runs and $R_{ij} \to 0$)

$$\hat{\beta} = \left[\underline{\underline{X}}^T \underline{\underline{R}}^{-1} \underline{\underline{X}}\right]^{-1} \underline{\underline{X}}^T \underline{\underline{R}}^{-1} \underline{y}$$

$$= \underline{\underline{D}}^{-1} \underline{\underline{D}}^{-T} \underline{\underline{X}}^T \underline{\underline{C}}^{-1} \underline{\underline{C}}^{-T} \underline{y} = \underline{b}_5 \text{ from page ①}$$

Later) The actual implementation uses a grid in (a,b) [not in (a,λ) as I originally planned]. The grid runs (by default) for $a \in [0, 0.8]$ and $b \in [-0.8, 0.8]$, with 2^G subdivisions (2^G+1 points) in the a direction, and 2^{G+1} subdivisions in the b direction. The default value of G is 3, so the grid is $a = 0, 0.1, 0.2, \ldots, 0.8$ (but the user can increase G to get a finer mesh). Binary search in 2D is used to find the optimal ℓ_{REML} value. (Which is why the grid is a power-of-2 in size!)

ℓ_{REML} is computed in the 7 step procedure on page ①, for any given data vector $\underline{y}$. $\underline{\underline{C}}$ is stored as a banded matrix, where $\underline{\underline{R}}$ is cut off when $|R_{ij}| < 0.01$. $\underline{\underline{X}}$ is stored as a sparse matrix if that is advantageous, since multiplications by $\underline{\underline{X}}$ and $\underline{\underline{X}}^T$ take up about 50% of the CPU time in this 7 step algorithm. Note that in calculating ℓ_{REML}, it is necessary to calculate $\hat{\beta}$ and $\hat{y}$ as well (steps 5 & 6).

GLTs in 3dREMLfit

We want to test the hypothesis that $\underline{\underline{G}}\,\underline{\beta} \stackrel{?}{=} \underline{0}$, where $\underline{\underline{G}}$ is an $r \times m$ matrix. For example, to test if any of the βs in a particular stimulus class are zero/nonzero, then r = # of βs in that stimulus, and if the β indexes in that stimulus class are β_{s_i} for $i = 1, 2, \ldots, r$, then the elements of $\underline{\underline{G}}$ are $g_{i,s_i} = 1$ $i = 1 \ldots r$, $g_{ij} = 0$ otherwise; e.g.

$$\underline{\underline{G}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$ where $\beta_3, \beta_4, \beta_5$ are being tested.

We test this hypothesis by computing $\hat{\beta}_R$ from $\hat{\beta}$, where $\hat{\beta}_R$ is the solution to the model $\underline{y} = \underline{\underline{X}}\,\underline{\beta}$ subject to the constraint $\underline{\beta} = 0$.

The GLSQ solution of this, given $\underline{\underline{R}}$, is

$$\hat{\beta}_R = \left[\underline{\underline{I}}_m - \underbrace{\left[\underline{\underline{X}}^T \underline{\underline{R}}^{-1} \underline{\underline{X}}\right]^{-1} \underline{\underline{G}}^T \left[\underline{\underline{G}} \left(\underline{\underline{X}}^T \underline{\underline{R}}^{-1} \underline{\underline{X}}\right)^{-1} \underline{\underline{G}}^T\right]^{-1} \underline{\underline{G}}}_{\text{define this to be } \underline{\underline{J}} = m \times m \text{ matrix}} \right] \hat{\beta}$$

subscript "R" means "restricted"

$\hat{y}_R = \underline{\underline{X}}\,\hat{\beta}_R$ = fitted model in restricted solution space

$\hat{w}_R = \underline{\underline{C}}^{-T} \underline{\underline{X}}\,\hat{\beta}_R$ = pre-whitened fitted restricted model

$SSE_R = \| \underline{\underline{C}}^{-T} y - \hat{w}_R \|^2$ = sum of squares of this restricted model fit

$$F_R = \frac{SSE_R - SSE_F}{r} \Big/ \frac{SSE_F}{n-m}$$ = F-statistic for hypothesis $\underline{\underline{G}}\,\underline{\beta} \stackrel{?}{=} 0$ with $(r, n-m)$ DoF

SSE_F = full model sum of squares

$= \| \underline{\underline{C}}^{-T} (y - \hat{y}) \|^2 = \| \underline{\underline{C}}\,\underline{\underline{P}}\,y \|^2$ from p(1)

For efficient calculation, write the matrix $\underline{\underline{J}}$ as

$$\underline{\underline{X}}^T \underline{\underline{R}}^{-1} \underline{\underline{X}} = \underline{\underline{D}}^T \underline{\underline{D}} \quad \text{(from before)} \qquad \underline{\underline{D}} \text{ is upper triangular } m \times m$$

$$\left(\underline{\underline{X}}^T \underline{\underline{R}}^{-1} \underline{\underline{X}}\right)^{-1} = \underline{\underline{D}}^{-1} \underline{\underline{D}}^{-T}$$

so

$$\underline{\underline{J}} = \underline{\underline{D}}^{-1} \underline{\underline{D}}^{-T} \underline{\underline{G}}^T \left[\underline{\underline{G}} \underline{\underline{D}}^{-1} \underline{\underline{D}}^{-T} \underline{\underline{G}}^T \right]^{-1} \underline{\underline{G}}$$

form matrix $\underline{\underline{F}} = \underline{\underline{D}}^{-T} \underline{\underline{G}}^T$ ($m \times m$ $m \times r$ $\Rightarrow$ $m \times r$), then QR factor it to matrix $\underline{\underline{Q}}\,\underline{\underline{E}}$ ($m \times r$ $r \times r$ upper triangular)

then $\left[\underline{\underline{G}} \underline{\underline{D}}^{-1} \underline{\underline{D}}^{-T} \underline{\underline{G}}^T \right]^{-1} = \left[\underline{\underline{F}}^T \underline{\underline{F}} \right]^{-1} = \left[\underline{\underline{E}}^T \underline{\underline{E}} \right]^{-1} = \underline{\underline{E}}^{-1} \underline{\underline{E}}^{-T}$

so

$$\underline{\underline{J}} = \underbrace{\underline{\underline{D}}^{-1} \underline{\underline{F}}}_{m \times r} \underbrace{\underline{\underline{E}}^{-1} \underline{\underline{E}}^{-T} \underline{\underline{G}}}_{r \times m}$$

$\equiv \underline{\underline{J}}_{left}$ $\equiv \underline{\underline{J}}_{right}$

(since $\underline{\underline{D}}$ and $\underline{\underline{E}}$ (and $\underline{\underline{C}}$) upper triangular, no need to actually invert these matrices; solving the systems is a trivial exercise)

Special cases:

$\underline{\underline{G}}$ has 1 row ($r=1$) then $\underline{\underline{D}}^{-T} \underline{\underline{G}}^T$ is just an m-vector and $\underline{\underline{G}} \underline{\underline{D}}^{-1} \underline{\underline{D}}^{-T} \underline{\underline{G}}^T$ is a scalar $= \| \underline{\underline{D}}^{-T} \underline{\underline{G}}^T \|^2$

Forming $\underline{\underline{F}}$:
- extract each row of $\underline{\underline{G}}$ into an m-vector $\underline{u}$
- solve $\underline{\underline{D}}^T \underline{v} = \underline{u}$ for the m-vector $\underline{v}$ (vector_rtrtran_solve in matrix.c)
- put $\underline{v}$ into $\underline{\underline{F}}$ matrix column

Next:

If $r=1$: $f = \|\underline{v}\|^2$ $\quad \underline{\underline{J}}_R = \frac{1}{f} \underline{\underline{G}}$

If $r>1$:
- use matrix_qrr in matrix.c to compute $\underline{\underline{E}}$ from $\underline{\underline{F}}$
- for each column of $\underline{\underline{G}}$
  - extract it to an r-vector $\underline{u}$
  - solve $\underline{\underline{E}}^T \underline{v} = \underline{u}$ for $\underline{v}$
  - solve $\underline{\underline{E}} \underline{w} = \underline{v}$ for $\underline{w}$
  - put $\underline{w}$ into column of $\underline{\underline{J}}_R$

Next: for each column of $\underline{\underline{F}}$,
- extract it to m-vector $\underline{u}$
- solve $\underline{\underline{D}}\underline{v} = \underline{u}$ for $\underline{v}$ (vector_rr_solve function)
- put $\underline{v}$ into column of $\underline{\underline{J}}_L$

Define $\hat{\underline{\beta}}_G = \underline{\underline{G}}\hat{\underline{\beta}}$

Covariance matrix of $\hat{\underline{\beta}}$ is $\mathrm{Cov}(\hat{\underline{\beta}} - \underline{\beta}_{true}) = (\underline{\underline{X}}^T \underline{\underline{R}}^{-1} \underline{\underline{X}})^{-1}$

so
$$\mathrm{Cov}(\hat{\underline{\beta}}_G - \underline{\beta}_G) = \underline{\underline{G}}(\underline{\underline{X}}^T \underline{\underline{R}}^{-1} \underline{\underline{X}})^{-1} \underline{\underline{G}}^T$$
$$= \underline{\underline{G}}\,\underline{\underline{D}}^{-1}\underline{\underline{D}}^{-T}\underline{\underline{G}}^T = \underline{\underline{E}}^T\underline{\underline{E}}$$

so Variance estimate for the i^{th} component of $\hat{\underline{\beta}}_G$ is

$$\tilde{\sigma}_i^2 = \hat{\sigma}^2 \cdot [\underline{\underline{E}}^T\underline{\underline{E}}]_{ii} = \hat{\sigma}^2 \cdot \sum_{j \le i} E_{ji}^2 \quad \text{(sum down columns of } \underline{\underline{E}}\text{)}$$

and the t-statistic for its significance (relative to the test $\hat{\beta}_{G,i} \stackrel{?}{=} 0$) is

$$t_i = \hat{\beta}_{G,i} \Big/ \tilde{\sigma}_i = \hat{\beta}_{G,i} \Big/ \sqrt{\hat{\sigma}^2 \cdot \sum_{j \le i} E_{ji}^2}$$

In 3dREMLfit, all F and t statistics are computed as GLTs. In 3dDeconvolve, special code deals with the simpler cases when $\underline{\underline{G}}$ is a matrix that just picks out some elements of $\underline{\beta}$. Since the REML fitting (a,b) estimation is by far the slowest part of the program, I decided to keep the code simple and only have a unified set of functions for dealing with statistics.

Although $\underline{\underline{G}}$ is usually quite sparse, it turns out not to speed things up to store $\underline{\underline{G}}$ as a sparse matrix – probably because r is usually so small. For $\underline{\underline{X}}$, it made quite a difference to store it in sparse form.

ARMA(1,1)

$$Y_n = U_n + bU_{n-1} + aY_{n-1} \qquad |a|<1 \quad |b|<1$$

$U_n \sim N(0,1)$ i.i.d.

[NoB.: this Y is not the signal, just the noise. Sorry for the notational mixup!]

Define $E[Y_n Y_{n-k}] = V_k$ = covariance at lag k

Note: $E[Y_n U_n] = 1$

$$V_0 = E\left[(U_n + bU_{n-1} + aY_{n-1})^2 \right]$$

$$= E\Big[U_n^2 + \underbrace{2bU_nU_{n-1}}_{=0} + 2a\underbrace{U_nY_{n-1}}_{=0} + b^2\overbrace{U_{n-1}^2}^{1} + 2ab\overbrace{U_{n-1}Y_{n-1}}^{1} + a^2\underbrace{Y_{n-1}^2}_{V_0} \Big]$$

$$= 1 + b^2 + 2ab + a^2 V_0 \quad \Rightarrow \boxed{V_0 = \frac{1+b^2+2ab}{1-a^2}}$$

$$V_1 = E\left[(U_n + bU_{n-1} + aY_{n-1}) Y_{n-1} \right]$$

$$= E\Big[\underbrace{U_nY_{n-1}}_{0} + b\underbrace{U_{n-1}Y_{n-1}}_{1} + a\underbrace{Y_{n-1}^2}_{V_0} \Big] = b + aV_0$$

$$= \frac{b - a^2b + a + ab^2 + 2a^2b}{1-a^2} = \boxed{\frac{(b+a)(1+ab)}{1-a^2}}$$

$$V_k = E\left[(U_n + bU_{n-1} + aY_{n-1}) Y_{n-k} \right] \qquad k>1$$

$$= E\Big[\underbrace{U_nY_{n-k}}_{0} + b\underbrace{U_{n-1}Y_{n-k}}_{0} + a\underbrace{Y_{n-1}Y_{n-k}}_{V_{k-1}} \Big] = aV_{k-1}$$

$$\therefore V_k = \frac{(b+a)(1+ab)}{1-a^2}\, a^{k-1} \qquad k \geq 1 \quad \Rightarrow \quad r_k = \text{correlation \# } k = \frac{(a+b)(1+ab)\,a^{k-1}}{1+2ab+b^2}$$

(R)

Further 3dREMLfit Implementation Details

- function matrix_qrr (in matrix.c) computes the "R" factor in the QR decomposition of the input matrix. This factor is upper triangular. No pivoting or balancing is done.
- functions vector_rr_solve and vector_rrtan_solve solve systems of the form $\underline{\underline{R}}\,\underline{v} = \underline{u}$ and $\underline{\underline{R}}^T\underline{v} = \underline{u}$ (respectively), where $\underline{\underline{R}}$ is upper triangular (this $\underline{\underline{R}}$ is not the correlation matrix!)
- similarly, matrix_rr_solve and matrix_rrtan_solve are functions to solve systems of the form $\underline{\underline{R}}\,\underline{\underline{U}} = \underline{\underline{V}}$ where $\underline{\underline{U}}$ and $\underline{\underline{V}}$ are now matrices.
- Sparse matrices are handled in 2 separate ways. $\underline{\underline{X}}$ can be stored sparsely (if at most 30% of its entries are nonzero). This "sparmat" struct (in remla.c) stores for each column j in $\underline{\underline{X}}$, a list of i indexes for which $X_{ij} \neq 0$, and the values of such X_{ij} elements. (If the column is full, the i-list is not needed.) Then there are 2 functions, vector_spc_multiply and vector_spc_multiply_transpose, to multiply $\underline{\underline{X}}$ and $\underline{\underline{X}}^T$ into vectors.
- The 'rcmat' struct holds narrow-banded matrices, which are either considered to be symmetric or triangular (depending on context). For each column (or row), the number of elements is stored and an element vector — the last element of which is the diagonal element. For example:

$$\begin{bmatrix} a_0 & b_0 & 0 & d_0 & 0 \\ & b_1 & c_0 & d_1 & 0 \\ & & c_1 & d_2 & e_0 \\ & & & d_3 & e_1 \\ & & & & e_2 \end{bmatrix}$$

column #0 has 1 element
column #1 has 2 elements
column #3 has 2 elements
column #4 has 4 elements
column #5 has 3 elements

Functions (in remla.c) exist to Cholesky decompose such a matrix (considered as symmetric), solve systems where the matrix is upper or lower triangular, and setup such a matrix from the ARMA(1,1) model.