FinM 331/Stat 339 Financial Data Analysis, Winter 2010

Floyd B. Hanson, Visiting Professor

Email: fhanson@uchicago.edu

Master of Science in Financial Mathematics Program
University of Chicago

Lecture 5

6:30-9:30 pm, 01 February 2010, Ryerson 251 in Chicago

7:30-10:30 pm, 01 February 2010 2010 at UBS in Stamford

7:30-10:30 am, 02 February 2010 at Spring in Singapore

5. Regression and Maximum Likelihood Tools:

5.1 Parametric Regression:[a]

Parametric regression involves **fitting to the sample data a small number of parameters**, e. g., mean μ and variance σ^2, while by contrast **nonparametric regression involving fitting on more of the local properties of the data** with much data preparation to smooth fitted curves. In addition, there are many flavors of parametric regression. Some do not rely on random properties much, like the **simple least squares** relating a **response data vector** analogous to an **dependent** variable **to a model depending on a few predictors** or **regressors** analogous to independent variables **and a few unknown parameters or coefficients that are to be estimated**. Other regression methods, such as **maximum likelihood estimation**, as the name implies, **rely on more probabilistic methods**, but then so do stochastic models in finance.

[a] See Rice (2007) Chapt. 14 or Carmona (2004) Chapt. 3 for more information or Weisberg (2005) Chapts. 1-4 for even more details on both statistical and computational errors.

5.2 Ordinary or Simple Least Squares Regression for Univariate Linear Models:

Given a sample of n observations of **responses** $\vec{Y} = [Y_i]_{n\times 1}$ that appear to satisfy a nearly linear relationship with respect to the same size sample of observations $\vec{X} = [X_i]_{n\times 1}$ called **predictors** or **regressors**. The **nearly linear relationship** could be suggested by plotting $\vec{Y}$ against $\vec{X}$ by using **MATLAB**'s `scatter(Y,X);` scatter plot function and eyeballing the plotted points. Least squares suggests that our cost function be the square the deviations from the linear model $y = \phi(x) \equiv b + mx$, where m is the slope and b is the intercept and minimize or find the least value of the sum of the squares of the deviation of the data from the straight line, i.e., our **objective is the quadratic cost function**,

$$\mathcal{L}_2(b,m) = \sum_{i=1}^{n} (Y_i - b - mX_i)^2. \tag{5.1}$$

The observations will not likely satisfy the model $\phi(x)$, instead there will be an error $Y_i = b + mX_i + e_i$ or $e_i \equiv Y_i - b - mX_i$ and $\mathcal{L}_2(b,m) = \sum_{i=1}^{n} e_i^2 = (n-1)\tilde{\sigma}_e^2$.

Alternatively, the sum of the absolute deviations,

$$\mathcal{L}_1(m,b) = \sum_{i=1}^{n} |Y_i - m \cdot X_i - b|, \tag{5.2}$$

but the cost model does not have such nice and justifiable properties.

The minimum can be found by searching for the critical points with respect to the unknown parameter set (b, m), so

$$\frac{\partial \mathcal{L}_2}{\partial b}(m,b) = -2\sum_{i=1}^{n}(Y_i - mX_i - b) = -2n(\overline{Y} - m\overline{X} - b) \stackrel{*}{=} 0, \tag{5.3}$$

yielding the implicit optimal intercept estimation,
$\widehat{b} = b^* = \overline{Y} - \widehat{m} \cdot \overline{X}$, where $(\overline{X}, \overline{Y})$ is the usual sample mean vector of the coordinates. Next,

$$\begin{aligned} \frac{\partial \mathcal{L}_2}{\partial m}(m,b) &= -2\textstyle\sum_{i=1}^{n}(Y_i - mX_i - \widehat{b})X_i \\ &= -2n(\overline{XY} - m\overline{X^2} - \widehat{b}\overline{X}) \stackrel{*}{=} 0, \end{aligned} \tag{5.4}$$

yielding the **least squares solution**,

$$\widehat{m} = m^* = \frac{\overline{XY} - \overline{X}\,\overline{Y}}{\overline{X^2} - (\overline{X})^2} = \frac{\widehat{\sigma}_{x,y}}{\widehat{\sigma}_x^2} \quad \text{if} \quad \widehat{\sigma}_x^2 > 0. \tag{5.5}$$

Hence, the **estimated slope** $\widehat{m}$ is given as the ratio of the data sample covariance to the variance of the **predictor** $\vec{X}$, both being biased or unbiased quadratic moments. **However, this estimated slope formula has possibly small differences in numerator and denominator, so its computation should be handled with care due to "Catastrophic Cancellations".** These are cancellations where significant digits are lost in subtractions.

The **estimated intercept** $\widehat{b}$ is usually calculated once the estimated slope $\widehat{m}$ is calculated since it serves no numerical purpose to eliminate the slope from the intercept formula, but could lead to a less precise calculation.

The **observation error** $\vec{e}$ between observed residual using the model comes with statistical conditions: $\mathbf{E}[\vec{e}] = \vec{0}$ since otherwise there will be bias so any non-zero value should be in the model and not in the noise and $\mathbf{Cov}[\vec{e}, \vec{e}^{\top}] = \sigma_e^2 I_n$ for **constant "sigma"** σ_e or else will interfere with the least squares fit.

The quality of the linear regression is just by how close the noted **coefficient of determination R^2** value is to one and is defined by formula,

$$R^2 \equiv 1 - \frac{\text{SSE}}{\text{TSS}} = 1 - \frac{(n-2)\widehat{\sigma}_e^2}{(n-1)\widehat{\sigma}_y^2}, \tag{5.6}$$

$\{$***Almost Total Correction!***$\}$, where the

$$\text{SSE} = \text{RSS} \equiv \sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2 \equiv \sum_{i=1}^{n}\widehat{e}_i^2 = (n-2)\widehat{\sigma}_e^2 \tag{5.7}$$

is the **sum of squared errors (SSE) or residual sum of squares (RSS)**, with $\widehat{Y}_i \equiv \widehat{m}X_i + \widehat{b}$ is the ith estimated response, with the 2 less degrees of freedom due to parameters accounting for the factor $n-2$. The

$$\text{TSS} \equiv \sum_{i=1}^{n}(Y_i - \overline{Y})^2 = (n-1)\widehat{\sigma}_y^2 \tag{5.8}$$

is the **total sum of squares (TSS) of the deviation**.

• ***Linear Regression Translation to Linear Algebra & MATLAB:***

Translating the linear regression to matrix-vector form, combine the parameters into a parameter vector be $\vec{p}=[b,m]^\top$, then the independent variable $(n\times 1)\vec{X}$ is extended to the $(n\times 2)$-matrix $A=[\texttt{ones(n,1)},\vec{X}]$, so that the usual linear algebraic equation $A*\vec{x}=\vec{b}$ is equivalent to the form $A\vec{p}=\vec{Y}$ with the correspondence $\vec{b}=\vec{Y}$ is the known RHS and $\vec{x}=\vec{p}$ is the unknown. Th linear regression, least squares problem is the find the minimum of the scalar quadratic form:

$$S(\vec{x})\equiv\left(A\vec{x}-\vec{b}\right)^\top\left(A\vec{x}-\vec{b}\right)=\vec{x}^\top A^\top A\vec{x}-2\vec{x}^\top A^\top\vec{b}+\vec{b}^\top\vec{b}, \tag{5.9}$$

after transpose algebra $((AB)^\top=B^\top A^\top)$. Since if $\vec{V}$ is a fixed, then

$$\nabla_x[\vec{x}^\top\vec{V}]=\nabla_x[\vec{V}^\top\vec{x}]=\vec{V} \tag{5.10}$$

and

$$\nabla_x[S](\vec{x})=2\left(A^\top A\vec{x}-A^\top\vec{b}\right)\stackrel{*}{=}\vec{0}, \tag{5.11}$$

at a critical point $\vec{x}^*$ if it exists, where the least squares solution is

$$\vec{p}^*=\vec{x}^*=\left(A^\top A\right)^{-1}A^\top\vec{b}. \tag{5.12}$$

- ***Ordinary Least Squares for Simple Linear Fit Example:***

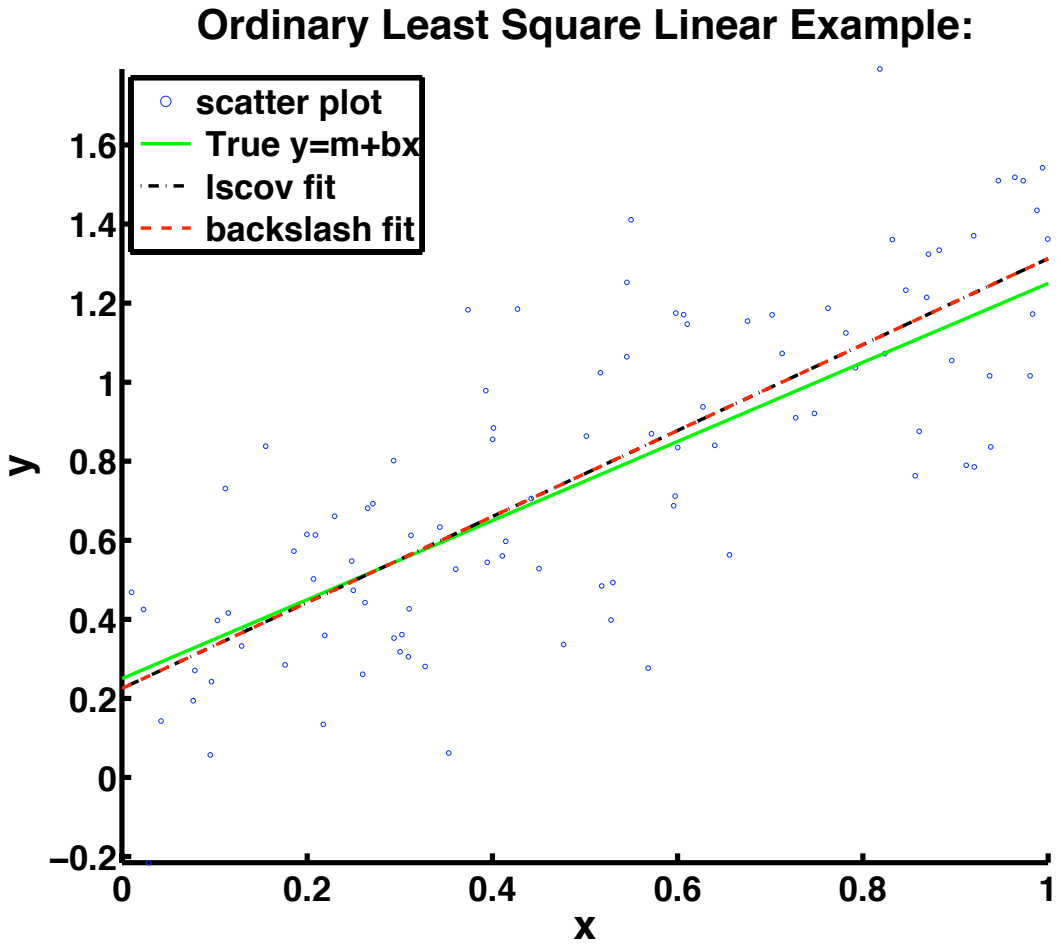

Figure 5.1: Ordinary least squares fit using indistinguishable **MATLAB** `lscov` and `A\b` **backslash** methods, with similar results for a simple, straight line fit with `scatterplot` data.

• *MATLAB Code for Simple Ordinary Least Squares Application:*

```
function LStest
clc
fprintf('\nLStest Output (%s):',datestr(now));
m = 1; b = 0.25; % True parm values;
fprintf('\nTrue: b = %7.4f; m = %7.4f;',b,m);
ptrue = [b;m];
n = 100;
x = rand(n,1); % Simulated  Uniform x-data;
sigma_e = 0.25;
err = 0.25*randn(n,1); % Simulated Normal Mean-Zero Error;
fprintf('\nsigma_e = %7.4f;',sigma_e);
y = b+m*x+err; % Simulated y-data with linear model;
TSS = (n-1)*var(y);
fprintf('\nTSS  = %9.3e',TSS);
A = [ones(n,1) x];
fprintf('\nsize(A)=[%i,%i]; size(y)=[%i,%i];',size(A),size(y));
[preg,sep,mse] = lscov(A,y); % Least Sqs Method, No Cov input
pbsl = A\y; % BackSlash (BS) Method (fast);
breg = preg(1); mreg = preg(2);
yhatreg = breg+mreg*x;
yhatbsl = pbsl(1)+pbsl(2)*x;
SSEreg = sum((y-yhatreg).^2); % Also, RSS=ResSumSqs
SSEbsl = sum((y-yhatbsl).^2);
```

```
fprintf('\nRSS=SSEreg = %9.3e; SSEbsl = %9.3e;',SSEreg,SSEbsl);
fprintf('\nRelResreg = %9.3e;',sqrt(norm(y-yhatreg)/norm(y)));
fprintf('\nRelResbsl = %9.3e;',sqrt(norm(y-yhatbsl)/norm(y)));
Rsqreg = 1-SSEreg/TSS; Rsqbsl = 1-SSEbsl/TSS;
fprintf('\nRsqreg = %9.3e; Rsqbsl = %9.3e;',Rsqreg,Rsqbsl);
fprintf('\nTrue:  b =%7.4f; m =%7.4f;',b,m);
fprintf('\nlscov: b = %7.4f; m = %7.4f;',breg,mreg);
fprintf('\nbkslh: b = %7.4f; m = %7.4f;',pbsl(1),pbsl(2));
fprintf('\nsqrt(norm(preg-ptrue)/norm(ptrue)) =%7.4f;' ...
    ,sqrt(norm(preg-ptrue)/norm(ptrue)));
fprintf('\nsqrt(norm(pbsl-ptrue)/norm(ptrue)) =%7.4f;' ...
    ,sqrt(norm(pbsl-ptrue)/norm(ptrue)));
fprintf('\nsqrt(norm(preg-pbsl)/norm(pbsl))   =%7.4f;' ...
    ,sqrt(norm(preg-pbsl)/norm(pbsl)));
fprintf('\nlscov: sep = [%9.3e,%9.3e];',sep);
fprintf('\nlscov: mse = %9.3e;',mse); % Caution: different scaling
xg = 0:0.1:1;
ytrue = b+m*xg;
yhatregp = breg+mreg*xg;
yhatbslp = pbsl(1)+pbsl(2)*xg;
%
figure(1); nfig = 1;
scrsize = get(0,'ScreenSize'); % figure spacing for target screen
ss = [5.0,4.5,4.0,3.5]; % figure spacing factors
```

```
scatter(x,y,8); hold on;
plot(xg,ytrue,'-g',xg,yhatregp,'-.k',xg,yhatbslp,'--r','LineWidth',2);
axis tight; hold off;
title('Ordinary Least Square Linear Example:'...
    ,'Fontsize',24,'FontWeight','Bold');
xlabel('x','Fontsize',24,'FontWeight','Bold');
ylabel('y','Fontsize',24,'FontWeight','Bold');
legend('scatter plot',' True y=m+bx',' lscov fit',' backslash fit' ...
    ,'Location','NorthWest');
set(gca,'Fontsize',20,'FontWeight','Bold','LineWidth',3);
set(gcf,'Color','White','Position' ...
   ,[scrsize(3)/ss(nfig) 60 scrsize(3)*0.60 scrsize(4)*0.80]); %[l,b,w,h]
fprintf('\n ');

============ LStest Output ========================

LStest Output (30-Jan-2010 14:34:46):
True: b =  0.2500; m =  1.0000;
sigma_e =  0.2500;
```

```
TSS  = 1.659e+01
RSS=SSEreg = 6.494e+00; SSEbsl = 6.494e+00;
RelResreg = 5.403e-01;
RelResbsl = 5.403e-01;
Rsqreg = 6.085e-01; Rsqbsl = 6.085e-01;
True:  b = 0.2500; m = 1.0000;
lscov: b =  0.2253; m =  1.0873;
bkslh: b =  0.2253; m =  1.0873;
sqrt(norm(preg-ptrue)/norm(ptrue)) = 0.2967;
sqrt(norm(pbsl-ptrue)/norm(ptrue)) = 0.2967;
sqrt(norm(preg-pbsl)/norm(pbsl))   = 0.0000;
lscov: sep = [5.125e-02,8.809e-02];
lscov: mse = 6.627e-02;
 >>
```

Remark: Some scaling differences mean variables like `mse`, supposed mean square error, are quite different, say from the Relative Residual `RelRes=sqrt(norm(y-yhat)/norm(y))`.

5.3 Multiple Linear Regressor Models with a Univariate Response:

Generalizing the prior univariate predictor variable, assuming again **univariate response** sample data, $\vec{Y} = [Y_i]_{n\times 1}$, consider the linear relationship to m-dimensional multivariate predictor with sample data array $X = [X_{i,j}]_{n\times m}$ and $\vec{1} \equiv [1]_{n\times 1}$,

$$\vec{Y} \simeq p_0\vec{1} + X\vec{p} \tag{5.13}$$

with predictor-independent coefficient p_0 and linear predictor coefficient or **parameter vector** $\vec{p} = [p_i]_{m\times 1}$. Define the **linear error** in the model to be

$$\vec{e} = [e_i]_{n\times 1} \equiv \vec{Y} - p_0\vec{1} - X\vec{p}. \tag{5.14}$$

Next we absorb the p_0 parameter by defining **extended linear parameter and model basis arrays** with the constant vector $\vec{X}_0 = [X_{i,0}]_{n\times 1} \equiv \vec{1}$,

$$\vec{p}_{\mathrm{ex}} \equiv [p_{i-1}]_{(m+1)\times 1} \;\&\; A \equiv [\vec{1}, X] = [X_{i,j-1}]_{n\times(m+1)}, \tag{5.15}$$

yielding the **multivariate linear model** in compact form,

$$\vec{Y} = A\vec{p}_{\mathrm{ex}} + \vec{e}. \tag{5.16}$$

• ***Multiple Linear (Multilinear) Least Squares Regression:***[a]

The least squares objective corresponds to minimizing the quadratic costs or residual sum of squares in 2-norm form,

$$\mathcal{L}(\vec{p}_{\mathbf{ex}}) = \mathbf{RSS}(\vec{p}_{\mathbf{ex}}) = \left(\vec{Y} - A\vec{p}_{\mathbf{ex}}\right)^{\top}\left(\vec{Y} - A\vec{p}_{\mathbf{ex}}\right) = \left\|\vec{Y} - A\vec{p}_{\mathbf{ex}}\right\|^{\mathbf{2}}. \quad (5.17)$$

In order to find the multivariate critical points with respect to the unknown parameter vector $\vec{p}_{\mathbf{ex}}$, we expand the quadratic form into more elementary products

$$\mathcal{L}(\vec{p}_{\mathbf{ex}}) = \vec{Y}^{\top}\vec{Y} - 2\vec{p}_{\mathbf{ex}}{}^{\top}A^{\top}\vec{Y} + \vec{p}_{\mathbf{ex}}{}^{\top}A^{\top}A\vec{p}_{\mathbf{ex}}, \quad (5.18)$$

where the reverse product transpose identity $(AB)^{\top} = B^{\top}A^{\top}$ has been used. Next the **gradient peel theorem**[b], $\nabla_{\vec{p}}\left[\vec{p}^{\top}\vec{Y}\right] = \vec{Y}$, with three applications, implies the critical point conditions,

$$\nabla_{\vec{p}_{\mathbf{ex}}}[\mathcal{L}(\vec{p}_{\mathbf{ex}})] = -2A^{\top}\vec{Y} + 2A^{\top}A\vec{p}_{\mathbf{ex}} \stackrel{*}{=} \vec{0}_{(m+1)\times 1}. \quad (5.19)$$

---

[a] **MATAB** reserves **multivariate linear regression** for a vector system of multiple linear regressor models of responses $\vec{y}$ and a array of predictors $\mathcal{X}$, as in a portfolio application.

[b] Hanson '07, Online Appendix B Preliminaries: Probability and Analysis Results, p. B45, *http://www.math.uic.edu/ hanson/pub/SIAMbook/bk0BprelimAppendfinal.pdf*

This leads to the **optimal parameter estimate**, provided that the $(m+1) \times (m+1)$ square matrix $(A^\top A)$ is invertible,

$$\widehat{p}_{\mathbf{ex}} = \vec{p}^{\,*}_{\mathbf{ex}} = \left(A^\top A\right)^{-1} A^\top \vec{Y}, \tag{5.20}$$

but the **non-square A will not be invertible in the ordinary sense, the $A^\top A$ may not be positive definite for $m > 1$, and there are possible catatrophic cancellation problems with estimated parameter formula**. The optimum, i.e., minimum, $\widehat{p}_{\mathbf{ex}}$, of the parameter vector depends on technical conditions, the most important is the objective quadratic form, the **parameter vector is linear in the response observation vector $\vec{Y}$**. This **estimate is unbiased** conditional on A, with $\mathbf{E}[\vec{e}] = \vec{0}$ and $\mathbf{Cov}[\vec{e}, \vec{e}^{\,\top}] = \sigma_e^2 I_n$ from the usual assumptions, is that $\vec{e} \overset{\mathbf{dist}}{=} \mathcal{N}(\vec{0}, \sigma_e^2 I_n)$, also it is assumed that $\mathbf{E}[e_i^2 | X_i]$ is constant, so

$$\begin{aligned} \mathbf{E}[\widehat{p}_{\mathbf{ex}} \mid A] &= \left(A^\top A\right)^{-1} A^\top \mathbf{E}[\vec{Y} \mid A] \\ &= \left(A^\top A\right)^{-1} A^\top \mathbf{E}[A\vec{p}_{\mathbf{ex}} + \vec{e} \mid A] \\ &= \left(A^\top A\right)^{-1} A^\top A \vec{p}_{\mathbf{ex}} = \vec{p}_{\mathbf{ex}}, \end{aligned} \tag{5.21}$$

which we take as the true parameter value.

In general, $A^\top A$ may not be positive definite due to negative X-correlations. **Testing for positive definiteness**, using $\vec{\overline{X}} \equiv \frac{1}{n}[\sum_{k=1}^n X_{k,i}]_{m\times 1}$ and $\overline{X^\top X} \equiv \frac{1}{n}[\sum_{k=1}^n X_{k,i}X_{k,j}]_{m\times m}$,

$$\begin{aligned}\tfrac{1}{n}\vec{p}_{\mathbf{ex}}^{\top}A^\top A\vec{p}_{\mathbf{ex}} &= \begin{bmatrix} p_0 & \vec{p}^{\top}\end{bmatrix}\begin{bmatrix} 1 & \vec{\overline{X}}^{\top} \\ \vec{\overline{X}} & \overline{X^\top X}\end{bmatrix}\begin{bmatrix} p_0 \\ \vec{p}\end{bmatrix} \\ &= p_0^2 + p_0\vec{\overline{X}}^{\top}\vec{p} + \vec{p}^{\top}\overline{X^\top X}\vec{p} \\ &= \left\| p_0 + \vec{\overline{X}}^{\top}\vec{p}\right\|^2 + \vec{p}^{\top}\mathbf{Cov}\left[X^\top, X\right]\vec{p},\end{aligned} \tag{5.22}$$

where the **completing the square** technique was used and $\mathbf{Cov}\left[X^\top, X\right] \equiv \frac{1}{n}[\sum_{k=1}^n (X_{k,i}-\overline{X}_i)(X_{k,j}-\overline{X}_i)]_{m\times m}$. So, when $m=1$ predictor variable with a sum of squares and semi-positive definiteness, but for $m>1$ definiteness may not be true if there are negative elements of variable covariance. Care must be taken in choosing statistical software.

The **least squares parameter solution** has a **geometric interpretation** that the least squares residual $\vec{\mathbf{Res}} = \vec{Y} - A\widehat{p}_{\mathbf{ex}}$ must be orthogonal to the predictor $A\widehat{p}_{\mathbf{ex}}$. This follows from some linear algebra starting with the scalar product orthogonal test:

$$\begin{aligned}(A\widehat{p}_{\mathbf{ex}})^{\top}\vec{\mathbf{Res}} &= \widehat{p}_{\mathbf{ex}}^{\top}A^{\top}(\vec{Y} - A\widehat{p}_{\mathbf{ex}}) \\ &= \widehat{p}_{\mathbf{ex}}^{\top}(A^{\top}\vec{Y} - A^{\top}A\widehat{p}_{\mathbf{ex}}) \\ &\stackrel{*}{=} \widehat{p}_{\mathbf{ex}}^{\top}\vec{0}_{(m+1)\times 1} = 0,\end{aligned} \tag{5.23}$$

by Eq. (5.19). Hence, the residual $\vec{\mathbf{Res}}$, the response observations $\vec{Y}$ and optimal predicted response $A\widehat{p}_{\mathbf{ex}}$ form a right triangle with $\vec{Y}$ on the hypotenuse.

The **estimated parameter covariance** matrix is similarly given by

$$\begin{aligned}\Sigma_{\widehat{p}_{\mathbf{ex}}|A} &= \mathbf{E}\left[(\widehat{p}_{\mathbf{ex}}-\vec{p}_{\mathbf{ex}})(\widehat{p}_{\mathbf{ex}}-\vec{p}_{\mathbf{ex}})^{\top}\middle|A\right] \\ &= (A^{\top}A)^{-1}A^{\top}\mathbf{E}\left[\vec{e}\,\vec{e}^{\top}\right]A(A^{\top}A)^{-1} \\ &= \sigma_e^2\cdot(A^{\top}A)^{-1}.\end{aligned} \tag{5.24}$$

The estimated response vector is defined by

$$\widehat{Y} \equiv A\widehat{p}_{\mathbf{ex}} = A(A^{\top}A)^{-1}A^{\top}\vec{Y} \equiv H\vec{Y}, \tag{5.25}$$

where $H \equiv A(A^{\top}A)^{-1}A^{\top} = [H_{i,j}]_{n\times n}$ is called the **hat or prediction matrix** and note that it is a square symmetric matrix. This matrix enters into the raw residuals or errors as $\widehat{e} \equiv \vec{Y} - \widehat{Y} = (I_n - H)\vec{Y}$ with mean $\mathbf{E}[\widehat{e}|A] = \vec{0}$ and covariance $\Sigma_{\widehat{e}|A} = \sigma_e^2(I_n - H)$, obviously correlated due to the $(I_n - H)$.

The **minimum residual sum of squares** is given on page 6 by $\mathbf{RSS} = ||\widehat{e}||^2 \equiv ||\vec{Y} - \widehat{Y}||^2$ and the unbiased variance associated with this residual sum of squares, when $n > m+1$, is $\widehat{\sigma}^2_{\widehat{e}} = ||\widehat{e}||^2/(n-m-1)$, where the **number of degrees of freedom** have been reduced by the number of parameters, $(m+1)$. The **standardized residuals** are given approximately by

$$\widehat{e}_i^{(\mathbf{std})} \simeq \widehat{e}_i/(\widehat{\sigma}_{\widehat{e}}\sqrt{1-H_{i,i}}). \tag{5.26}$$

The **F-test** can be used for a test of normality, such that

$$F = \frac{\mathbf{SS}_{reg}}{\widehat{\sigma}^2_e} = \frac{\mathbf{TSS} - \mathbf{RSS}}{\widehat{\sigma}^2_e}, \tag{5.27}$$

where $\mathbf{SS}_{reg}$ is the **sum of squares due to the regression**. (Note similarity to R^2).

In computational statistics, the relative residual,

$$\mathbf{RelRes}_y = ||\vec{Y} - \widehat{Y}||/||\vec{Y}|| \tag{5.28}$$

is often used since it is readily calculated once the parameter estimate is available.

• *MATLAB Functions for Least Squares:*

MATLAB contains a large amount of functions for linear regression and its variations, but only a few are listed here.

1. **Back Slash Method or Left Matrix Division** ($\backslash$ or **`mldivide`**): remember the statement that A did not have a ordinary inverse, well it has a generalized inverse such that $\widehat{p} = A \backslash \vec{Y}$ (read this from right to left for A **divided into** $\vec{Y}$, as you would read $\vec{Y}/A$ in the reverse order) producing a least square approximation solution,[a] the method depending on input. The **`mldivide`** uses many methods depending on **`A`**. The more general **`regress`** function uses powerful backslash technique, but produces added statistics.
2. **`[phat,stdp,mse]=lscov(A,Y);`** produces the least square solution to $A\vec{x} = \vec{b}$ by solving $(A\vec{p} - \vec{Y})'(A\vec{p} - \vec{Y})$ for $(m \times 1)$ **`phat`** from $(n \times m)$ **`A`** and $(n \times 1)$ **`Y`** (in our notation, $(n, m) \rightarrow (n, m+1)$ and (A, Y, p) $\rightarrow$ $(A, \vec{Y}, \vec{p}_{\mathrm{ex}})$). The extra output are estimated standard errors in **`stdp`** of **`phat`** and mean squared error in **`mse`**. See the examples in **`help lscov`** .

[a] D. & N. Higham '05, MATLAB Guide, pp.126-127, say that it actually performs faster and more accurately than some other methods.

3. `[phat,pbci,r,rci,stats]=regress(X,Y,alpha);`
primary produces the the $\ell \times 1$ estimated linear parameter coefficient array `phat`$= \widehat{p}$ for the responses in univariate `Y` from **multilinear regression** on $\ell \times n$ predictor data `X` and $n \times 1$ response data `y`. **(Caution: If a constant parameter p_0, then the first column of `X` is a column of ones and $\ell = m+1$.)** The `bci` is a $\ell \times 1$ array for estimated parameter confidence intervals in each of ℓ parameters using an ***out of confidence*** parameter `alpha`, `r` is the $n \times 1$ linear fit residuals or errors, `rci` is a $n \times 1$ residual confidence intervals for outlier diagnosis, an outlier being outside the `100(1-alpha)%` residual confidence interval, and `stats` is an 1×4 statistics row vector containing $[R^2-\text{statistic}, F-\text{statistic}, P-\text{value of F}, \widehat{\sigma}_e^2]$. See `Help regress` for an example application for $\ell = 4$, where `A=[ones(size(x1)),x1,x2,x1.*x2]`, the fourth *linear* term is a data cross product $x1.*x2$, and a `scatter3` scatterplot of the y vs. $(x1, x2)$ results. Note, multilinear only means a linear combination of functions of data and are also called **basis functions**.

4. `stats=regstats(y,x,model,whichstats);` produces similer linear regression results like **`regress`**, but make possible a richer set of statistical diagnostic tests and allows linear combinations of both linear and quadratic functions of $\vec{x}$. The option **`model`** can be `'linear','purequadratic','interaction','quadratic'`, the latter is a general quadratic function. The option **`whichstats`** allows many statistical tests and measures with output in the output structure **`stats`** and using `'all'` gives all, but if the output structure is omitted, then which statistics can be selected on a easier to use GUI menu.

5. `[b,stats]=robustfit(x,y,wfun,tune);` primary produces the weighted least square or linear regression for the $p \times 1$ estimated linear parameter coefficient array `b`$=\widehat{p}$ for the responses in univariate `y` from **multilinear regression** on $\ell \times n$ predictor data `x` and $n \times 1$ response data `y` **(Caution: If a constant parameter p_0 is in the model, then `robustfit` automatically adds ones to the first column of `x` and $\ell = m$ at input, so do NOT add ones. Also, the input order for `x,y` in robustfit is opposite to the tmbty,x in `regress` and `regstats`.)** See **`Help robustfit`** for a long list of options. If the options **`wfun,tune`**, are omitted the bi-square weight function and its tuning constant are used.

{**Note: The `robustfit` weighting can be used for identifying data outliers, so in traditional practice can be discarded or lessened by weighting or could be enhanced if value at risk was a serious concern. See also `recoplot`, the residual order plot.**}

6. `[p,S,mu]=polyfit(x,y,n);` finds the fit for vector `y` as a polynomial `p` of degree `n` in standardized vector variable $\widehat{x}$`=(x-mu(1))/mu(2)` (i.e., note `m=1`), a **highly recommended centering and scaling of data to improve any method computation**, where `mu=[mean(x),std(x)];`. Also `S` is the structure that gives extra information (see Help) reading the output by `[y,delta]=polyval(p,x,S,mu);` where `y`$\pm$`delta` gives the error supposedly within 50% confidence.

7. `[U,V,D]=svd(x,0);` produces the **singular value decomposition (SVD)** of the $n \times m$ array `x` where `U` is an $n \times n$ orthogonal array (i.e., transpose is the inverse), `V` is an $m \times m$ orthogonal array and `D` is generally a $n \times m$ diagonal array, but the second `0` input option reduces it to the necessary $m \times m$, when $n \geq m$, but in our notation $n \geq m' + 1$ with the added constant column, so for `svd` then `m=m'+1`.[a] SVD used to be one of a set of methods used by the backslash method.

[a] See D. & N. Higham '05, MATLAB Guide, pp.130-131. In general, the least squares problem is susceptible to ill-conditioning due to possible problems in catastrophic cancellation, but SVD was developed as a robust method for this problem. See also G. Forsythe, M. Malcolm and C. Moler, *Computer Methods for Mathematical Computations*, 1977, for pre-MATLAB background by Moler and his mentor Forsythe. ***Note, mathematical infinite precision is unrealistic compared to the computational finite precision in practice.***

5.4 Weighted Least Squares :

Since **outliers or extreme rare events happen**, there may be a desirable to **reduce their bias if the objective is equilibrium type models** or if the interest is the larger value at risk the analyst may want to enhance the effect to **compensate for the bias of regression to average out the outliers**.

Let $\boldsymbol{W}=[w_i\delta_{i,j}]_{n\times n}$ be a diagonal matrix of constant positive weights w_i, so $\boldsymbol{W^{-1}}=[\delta_{i,j}/w_i]_{n\times n}$, $\boldsymbol{W^{1/2}}=\sqrt{\boldsymbol{W}}=[\sqrt{w_i}\delta_{i,j}]_{n\times n}$ and $\boldsymbol{W}^\top=\boldsymbol{W}$. Next, the observational error or residual $\vec{e}$ is replaced by a weighted version $\vec{e}_w=\vec{e}/\sqrt{\boldsymbol{W}}$, so that the basic statistics are $\mathbf{E}[\vec{e}_w]=\vec{0}_n$ and $\mathbf{Cov}[\vec{e}_w,\vec{e}_w^\top]=\sigma_e^2 I_n W^{-1}$.

The residual is then

$$\vec{\mathbf{Res}} = \vec{Y} - A\vec{p}_{\mathbf{ex}} = \vec{e}/\sqrt{W} \tag{5.29}$$

with extended observations array $A = [\vec{1}, \ \vec{X}]$ and the residual sum of squares is

$$\mathbf{RSS}(\vec{p}_{\mathbf{ex}}) = \vec{e}^{\top}\vec{e} = (\vec{Y} - A\vec{p}_{\mathbf{ex}})^{\top} W (\vec{Y} - A\vec{p}_{\mathbf{ex}}), \tag{5.30}$$

so the theoretical least squares estimate is

$$\widehat{p}_{\mathbf{ex}} = \left(A^{\top} W A\right)^{-1} A^{\top} W \vec{Y}. \tag{5.31}$$

The implementation of **weighted least square (WLS)**, see **`MATLAB's robustfit`**, begins with scaling the predictor observations by letting

$$M \equiv \sqrt{W}\, A = \left[\sqrt{w_i} X_{i,j-1}\right]_{n\times(m+1)}, \tag{5.32}$$

recalling that $\vec{X}_0 = \vec{1}$, and the response observations by letting

$$\vec{Z} \equiv \sqrt{W}\vec{Y} = \left[\sqrt{w_i} Y_i\right]_{n\times 1}, \tag{5.33}$$

so the least squares parameter estimate becomes

$$\widehat{p}_{\mathbf{ex}} = \left(M^{\top} M\right)^{-1} M^{\top} \vec{Z}. \tag{5.34}$$

● ***Ordinary and Weighted Least Squares for Simple Linear Fit Example with Normal-Poisson Error:***

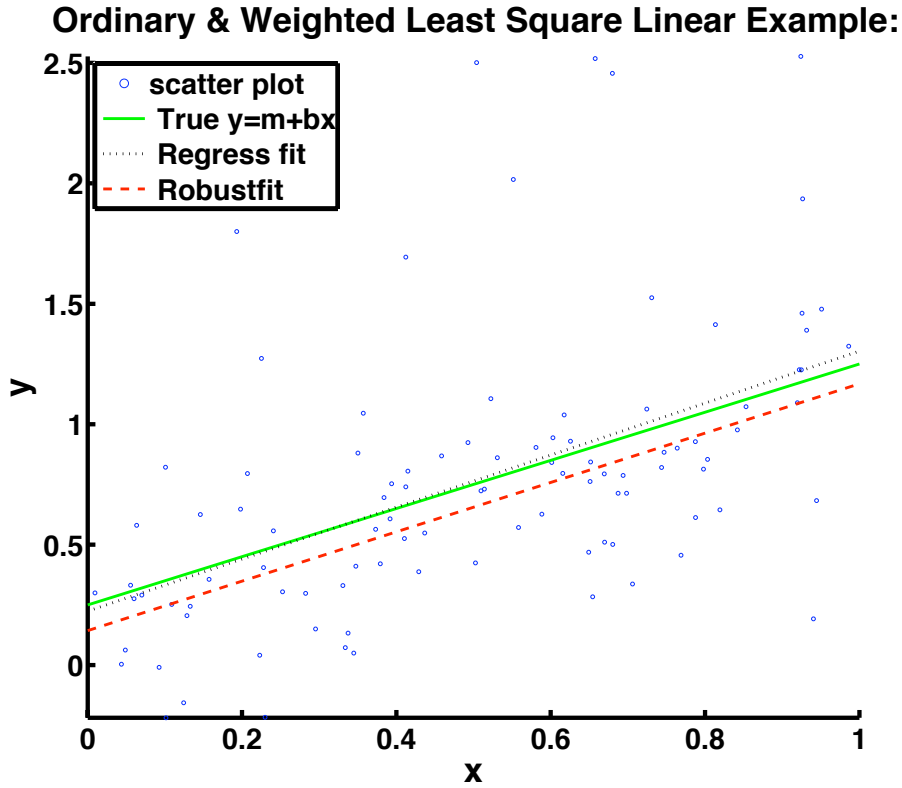

Figure 5.2: Ordinary least squares fit using **`MATLAB's regress`** and weighted least squares **`robustfit`** methods, with dissimilar results for a straight line fit with **`scatterplot`** data from combined normal and Poisson jump random error.

• *MATLAB Code for Ordinary & Weighted Least Squares Application with Normal-Poisson Error:*

```
function RegRobtest
clc
fprintf('\nRegRobtest Output (%s):',datestr(now));
m = 1; b = 0.25; % True parm values;
fprintf('\nTrue: b = %7.4f; m = %7.4f;',b,m);
ptrue = [b;m];
n = 100;
x = rand(n,1); % Simulated Uniform x-data;
sigma_e = 0.30; nu_e = 1.5; Lambda_e = 0.05;
% JD-Simulation Zero-Mean error:
err = sigma_e*randn(n,1) ...
    + nu_e*(poissrnd(Lambda_e,n,1)-Lambda_e*ones(n,1));
fprintf('\nsigma_e = %7.4f; nu_e = %7.4f; Lambda_e = %7.4f;'...
    ,sigma_e,nu_e,Lambda_e);
y = b+m*x+err; % Simulated y-data with linear model;
TSS = (n-1)*var(y);
fprintf('\nTSS = %9.3e',TSS);
A = [ones(n,1) x];
fprintf('\nsize(A)=[%i,%i]; size(y)=[%i,%i];',size(A),size(y));
[preg,pci,res,resci,statreg] = Regress(y,A); % Multlinear Regression;
breg = preg(1); mreg = preg(2);
yhatreg = breg+mreg*x;
```

```
fprintf('\nRelResReg =%9.3e;',sqrt(norm(y-yhatreg)/norm(y)));
[prob,statrob] = robustfit(x,y); % Weighted Least Sqs Method;
yhatrob = prob(1)+prob(2)*x;
fprintf('\nRelResRob =%9.3e;',sqrt(norm(y-yhatrob)/norm(y)));
fprintf('\nTrue:      b =%7.4f; m =%7.4f;',b,m);
fprintf('\nRegress:   b =%7.4f; m =%7.4f;',breg,mreg);
fprintf('\nRobustfit: b =%7.4f; m =%7.4f;',prob(1),prob(2));
fprintf('\nsqrt(norm(preg-ptrue)/norm(ptrue)) =%7.4f;' ...
    ,sqrt(norm(preg-ptrue)/norm(ptrue)));
fprintf('\nsqrt(norm(prob-ptrue)/norm(ptrue)) =%7.4f;' ...
    ,sqrt(norm(prob-ptrue)/norm(ptrue)));
fprintf('\nsqrt(norm(preg-prob)/norm(prob))   =%7.4f;' ...
    ,sqrt(norm(preg-prob)/norm(prob)));
SSEreg = sum((y-yhatreg).^2); % Also, RSS=ResSumSqs
SSErob = sum((y-yhatrob).^2); % Also, RSS=ResSumSqs
fprintf('\nRSS: SSEreg =%9.3e; SSErob =%9.3e;',SSEreg,SSErob);
Rsqreg = 1-SSEreg/TSS; Rsqrob = 1-SSErob/TSS;
fprintf('\nRsqreg =%9.3e; Rsqrob =%9.3e;',Rsqreg,Rsqrob);
fprintf('\nstatreg: R^2 =%7.4f; F = %7.4f;' ...
    ,statreg(1,1),statreg(1,2));
fprintf(' P-value)(F) =%7.4f; Var(error) =%7.4f;' ...
    ,statreg(1,1),statreg(1,1));
fprintf('\nstatrob Sigmas: OLS_s=%7.4f; Robust_s=%7.4f;' ...
    ,statrob.ols_s,statrob.robust_s);
```

```
fprintf(' MAD_s=%7.4f; final_s=%7.4f;' ...
    ,statrob.mad_s,statrob.s);
fprintf('\nstatrob: SE_p = [%7.4f; %7.4f];',statrob.se);
fprintf('\nstatrob: Corr_ps=[%7.4f, %7.4f; %7.4f, %7.4f];' ...
    ,statrob.coeffcorr);
statrob,
%
xg = 0:0.1:1;
ytrue = b+m*xg;
yhatregg = breg+mreg*xg;
yhatrobg = prob(1)+prob(2)*xg;
%
figure(1); nfig = 1;
scrsize = get(0,'ScreenSize'); % figure spacing for target screen
ss = [5.0,4.5,4.0,3.5]; % figure spacing factors
scatter(x,y,8); hold on;
plot(xg,ytrue,'-g',xg,yhatregg,':k',xg,yhatrobg,'--r','LineWidth',2);
axis tight; hold off;
title('Ordinary & Weighted Least Square Linear Example:'...
    ,'Fontsize',24,'FontWeight','Bold');
xlabel('x','Fontsize',24,'FontWeight','Bold');
ylabel('y','Fontsize',24,'FontWeight','Bold');
legend('scatter plot',' True y=m+bx',' Regress fit',' Robustfit' ...
    ,'Location','NorthWest');
```

```
set(gca,'Fontsize',20,'FontWeight','Bold','LineWidth',3);
set(gcf,'Color','White','Position' ...
   ,[scrsize(3)/ss(nfig) 60 scrsize(3)*0.60 scrsize(4)*0.80]); %[l,b,w,h]
fprintf('\n ');
========== CORRECTED Output ====================
RegRobtest Output (30-Jan-2010 14:04:30):
b =  0.2500; m =  1.0000;
sigma_e =  0.3000; nu_e =  1.5000; Lambda_e =  0.0500;
TSS = 3.247e+01
size(A)=[100,2]; size(y)=[100,1];
RelResReg =7.140e-01;
RelResRob =7.229e-01;
True:      b = 0.2500; m = 1.0000;
Regress:   b = 0.2241; m = 1.0794;
Robustfit: b = 0.1434; m = 1.0245;
sqrt(norm(preg-ptrue)/norm(ptrue)) = 0.2846;
sqrt(norm(prob-ptrue)/norm(ptrue)) = 0.3258;
sqrt(norm(preg-prob)/norm(prob))   = 0.3072;
RSS: SSEreg =2.372e+01; SSErob =2.492e+01;
Rsqreg =2.695e-01; Rsqrob =2.326e-01;
statreg: R^2 = 0.2695; F = 36.1469; P-value)(F) = 0.2695;
Var(error) = 0.2695;
statrob Sigmas: OLS_s= 0.4920; Robust_s= 0.3220; MAD_s= 0.2732;
final_s= 0.3302;
```

```
statrob: SE_p = [ 0.0690;  0.1205];
statrob: Corr_ps=[ 1.0000, -0.8780; -0.8780,  1.0000];
statrob =
        ols_s: 0.4920
     robust_s: 0.3220
        mad_s: 0.2732
            s: 0.3302
        resid: [100x1 double]
        rstud: [100x1 double]
           se: [2x1 double]
         covb: [2x2 double]
    coeffcorr: [2x2 double]
            t: [2x1 double]
            p: [2x1 double]
            w: [100x1 double]
            R: [2x2 double]
          dfe: 98
            h: [100x1 double]
 >>
```

Remark: Some differences with reasonable intercept (b) but poor slope (m), the parameter RMS difference being 0.3072. The OLS-sigma is much greater than the Robust-sigma or Final-sigma.

5.5 General Linear Least Squares (GLLS) Method for General Linear Model (GLM):

The use of only linear predictor variables is highly restrictive and unrealistic, since the model should depend on the available science models and not on the modeler's limited toolbox of models. In the **general least squares method** a **general linear model** is assumed that the response variable Y is a **linear combination of functions** of the predictor variable $\vec{X}$ that includes linear and nonlinear ones, but the model must be a linear in the parameters, $\vec{c} = [c_i]_{p\times 1}$, which happen to be the coefficients,

$$y = \sum_{k=1}^{p} c_k \phi_k(\vec{x}), \tag{5.35}$$

where the $\phi_k(\vec{x})$ are the general nonlinear fit functions. An example is $\phi_k(\vec{x}) = 1$, the constant function, $\phi_k(\vec{x}) = x_i$, the linear in component x_i, or $\phi_k(\vec{x}) = x_i x_j$, the interactive function if $i \neq j$ and pure quadratic if $i = j$.

Unfortunately, this excludes parameters in the function like $\phi_k(\vec{x};\vec{\beta})$, such as when trying to fit multi-parameter densities where the parameters cannot be formed into a linear combinations coefficient.

Hence, we need to fit the response-predictor data to determine a **coefficient parameter** p-vector $\boldsymbol{c}$ with respect to the corresponding **fit basis function** p-vector $\vec{\Phi}(\vec{x}) = [\phi_i(\vec{x})]_{p\times 1}$. The **objective is to minimize the GLLS chi-square function**:

$$\chi^2(\vec{c}) = \sum_{i=1}^{n} \left(Y_i - \sum_{k=1}^{p} c_k \phi_k(\vec{X}_i) \right)^2 \Big/ \sigma_i^2, \tag{5.36}$$

where the $w_i = 1/\sigma_i^2$ are the **weight constants or functions** for $i = 1:m$. For simplicity, it is assumed that $\sigma_i^2 = \sigma_{y,i}^2$ is a variance associated with the response variable Y_i, so that for the time being, $\sigma_{x,i}^2 \equiv 0$, i.e., we have precise $\vec{X}_i$ or $\delta\vec{X}_i \equiv 0$.

Here, the array

$$A = [A_{i,j}]_{n\times p} \equiv \sqrt{W}\,\Phi^{\top}(X) \equiv \left[\phi_j(\vec{X}_i)/\sigma_i\right]_{n\times p}, \qquad (5.37)$$

is called the **design matrix**, where $\vec{X}_i = [X_{i,j}]_{1\times m}$, m is the number of predictor variables, for $i = 1:n$ predictor observations, while $X = [X_{i,j}]_{n\times m}$ is the array of all predictor observations and $W = [w_i\delta_{i,j}]_{n\times n}$ is the diagonal weight matrix. Further, let $\vec{Y} = [Y_i]_{n\times 1}$ is the vector of observed responses with assigned variance vector $\vec{\sigma} \equiv [\sigma_i]_{n\times 1}$.

It is reasonable to assume that $n \gg p$, that there is much more data then there are parameters, so that the system of equations in severely over-determined and an averaged solution is necessary. The more over-determined the system is, the better it is for the statistics. The number of **degrees of freedom (DOF)** is `DOF=n-p>0`. Let the scaled output be $\vec{Z} = [Z_i]_{n\times 1} \equiv [Y_i/\sigma_i]_{n\times 1} \equiv \sqrt{W}[Y_i]_{n\times 1}$.

Thus, Eq. (5.54) can be written in matrix-vector form,

$$\chi^2(\vec{c}) = (\vec{Z} - A\vec{c})^\top (\vec{Z} - A\vec{c})^\top. \tag{5.38}$$

The optimal critical condition for the coefficient parameters is

$$\vec{0}_p \stackrel{*}{=} \nabla_c[\chi^2](\vec{c}^*) = -2(A^\top \vec{Z} - A^\top A\vec{c}^*), \tag{5.39}$$

yielding the least squares **normal equation** for the parameter vector

$$A^\top A\vec{c}^* = A^\top \vec{Z} \tag{5.40}$$

and the estimate,

$$\widehat{c} = \vec{c}^* = (A^\top A)^{-1} A^\top \vec{Z}. \tag{5.41}$$

This is essentially the same form obtained for either ordinary or weighted least squares regression. In fact, there is an example in the **MATLAB help** for the OLS **regress** function,

```
A=[ones(size(x1)),x1,x2,x1.*x2],
```
(5.42)

that show how you can include simple GLLS basis functions into OLS variables. In a sense, the raw data $[x1, x2]$ is formed into substitute vectors, as long as still independent,

```
[X1,X2,X3,X4]=[ones(size(x1)),x1,x2,x1.*x2].
```
(5.43)

• ***Singular Value Decomposition (SVD):***

Although, the product matrix $\boldsymbol{A^\top A}$ is symmetric and is usually invertible, it contains no more information than $\boldsymbol{A}$ itself and it is desirable to work just with $\boldsymbol{A}$. The singular value decomposition are an ideal way of doing this for the least squares design matrix $\boldsymbol{A}$. SVD for $\boldsymbol{A}$ has the form,

$$\boldsymbol{A = UDV^\top}, \tag{5.44}$$

where $\boldsymbol{U = [U_{i,j}]_{n\times n}}$ is the same size as $\boldsymbol{A}$ and is **column othogonal**, $\boldsymbol{U^\top U = I_n}$ or $\boldsymbol{\sum_{k=1}^n U_{k,i}U_{k,j} = \delta_{i,j} = \vec{U}_i^{(col)}\cdot\vec{U}_j^{(col)}}$, $\boldsymbol{V = [V_{i,j}]_{p\times p}}$ is square and row or column orthogonal, $\boldsymbol{V^\top V = I_p = VV^\top}$, and $\boldsymbol{D = [s_i\delta_{i,j}]_{n\times p}}$ is a diagonal matrix with **singular values** $\boldsymbol{s_i}$ along the diagonal and measuring the effect of $\boldsymbol{A}$ on a vector with respect to a norm, such that $\boldsymbol{\min_i(s_i) \le ||A\vec{u}||/||\vec{u}|| \le \max_i(s_i)}$, $\boldsymbol{||\vec{u}|| \neq 0}$. Thus,

$$\boldsymbol{D = I_nDI_n = U^\top UDV^\top V = U^\top AV.} \tag{5.45}$$

The **SV problem (SVP)** is $\boldsymbol{AV = sU}$. See **`MATLAB's svd`** function.

For GLLS, if $\boldsymbol{A}$ has an **SVD**, i.e., $\boldsymbol{A}=\boldsymbol{UDV}^{\top}$, then using the transpose reversed product algebra, i.e., $(\boldsymbol{AB})^{\top}=\boldsymbol{B}^{\top}\boldsymbol{A}^{\top}$,

$$\boldsymbol{A}^{\top}=\left(\boldsymbol{UDV}^{\top}\right)^{\top}=\left(\boldsymbol{V}^{\top}\right)^{\top}\boldsymbol{D}^{\top}\boldsymbol{U}^{\top}=\boldsymbol{VD}^{\top}\boldsymbol{U}^{\top}, \tag{5.46}$$

since the transpose of transpose does nothing to a matrix and noting that a non-square the diagonal is not symmetric so $\boldsymbol{D}^{\top}\neq\boldsymbol{D}$. The desired product $\boldsymbol{A}^{\top}\boldsymbol{A}$ decomposes as

$$\boldsymbol{A}^{\top}\boldsymbol{A}=\boldsymbol{VD}^{\top}\boldsymbol{U}^{\top}\boldsymbol{UDV}^{\top}=\boldsymbol{VD}^{\top}\boldsymbol{I}_{n}\boldsymbol{DV}^{\top}=\boldsymbol{VD}^{\top}\boldsymbol{DV}^{\top}, \tag{5.47}$$

where $\boldsymbol{D}^{\top}\boldsymbol{D}=[\boldsymbol{s}_{i}^{2}\boldsymbol{\delta}_{i,j}]_{p\times p}$, which is square and genuinely symmetric. Next, applying the SVD (5.46) and (5.47) to the normal equation (5.40) for the parameter vector,

$$\boldsymbol{VD}^{\top}\boldsymbol{DV}^{\top}\vec{\boldsymbol{c}}^{*}=\boldsymbol{VD}^{\top}\boldsymbol{U}^{\top}\vec{\boldsymbol{Z}} \tag{5.48}$$

and inverting the coefficients $\boldsymbol{V}$, $(\boldsymbol{D}^{\top}\boldsymbol{D})$ and $\boldsymbol{V}^{\top}$ of $\vec{\boldsymbol{c}}^{*}$ one by one yields

$$\widehat{\boldsymbol{c}}=\vec{\boldsymbol{c}}^{*}=\boldsymbol{V}\widehat{\boldsymbol{D}}\boldsymbol{U}^{\top}\vec{\boldsymbol{Z}}, \tag{5.49}$$

where $\widehat{\boldsymbol{D}}=(\boldsymbol{D}^{\top}\boldsymbol{D})^{-1}\boldsymbol{D}^{\top}=[\boldsymbol{\delta}_{i,j}/\boldsymbol{s}_{i}]_{p\times n}$, would be "$\boldsymbol{D}^{-1}$" if invertible, but $\boldsymbol{D}$ is almost always not square.

• ***SVD Parameter Solution:***

Hence, the least squares solution is, given SVD matrices for $\boldsymbol{A}$, is reduced to matrix-vector multiplication, a big advantage for a matrix computational system like **MATLAB**. However, by expanding by matrix elements the multiplications can be reduced to partial matrix multiplication:

$$\begin{aligned}\vec{c}^{*} = [c_i^*]_{p\times 1} &= \left[\sum_{j=1}^{p} V_{i,j} \sum_{k=1}^{p} \frac{1}{s_j}\delta_{j,k} \sum_{\ell=1}^{n} U_{k,\ell} Z_\ell\right]_{p\times 1} \\ &= \left[\sum_{j=1}^{p} V_{i,j} \frac{1}{s_j} \sum_{\ell=1}^{n} U_{j,\ell} Z_\ell\right]_{p\times 1} \\ &= \sum_{j=1}^{p} \frac{1}{s_j}\left(\left(\vec{U}_j^{(col)}\right)^{\top}\vec{Z}\right)\vec{V}_j^{(col)},\end{aligned} \tag{5.50}$$

where the column vectors are $\vec{U}_j^{(col)} \equiv [U_{i,j}]_{p\times 1}$ and $\vec{V}_j^{(col)} \equiv [V_{i,j}]_{n\times 1}$. Note that $((\vec{U}_j^{(col)})^{\top}\vec{Z})$ is a scalar product, so the vector-orientation of $\vec{c}^{*}$ comes mainly from $\vec{V}_j^{(col)}$.

• ***SVD Parameter Sensitivities:***

The vector-form of SVD parameter solution is also useful for determining the parameter sensitivities to the output responses, e.g., the sensitivity of parameter c_j^* to the original response $Y_i = \sigma_i Z_i$ is

$$\frac{\partial c_j^*}{\partial Y_i} = \sum_{k=1}^{p} \frac{1}{s_k} \left(\left(\vec{U}_k^{(col)} \right)^\top \frac{\partial \vec{Z}}{\partial Y_i} \right) V_{j,k} = \frac{1}{\sigma_i} \sum_{k=1}^{p} \frac{1}{s_k} U_{i,k} V_{j,k}. \quad (5.51)$$

This sensitivity can be used to estimate the variance of c_j^*, under restored response weighting,

$$\begin{aligned} \widehat{\sigma}^2_{c_j^*} &= \sum_{i=1}^{n} \widehat{\sigma}_i^2 \left(\frac{\partial c_j^*}{\partial Y_i} \right)^2 = \sum_{i=1}^{n} \sum_{k=1}^{p} \frac{1}{s_k} U_{i,k} V_{j,k} \sum_{\ell=1}^{p} \frac{1}{s_\ell} U_{i,\ell} V_{j,\ell} \\ &= \sum_{k=1}^{p} \frac{1}{s_k^2} V_{j,k}^2, \end{aligned} \quad (5.52)$$

using the column othogonality $\sum_{i=1}^{n} U_{i,k} U_{i,\ell} = \delta_{\ell,k}$. Similarly, the estimated covariance is $\widehat{\mathrm{cov}}(c_i^*, c_j^*) = \sum_{k=1}^{p} V_{i,k} V_{j,k} / s_k^2$.

• ***Fitting General Nonlinear Models (GNLM):***

Given a general nonlinear model $y = f(x;\vec{c})$, beyond the linear combination form $y = \sum_{j=1}^{p} c_j \phi_j(x)$, the primary technique is to transform the GNLM, if possible, to a form suitable for least squares. Due to complexity, the GNLM can be numerically ill-conditioned, i.e., difficult to compute. Some examples:

1. **Exponential-like Density**: $y = f(x;\vec{c}) = c_2 \exp(-c_1 x)$. Let $z = \ln(y) = -c_1 x + \ln(c_2)$, so letting $z = \sum_{j=1}^{2} b_j \phi_j(x)$ with $\phi_1(x) = 1$, $\phi_2(x) = x$, $b_1 = \ln(c_2)$ and $b_2 = -c_1$. Problem is that for the single parameter exponential distribution $c_1 = c_2 = 1/\mu$, so least square errors are likely to lead to inconsistencies.
2. **Normal-like Density**: $y = f(x;\vec{c}) = \dfrac{\exp(-(x-c_1)^2/(2c_2))}{\sqrt{2\pi c_2}}$. Let $z = \ln(y) = -(x-c_1)^2/(2c_2) - \ln(\sqrt{2\pi c_2})$, so letting $z = \sum_{j=1}^{3} b_j \phi_j(x)$ with $\phi_1(x) = 1$, $\phi_2(x) = x$, $\phi_3(x) = x^2$, $b_1 = -c_1^2/(2c_2) - \ln(\sqrt{2\pi c_2})$, $b_2 = c_1/c_2$ and $b_3 = -1/(2c_2)$. Problem is that for the two parameter normal distribution leads to three related coefficients and inconsistencies.

- ***General Nonlinear Least Square Fit Methods:***

Let the nonlinear objective be

$$\chi^2(\vec{c}) = \chi^2(\vec{X}, \vec{Y}, \vec{\sigma}; \vec{c}) = \sum_{i=1}^{n} \frac{(Y_i - f(X_i; \vec{c}))^2}{\sigma_i^2}. \tag{5.53}$$

Searching for critical points in p-dimensions,

$$\frac{\partial \chi^2}{\partial c_j}(\vec{c}) = -2 \sum_{i=1}^{n} \frac{(Y_i - f(X_i; \vec{c}))}{\sigma_i^2} \frac{\partial f}{\partial c_j}(X_i; \vec{c}) \stackrel{*}{=} 0 \tag{5.54}$$

for $j = 1:p$ and where $\vec{c}^* = \min_{\vec{c}} [\chi^2(\vec{c})]$. Difficulties are there is no closed-form linear normal equation there is no SVD or similar methods and the optimization may be ill-conditioned with local minima hiding a global minima.

- ***General Nonlinear Least Square Fit Functions in MATLAB:***

The nonlinear least squares is the kind of method needed to find the market distribution for assets or other financial instruments. Derivative optimization methods like methods of steepest descents and variations of Newton's method may be time-consuming and for complicated distribution models that derivative formulas may be quite complex. For general users, it may be desirable to use direct methods which by definition do not use derivatives or derivative methods for which the code asks only for an input of the function χ^2 and an initial value of the parameter $\vec{c}_0$. Some of these in **MATLAB**, listed in the simplest or next to simplest form:

1. `c=fminsearch(chi2,c0)` : The `chi2` is a function handle for the objective function assumed to include the nonlinear fit function `f`, `c0` is an input starting vector value and `c` is the output answer. This is the granddaddy of direct search and uses **Nelder-Mead's (1994) down-hill simplex method** which employs flexible super-triangles in the parameter space to search for unconstrained minima. It is fast and fairly robust, but is is just a basic minimization method. Related function `fminunc` is an unconstrained derivative minimum search method in the `Optimization Toolbox`. Functions `fminbnd` and `fmincon` treat constrained minima, the first by direct search and the second by derivative search. See `MATLAB help`for more information.

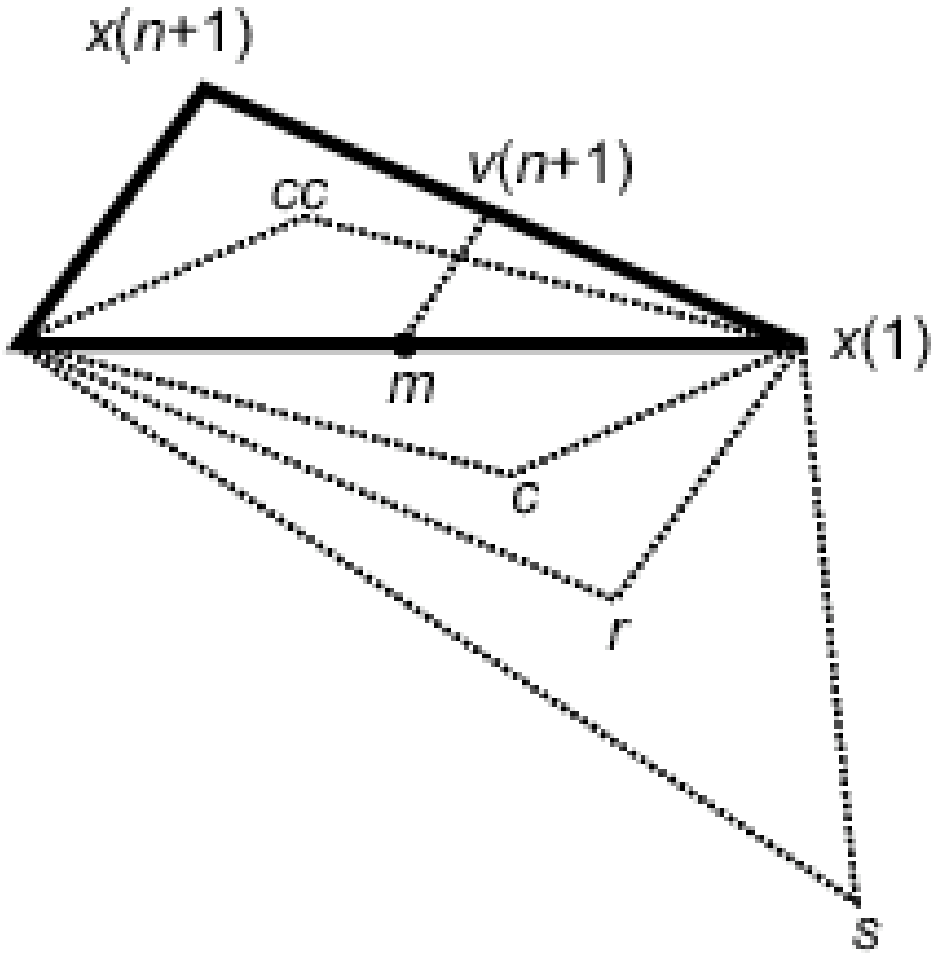

Figure 5.3: MATLAB Down-Hill Simplex Algorithm Illustration: *Fminsearch Algorithm, Unconstrained Nonlinear Optimization page, Optimization Toolbox, 2008*. Triangular with $(x(1), x(2), x(n+1) \approx x(3))$.

2. `[c,Res,Cov,mse]=nlinfit(X,Y,f,c0)`: This is a **nonlinear regression function**, where `X` is the input predictor, `Y` is the input response, `f` is the function handle for the nonlinear fit function, `c0` is an input starting vector value and `c` is the output answer. Extra output is residual `Res`. Jacobian `J`, covariance matrix `Cov` and an error term `mse`. It is similar to `robustfit` multilinear weighted least squares, but `nlinfit` also uses the Jacobian of derivatives fot `f`. Also, `nlinfit` has several auxiliary functions and a `GUI` tool:

   - `cci=nlparci(c,Res,'covar',Cov)`:
     This gives 95% confidence intervals `cci` for the output parameter `c,Res,Cov` of `nlinfit`, else insert the input option pair {`,'alpha',alpha`}. The `'covar'` is the parameter label for `Cov`.
   - `[ypred,delta]=nlpredci(f,X,c,Res,'covar',Cov)`:
     This gives predicted values `ypred` for 95% CI half-widths `delta`. It also uses the output parameters `c,Res,Cov` of `nlinfit`.

- `nlintool(X,Y,f,c0,alpha,'Xname','Yname')`: This is the graphical user interface for `nlinfit`, can take optional arguments like the complementary CI parameter `alpha` and plot XY-labels `'Xname','Yname'`.

5.6 Maximum Likelihood Estimation Method:

Maximum likelihood estimation methods are a form of regression where the objective is derived from maximum of the likelihood, i.e., the most probable state of the distribution. The maximum probability occurs at the maximum of the density, called the **mode**. For example, the normal distribution density is

$$f_Z^{(n)}(z;\mu,\sigma^2)=\exp(-(z-\mu)^2/(2\sigma^2))/\sqrt{2\pi\sigma^2} \qquad (5.55)$$

and checking for critical points,

$$0\stackrel{*}{=}(f_Z^{(n)})'(z^*;\mu,\sigma^2)=-\frac{2(z^*-\mu)}{\sigma^2}f_Z^{(n)}(z^*;\mu,\sigma^2), \qquad (5.56)$$

we find the critical point and mode to be at the mean $z^*=\mu$ since that can be the only interior critical point, the density only vanishing at infinity. Note that we effectively computed the logarithmic derivative $d\ln(f_Z^{(n)})/dz=(f_Z^{(n)})'/f_Z^{(n)}$. Often, the log-likelihood using the logarithm of the density rather than the density itself for the maximum likelihood to avoid the over-dominance of exponentials.

Often we are looking for a multidimensional mode (the value of the variate where the maximum probability occurs) and in analogy with the normal density and it is easier to look at the exponent by using the logarithm than at the exponential itself. Since if the kernel or most important part of the distribution is of the form $\boldsymbol{K(x) = \exp(\phi(x))}$ then $\boldsymbol{\ln(K(x)) = \phi(x)}$ and

$$\boldsymbol{K'(x) = \phi'(x)K(x) = (\log(K))'(x)K(x),} \qquad (5.57)$$

so the determination of the maximum likelihood location or mode by critical point analysis of this simple kernel is related to the simpler critical point analysis of the logarithm of the kernel, i.e., zeros of the logarithmic derivative of the kernel, assumed unimodal.

- ***A Probabilistic Introduction to Maximum Log-Likelihood (MLE) for Financial, Exploratory Data Analysis Applications:***[a]

Let us return to the **geometric Brownian motion** or linear diffusion with constant coefficients model for an asset price $A(t)$, where

$$dA(t) = A(t)(\mu dt + \sigma dW(t)), \tag{5.58}$$

where the statistics (μ, σ) are constant and $W(t)$ is the theoretical model for Brownian motion. Since the equation is linear in the asset price, it is convenient to transform to log-pricing using the stochastic dt precision chain rule to to get the **arithmetic Brownian motion**,

$$d\log(A(t)) = (\mu - \sigma^2/2)dt + \sigma dW(t). \tag{5.59}$$

Further, since our financial data is discrete and assuming the time-step Δt is sufficiently small, the equation is converted to **asset log-return form**,

$$\mathrm{LR}_i = \Delta \log(A_i) = m_\ell + \sqrt{v_\ell} Z_i \tag{5.60}$$

where, simplifying, $m_\ell \equiv (\mu - \sigma^2/2)\Delta t$ and $v_\ell \equiv \sigma^2 \Delta t > 0$ are log-coefficients from using $\Delta W_i = \sqrt{\Delta t} Z_i$ with $Z_i \overset{\text{dist}}{\underset{\text{IID}}{=}} \mathcal{N}(0,1) \; \forall \, i$.

[a] See Carmona '04, p. 124ff; Hull '06, p. 567ff; Hanson 01'-08' computational finance papers, ***http://www.math.uic.edu/ hanson/compfinpapers.html***.

The **log-return distribution** for each i, with $X = \mathrm{LR}_i$, is given by definition,

$$\begin{aligned} F_X(x_i) &\equiv \mathrm{Prob}[X \le x_i] = \mathrm{Prob}[m_\ell + \sqrt{v_\ell} Z_i \le x_i] \\ &\overset{\mathrm{alg}}{=} \mathrm{Prob}[Z_i \le (x_i - m_\ell)/\sqrt{v_\ell}] \\ &\overset{\mathcal{N}}{=} F_Z^{(\mathrm{n})}((x_i - m_\ell)/\sqrt{v_\ell}; 0, 1)\,. \end{aligned} \tag{5.61}$$

Thus, upon differentiation, the ith **likelihood function** is the density function,

$$\begin{aligned} \mathrm{LH}_i(m_\ell, v_\ell) = f_X(x_i) &= \frac{1}{\sqrt{v_\ell}} f_Z^{(\mathrm{n})}((x_i - m_\ell)/\sqrt{v_\ell}; 0, 1) \\ &\overset{\mathcal{N}}{=} \frac{1}{\sqrt{2\pi v_\ell}} \exp\left(-0.5(x_i - m_\ell)^2/v_\ell\right)\,. \end{aligned} \tag{5.62}$$

Further, the Z_i are **IID normal**, so the total density is the product of all the individual density for the log-return data count for $i = 1:n$.

Hence,

$$\begin{aligned}\mathbf{LH^{(n)}}(m_\ell, v_\ell) &= f_{\vec{X}}(\vec{x}) \overset{\text{IID}}{=} \prod_{i=1}^{n} f_X(x_i) \\ &= \frac{1}{(\sqrt{v_\ell})^n} \prod_{i=1}^{n} f_Z^{(n)}((x_i - m_\ell)/v_\ell; 0, 1) \\ &\overset{\mathcal{N}}{=} \frac{1}{(\sqrt{2\pi v_\ell})^n} \prod_{i=1}^{n} \exp\left(-0.5(x_i - m_\ell)^2/v_\ell\right).\end{aligned} \tag{5.63}$$

For further simplicity, we take logarithms, turning the products into sums, to get the log-likelihood function (the natural logarithm, here $\log \Leftrightarrow \ln$),

$$\begin{aligned}\mathbf{LLH^{(n)}}(m_\ell, v_\ell) &= \log\left(f_{\vec{X}}(\vec{x})\right) = \sum_{i=1}^{n} \log(f_X(x_i)) \\ &= -\frac{1}{2v_\ell} \sum_{i=1}^{n} (x_i - m_\ell)^2 - \frac{n}{2} \log(2\pi v_\ell),\end{aligned} \tag{5.64}$$

ending up with the negative of the least squares objective. Although often the log-likelihood function is multiplied by a minus sign since most optimal solvers are written as minimizers,

The system bias in linear regression is that they are natural for normally distributed randomness. There are several variants of maximum likelihood estimation based on other distributions, e.g., multinomial maximum likelihood estimation[a] and **MATLAB**'s generic maximum likelihood function **mle** specialized to 17 named distribution options using the response data. Seeking critical points,

$$\frac{\partial \mathrm{LLH}^{(n)}}{\partial m_\ell}(m_\ell, v_\ell) = \frac{1}{v_\ell}\sum_{i=1}^{n}(x_i - m_\ell) \stackrel{*}{=} 0, \tag{5.65}$$

and

$$\frac{\partial \mathrm{LLH}^{(n)}}{\partial v_\ell}(m_\ell, v_\ell) = \frac{1}{2v_\ell^2}\sum_{i=1}^{n}(x_i - m_\ell)^2 - \frac{n}{2v_\ell} \stackrel{*}{=} 0, \tag{5.66}$$

gives the simultaneous estimates,

$$\widehat{m}_\ell = m_\ell^* = \frac{1}{n}\sum_{i=1}^{n} x_i \equiv \overline{x} \;\;\&\;\; \widehat{v}_\ell = v_\ell^* = \frac{1}{n}\sum_{i=1}^{n}(x_i - m_\ell^*)^2 \equiv \widehat{\sigma}_x^2. \tag{5.67}$$

This justifies using the mean and variance directly from the log-return data.

[a]For instance, Hanson with Westman and Zhu 04a' and 04b', computational finance papers, *http://www.math.uic.edu/ hanson/compfinpapers.html*.

This is essentially what one would expect and could have been obtained from `mean(LR)` and `var(LR)`. Often, practitioners may ***a prior*** set $m_\ell = 0$, assuming that the mean log-return is usually small anyway. Finally, we need to convert back to standard model coefficient, assuming that Δt, usually one trading day in years, is known,

$$\widehat{\sigma} = \frac{\widehat{\sigma}_x}{\sqrt{\Delta t}} \quad \& \quad \widehat{\mu} = \frac{\overline{x} + \widehat{\sigma}_x^2/2}{\Delta t}, \tag{5.68}$$

the latter form seems to contradict the practice of throwing out the mean m_ℓ since $\widehat{\sigma}_x^2/2 > 0$.

One **advantage of maximum likelihood estimation (MLE)**, it that all problems are not forced into one template objective such as least squares for the normal noise model, but yield an objective that is more natural for the target problem and its distribution.

(not presented)

• ***Using Subsamples to Get Some Estimate of Time Dependence of Model Coefficients:***

If the sample size it sample size n is sufficiently large, then subsamples can be used to get an estimate of time dependence of the model coefficients. Since the sample index i is associated with a trading day, picking $i = k$ for some trading day t_i, marking a log-return between i and $i+1$, and then a half bandwidth iw, then our MLE estimates would be

$$\widehat{m}_{\ell,k} = \frac{1}{n} \sum_{i=k-iw}^{k+iw} x_i \equiv \overline{x}_k \tag{5.69}$$

and

$$\widehat{v}_{\ell,k} = \frac{1}{n} \sum_{i=k-iw}^{k+iw} (x_i - \widehat{m}_{\ell,k})^2 \equiv \widehat{\sigma}^2_{x,k}, \tag{5.70}$$

where $k - iw \geq 1$ and $k + iw \leq n$. The index k represents the center of a moving window in time.

(not presented)

- ***General Maximum Likelihood Function in MATLAB:***

`[chat,cci]=mle(Y,'distribution','DistName','alpha',alpha)`: This is a maximum likelihood estimator for a large number of specialized distributions. The required input is data `Y`, but the optional input is the `'distribution'` parameter paired with the name value `'DistName'` which can be `'bernoulli'`, `'bernoulli'`, `'exponential'`, `'generalized pareto'`, `'lognormal'`, `'normal'` (default), `'poisson'`, `'uniform'` and others. The output is the estimated parameters `chat` and the parameter confidence interval `cci` at complementary level `alpha`, but if pair `'alpha', alpha` is omitted then the CI is at the MATLAB default of `0.05` or 95% CI. Many of the `mle` special distributions are prepackaged, such as `binofit` or `poissfit`. The auxiliary function `mlecov` with similar arguments outputs the parameter estimate covariance matrix.

** Reminder: Lecture 5 Homework Posted in Chalk Assignments, due by Lecture 6 in Chalk Assignments!*

*** Summary of Lecture 5:**

1. **Ordinary Least Squares or Linear Regression**
2. **Multilinear Regression**
3. **Weighted Least Squares**
4. **General Linear Least Squares**
5. **Maximum Likelihood Estimation Method (part 1)**