Application of Matlab for Finance

October 21, 2017

Today's Class

- Maximum Likelihood Estimator (MLE)

Basic Linear Regression: OLS

$$Y = X\beta + \epsilon, \quad \epsilon \sim N(0, \sigma)$$

- Y: $T \times 1$ vector of dependent variables
- ant)
- $Y - X\beta)$
-
- Mean squared error (MSE)/variance of resid
- Goodness-of-fit against the benchmark: hist

$$R^2 = 1 - \frac{\sum_{i=1}^{T}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{T}(Y_i - \bar{Y})^2} = 1 - \frac{\hat{\epsilon}'\hat{\epsilon}}{(Y - \bar{Y})'(Y - \bar{Y})}$$

$$= 1 - \frac{Sum\ Squares\ of\ Residual}{Sum\ Squares\ of\ Total}, \qquad \bar{Y} = \frac{\sum_{i=1}^{T} Y_i}{T}$$

OLS Regression Code

- `b = regress(y,X)` estimates the M-by-1 β coefficient estimations of regression $y = X\beta$ and stored in output variable `b`.
-

  - `r`: T-by-1 matrix of the residual
  - `rint`: T-by-2 matrix of intervals that can be us
    outliers (if zero is not inside).
  - `stats`: 1-by-4 vector contains the R2 statistic, the F statistic and its p-value, and the estimate of the error variance (MSE).

Exercise 1: Basic Linear Regression

- Estimate the coefficients of the Fama-French Three-Factor model

- $_M$
- SMB is the size factor
- HML is the value factor

- r_M, SMB, HML are stored in sheet `F` `FF_Data.xlsx`.
- Individual portfolio returns are stored in the `IndusPort` worksheet.

Exercise 1: Fama French Model


```
% Estimate the model on S, V1
% alternative code of reading from excel file
[ff_fact, ff_txt]= xlsread('FF_Data.xlsx','Factors');
[ff_port, ff_port_txt]= xlsread('FF_Data.xlsx','IndusPort');

% A
x = f
r
%
y = f

% Regress with loop for each asset: the coefficients for each asset are
% stored in each row of beta
[T,K] = size(y);
for i = 1: K
    beta(i,:) = regress(y(:,i),x);
end

% perform the same regression: read beta estimations and the statistics
% of the model
for i = 1: K
   [beta(i,:),¬,¬,¬,stats(i,:)] = regress(y(:,i),x);
end
```

Linear Model Regression

`res = fitlm(X,y,modelspec)`

- `res`: the LinearModel Object for the responses `y` to be fit to data matrix `X`;
- ,

-
  - `res.Coefficients.pValue` re
    estimation

- The `fitlm` function assumes a constant ter
  - Original model `y ~ 1 + x1 + x2 + x3`
  - `terms = '1'`
  - `res = removeTerms(res,terms)`
  - New model `y ~ x1 + x2 + x3`

Exercise 1: Fama-French Model


```
%
y
r
t
res = removeTerms(res,terms)
beta_full = res.Coefficients.Estimate
```

Rolling and Recursive Regressions

- Static coefficients may fail to adjust changes in the economy

$$Y = X\beta + \epsilon, \quad \epsilon \sim N(0, \sigma)$$

-

- Timely adjusted coefficient shall reflect more u
- If coefficient change gradually, then similar coe time periods (seasonal effect)

- The rolling estimates is a combination of true coe sampling errors
  - True coefficient is trending: estimates display trend and noise
  - True coefficient is constant: estimates display random fluctuation and noise

Rolling and Recursive Regressions

- For a given window width τ, using the τ observations $(t = 1, \ldots, \tau)$ for estimation
-
-
- For a sample of observations k, it shall be $k - \tau + 1$ estimates of β

Exercises 2

- For the same portfolios in the previous exercise, c βs using 60 consecutive observations
- Do the coefficient for market exposure appear constant?

Exercise 2: Rolling Estimation


```
% Rolling Regressions for 1 asset
% Read Data Data
dd = ff_txt(2:end,1);
ddnum = datenum(dd,'dd/mm/yyyy');

% R

% D
t
%
f

    y_roll = y_cnsmr(t-tau:t);
    beta_roll(t,:)= regress(y_roll,x_roll);
end

% only plot the exposure to the market risk:
plotdd = ddnum(tau+1:end,1);
plotb_full = beta_full(1)*ones(length(plotdd),1);
plotb_roll = beta_roll(tau+1:end,1);
% Plot the data
plot(plotdd, plotb_full, plotdd, plotb_roll);
datetick('x');
xlim([plotdd(1), plotdd(end)]);
% Add a legend
legend('Constant \beta','Rolling \beta','Location','NorthWest');
```

Estimated Beta

Maximum Likelihood

- Maximum likelihood is a popular method to estimate parameters in econometric models. In many cases, closed form estimators are not available and so non-linear optimizers are required.

- Suppose $\mathbf{X} = (x_1, x_2, ..., x_n)$ are the samples taken from a random
$f(x ; \theta)$,

-
-
to the product of the marginal densities, which is al
likelihood function with respect to the paramet

$$L(\theta; X) = \prod_{i=1}^{n} f_i(x_i; \theta) = f_1(x$$

- The Maximum Likelihood Estimator (MLE) is the parameter set $\hat{\theta}$ that maximizes the likelihood function $L(\theta; X)$

$$L(\hat{\theta}; X) = \max_{\theta \in \Theta} L(\theta; X)$$

Log Maximum Likelihood

- It is often rather difficult to directly maximise the $L(\theta; X)$. It is much easier to maximise the log-likelihood function since $ln(.)$ is a monotonic function that maximising $L(\theta; X)$ is equivalent to maximising $\ln L(\theta; X)$

$$L(\theta; X) \quad \max \qquad \ln L(\theta; X) \quad \max$$

-

$i=$

- The crucial is to have explicit pdf function to maxi contains various options for different distribu

$$\ln L(\theta; X) = \sum_{i=1}^{n} (\ln \frac{1}{\sqrt{2\pi}} - \ln \sigma - \frac{(X-\mu)}{2\sigma^2})$$

$$= n \ln \frac{1}{\sqrt{2\pi}} - n \ln \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (X-\mu)^2$$

Log Maximum Likelihood: Exercises

- Use help to find the `mle` function in MATLAB
-
- sample

- Note: the `mle` function only allows one tim observations) at a time.

Exercise 3: MLE


```
% M
% c
r
p
%
p
% Display
disp('Consumption Industry Returns')
disp('MLE estimates of mu, sigma')
disp(parm_est)
disp('sample estimates of mu, sigma')
disp(parm_data
```

Extra: AR, MA and ARMA code

- Autoregressive Model: AR(p)

$$Y_t = c + \sum_{i=1}^{p} \varphi_i Y_{t-i} + \epsilon_t$$

-

t t i $t-i$ i

- ARMA(p,q) Model (set $\mu = 0$)

$$Y_t = c + \epsilon_t \sum_{i=1}^{p} \varphi_i Y_{t-i} + \sum_{i=1} \theta_i \epsilon_{t-i}$$

- If Y_t is integrated, use the ARIMA(p,D,q) model, where D is the number of difference order

Extra: AR, MA and ARMA code

- Code in matlab: `arima(p,D,q)`.
  - AR(1): `arima(1,0,0)`
  - MA(2): `arima(0,0,2)`
  - ARMA(2,3): `arima(2,0,3)`
-
-

Augmented Dicky-Fuller test `adftes`
presence of unit-root in the underlying return se

- `res = adftest(y)`
  - *res* = 0: fail to reject the null hypothesis of a unit r autoregressive alternative.
  - *res* = 1: reject the null hypothesis and conclude that the y is stationary.