

## Assignment 3

This highlights the use of linear regression and to examine the evidence for time-varying betas

### Contents

- Clear the workspace and close all figures
- Load the data
- Estimate the regressions
- Standard Errors and t-Stats
- Explanatory Power
- Recursive Plots
- Assessing the evidence

### Clear the workspace and close all figures

It is always a good idea to clear the workspace when starting a new project

```
clear all
close all
```

### Load the data

The data have been formatted in an Excel file and can be read using `xlsread`

```
FFData = xlsread('FF_1963_2009.xls');
% The factors are ordered Date VWMe SMB HML RF
FFDates = c2mdate(FFData(:,1));
VWMe = FFData(:,2);
SMB = FFData(:,3);
HML = FFData(:,4);
Rf = FFData(:,5);
% The returns are ordered SL SM SH BL BM BH. Note the use of excess returns
FFRets(:,1) = FFData(:,6) - Rf;
FFRets(:,2) = FFData(:,7) - Rf;
FFRets(:,3) = FFData(:,8) - Rf;
FFRets(:,4) = FFData(:,9) - Rf;
FFRets(:,5) = FFData(:,10) - Rf;
FFRets(:,6) = FFData(:,11) - Rf;
% Save as a mat file
save FF_1963_2009 FFDates VWMe SMB HML Rf FFRets
```

### Estimate the regressions

The first step is to estimate the regressions. The block of code uses cell arrays which are objects which can be used to store

arbitrary other arrays or matrices. These are initialized using the command `cell` (can also be initialized using `{}` as below). This code stores the output of `ols` in a cell array using `{}`.

```
% Setup the regressors
X = [VWMe SMB HML];
% Initialize cells to hold the output
beta = cell(6,1);
tstat = cell(6,1);
s2 = cell(6,1);
VCV = cell(6,1);
VCVWhite = cell(6,1);
R2 = cell(6,1);
% Loop over the series
for i=1:6
    [beta{i},tstat{i},s2{i},VCV{i},VCVWhite{i},R2{i}]=ols(FFRets(:,i),X,1);
end

% Construct some regular matrices from the cell arrays by storing the data in the cell arrays
in the
% columns of the output matrices
betaMatrix = zeros(4,6);
stdErrMatrix = zeros(4,6);
stdErrMatrixWhite = zeros(4,6);
tStatMatrix = zeros(4,6);
tStatMatrixWhite = zeros(4,6);
R2Matrix = zeros(1,6);
% Loop over the 6 series
for i=1:6
    betaMatrix(:,i) = beta{i};
    stdErrMatrix(:,i) = sqrt(diag(VCV{i}));
    stdErrMatrixWhite(:,i) = sqrt(diag(VCVWhite{i}));
    tStatMatrix(:,i) = beta{i}./stdErrMatrix(:,i);
    tStatMatrixWhite(:,i) = beta{i}./stdErrMatrixWhite(:,i);
    R2Matrix(i) = R2{i};
end
% mprint is a useful function for printing matrices. It requires an information structure
% containing the row and column names. See help mprint for more information
info.rnames = strvcat(' ','Intercept','VWMe','SMB','HML'); %#ok<*VCAT>
info.cnames = strvcat('SL','SM','SH','BL','BM','BH');
disp('      Betas');
mprint(betaMatrix,info)
```

Betas

	SL	SM	SH	BL	BM	BH
Intercept	-0.0018	0.0008	0.0007	0.0015	-0.0007	-0.0011
VWMe	1.0836	0.9638	1.0035	0.9831	1.0048	1.0632
SMB	1.0147	0.8070	0.8546	-0.1668	-0.1505	-0.0066
HML	-0.2636	0.3606	0.6849	-0.3064	0.3431	0.7451

## Standard Errors and t-Stats

mprint is again used to display the standard errors and t-stats. The heteroskedasticity robust t-statistics are generally smaller (so the standard errors are larger) although the significance does not depend on the covariance estimator used.

```
disp('      Standard Errors (Homo)');
mprint(stdErrMatrix,info)
disp('      Standard Errors (Hetero)');
mprint(stdErrMatrixWhite,info)
disp('      t-Stats (Homo)');
mprint(tStatMatrix,info)
disp('      t-Stats (Hetero)');
mprint(tStatMatrixWhite,info)
```

### Standard Errors (Homo)

	SL	SM	SH	BL	BM	BH
Intercept	0.0004	0.0004	0.0003	0.0004	0.0005	0.0004
VWMe	0.0101	0.0084	0.0065	0.0083	0.0120	0.0105
SMB	0.0139	0.0116	0.0089	0.0115	0.0166	0.0144
HML	0.0153	0.0127	0.0098	0.0126	0.0182	0.0158

### Standard Errors (Hetero)

	SL	SM	SH	BL	BM	BH
Intercept	0.0004	0.0003	0.0003	0.0003	0.0005	0.0004
VWMe	0.0105	0.0103	0.0092	0.0110	0.0152	0.0109
SMB	0.0177	0.0155	0.0147	0.0184	0.0221	0.0175
HML	0.0204	0.0242	0.0160	0.0180	0.0302	0.0206

### t-Stats (Homo)

	SL	SM	SH	BL	BM	BH
Intercept	-4.2734	2.1997	2.6327	4.1681	-1.3984	-2.4291
VWMe	107.5963	114.8811	155.2747	118.2396	83.4525	101.6778
SMB	72.9408	69.6355	95.7263	-14.5217	-9.0491	-0.4563
HML	-17.2757	28.3662	69.9454	-24.3251	18.8103	47.0330

### t-Stats (Hetero)

	SL	SM	SH	BL	BM	BH
Intercept	-4.4054	2.2743	2.6613	4.2705	-1.4138	-2.5165
VWMe	103.3262	93.8446	109.3014	89.3405	66.1057	97.6739
SMB	57.3167	52.1208	58.1985	-9.0705	-6.8133	-0.3773
HML	-12.8963	14.8776	42.7533	-17.0382	11.3558	36.1668

## Explanatory Power

These models explain a substantial amount of the cross-sectional variation (92-98%)

```
disp('      R-square');
```

```
info.rnames = strvcats(' ', 'R2'); %#ok<*VCAT>
mprint(R2Matrix,info);
```

	R-square					
	SL	SM	SH	BL	BM	BH
R2	0.9804	0.9775	0.9872	0.9708	0.9281	0.9526

## Recursive Plots

The recursive plots are the most difficult part of this assignment

```
% Save the X data as fullX
fullX = X;
% Get the size of the data
T = size(fullX ,1);
% Set up some labels for use later. Note that these are cell array (since they use { })
labels = {'VWMe', 'SMB', 'HML'};
assets = {'Small-Low', 'Small-Medium', 'Small-High', 'Big-Low', 'Big-Medium', 'Big-High'};
% Loop over the assets
for i=1:6
    % Display the asset name. Note the use of the cell array.
    disp(assets{i});
    % Initialize a matrix to hold the rolling beta estimates
    rollingBetas = zeros(T-60+1,4);
    for j=60:T
        % Set up the X and Y data to contain observations j-59 to j
        Y = FFRets(j-59:j,i);
        X = fullX(j-59:j,:);
        rollingBeta=ols(Y,X,1);
        rollingBetas(j-59,:) = rollingBeta;
    end
    % Get the full sample VCV and scale it to compute the standard errors
    stdErrors = sqrt(diag(T/60*VCVWhite{i}));
    % Open a figure for the subplots
    figure(i);
    for j=1:3
        % Loop over the subplots. Note that subplot j will contain B(j+1).
        subplot(3,1,j);
        stdErr = stdErrors(j+1);
        % Set up all x and y data to be plotted. Note that the dates are truncated
        % symmetrically so that the center of the rolling window will be plotted
        % against the beta.
        x1 = FFDates(30:T-30);
        y1 = rollingBetas(:,j+1);
        x2 = x1;
        y2 = beta{i}(j+1) * ones(T-60+1,1);
        x3 = x1;
        y3 = y2 - 1.96 * stdErr;
```

```

x4 = x1;
y4 = y2 + 1.96 * stdErr;
% Plot the data
h=plot(x1,y1,x2,y2,x3,y3,x4,y4);
% Set the axis tight
axis tight
% Set the colors and line styles
set(h(1), 'Color',[0 0 .8], 'LineWidth',2)
set(h(2), 'Color',[.8 0 0], 'LineWidth',2)
set(h(3:4), 'Color',[0 0 0], 'LineWidth',1, 'LineStyle', ':')
% Change the x labels to dates
datetick('x', 'keeplimits');
% Set a title. Note the use of the cell array.
t = title(labels{j});
% These are handle graphics. They are used to make the plot look better
set(t, 'FontName', 'Times New Roman', 'FontSize',14);
% This command does the same but for the tick labels
set(gca, 'FontName', 'Times New Roman', 'FontSize',14);
% Get the figure handle for the current axis
fig = get(gca, 'Parent');
set(fig, 'Position',[100 100 800 600], 'Color',[1 1 1])
set(fig, 'Color',[1 1 1], 'InvertHardcopy', 'off')
end
end

```

Small-Low

Small-Medium

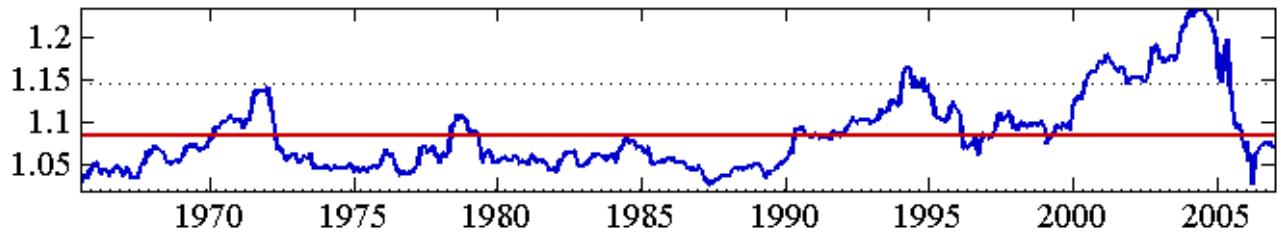
Small-High

Big-Low

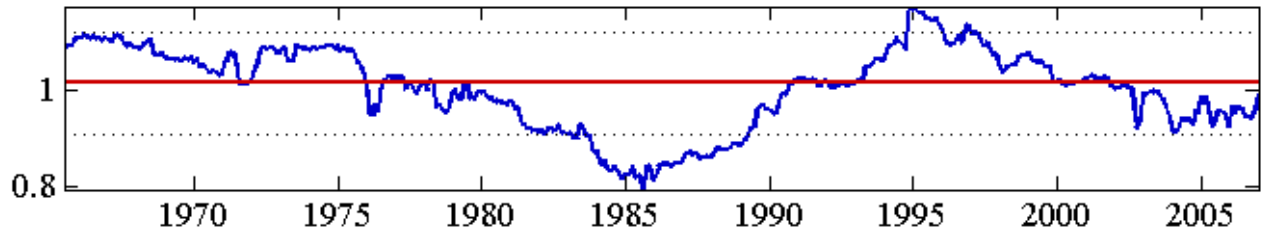
Big-Medium

Big-High

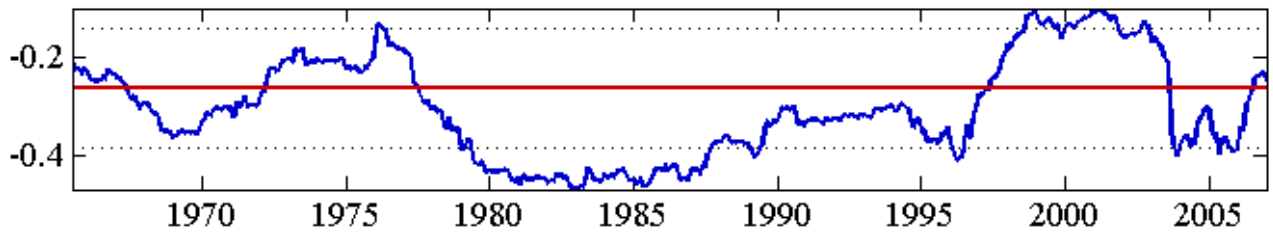
VWMe



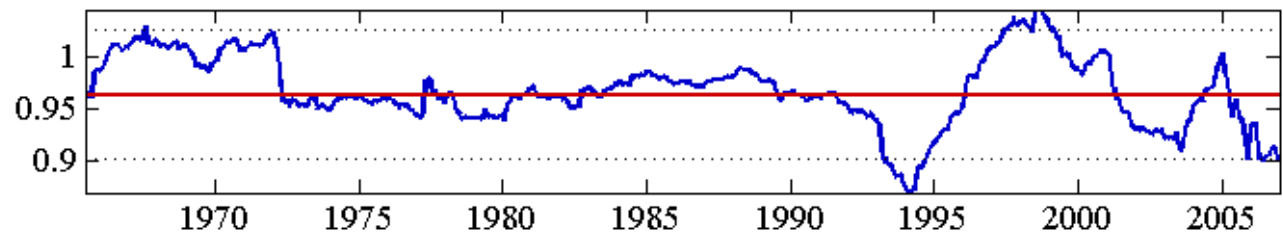
SMB



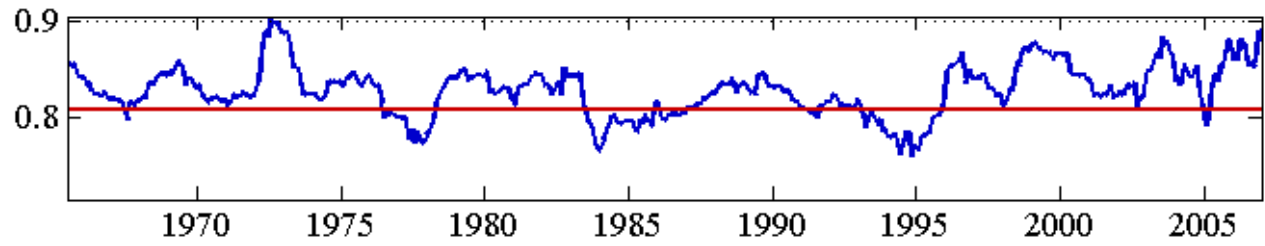
HML



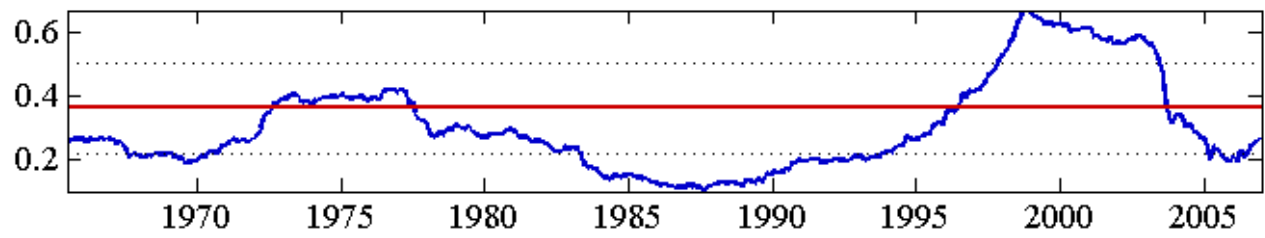
VWMe



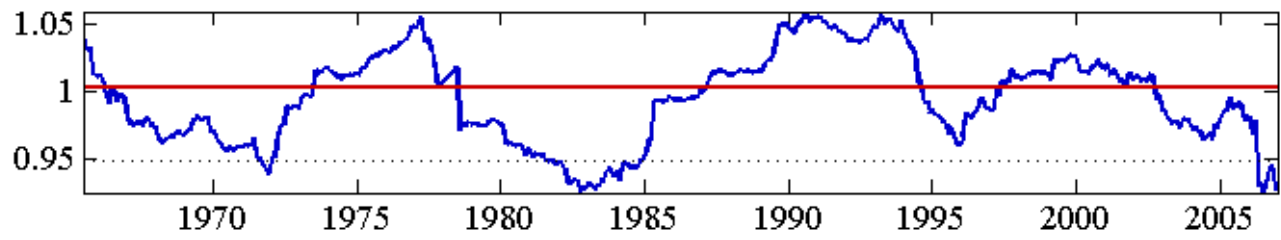
SMB



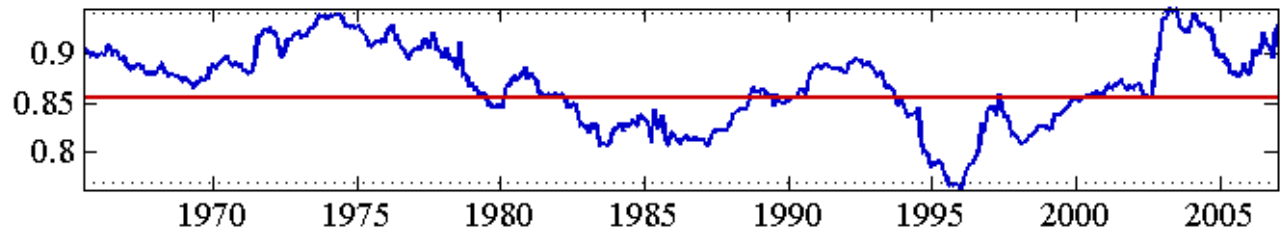
HML



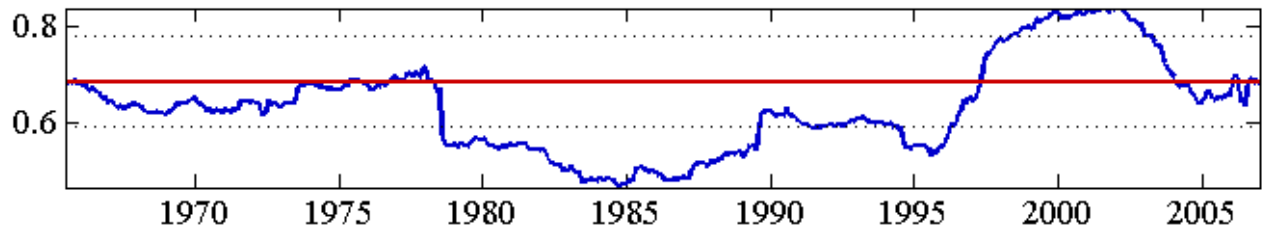
VWMe



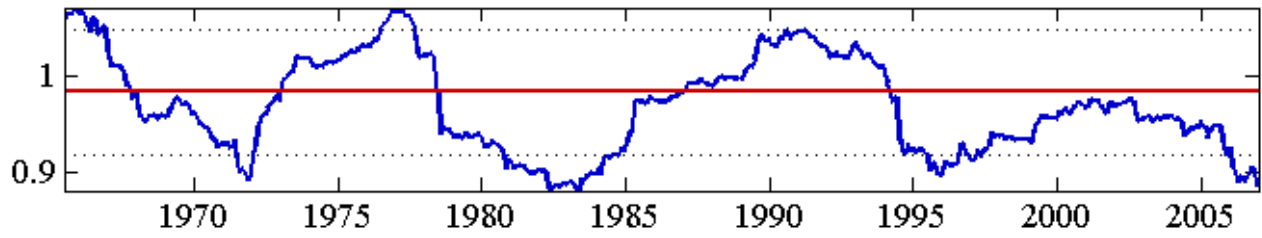
SMB



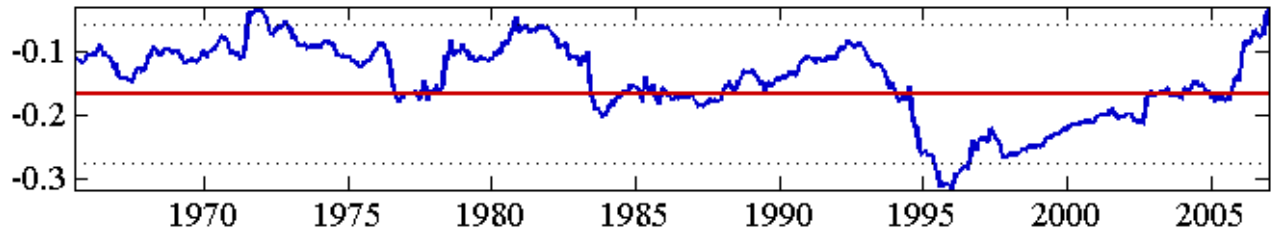
HML



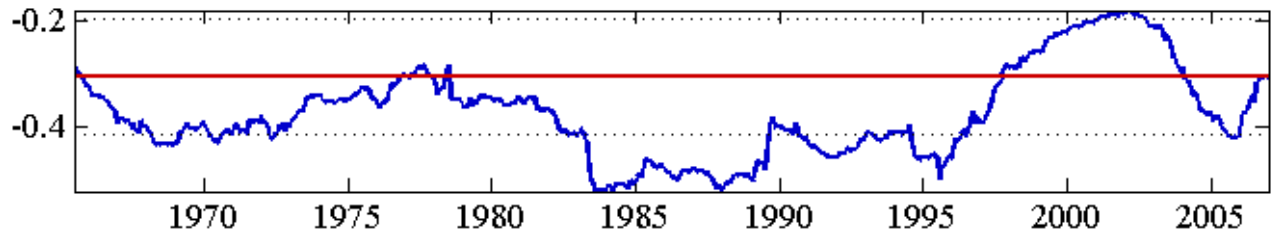
VWMe



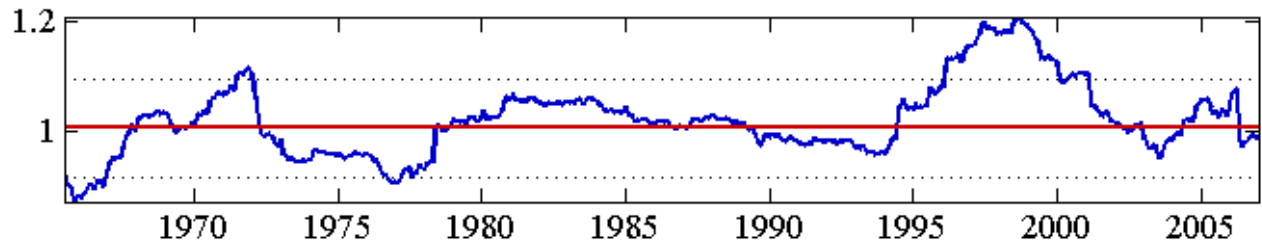
SMB



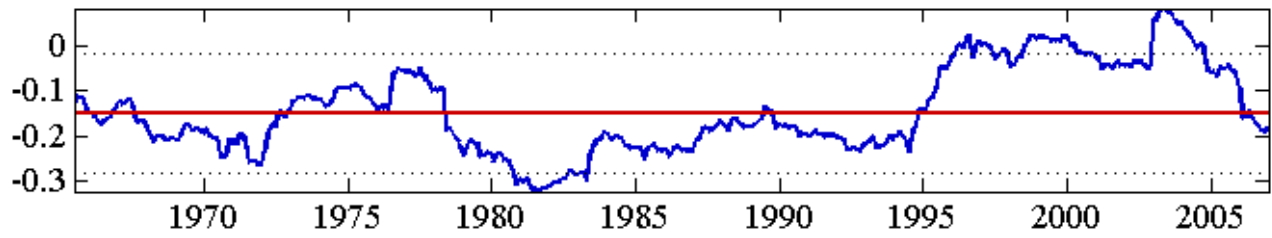
HML



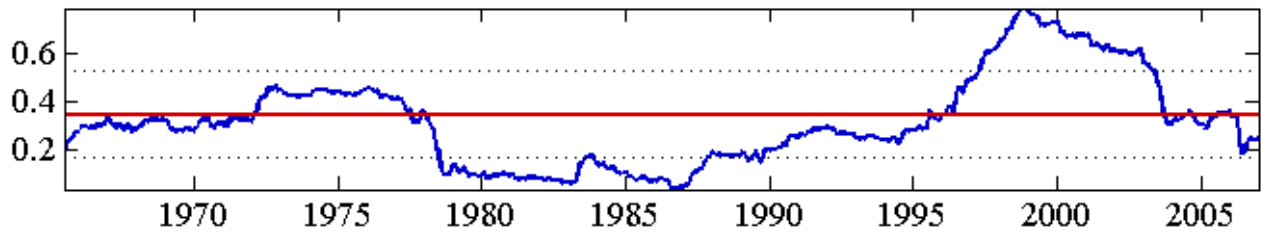
VWMe

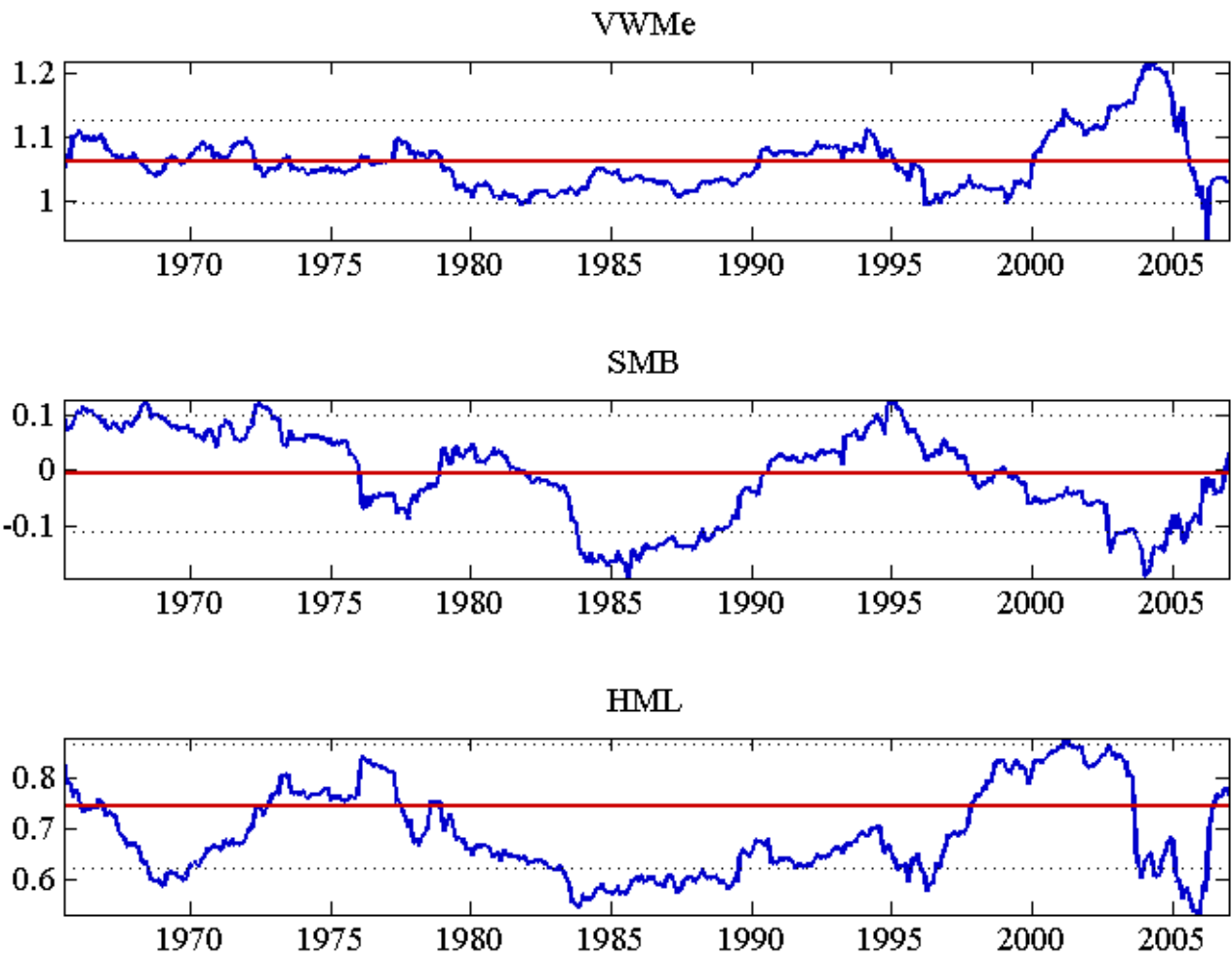


SMB



HML





### Assessing the evidence

There is mixed evidence. The confidence intervals are 95% and so there should be some variation. However, figure 2 (SM) appears to be outside the 95% CI for much of the sample.