Low Frequency Illiquidity Measures

Li Yi[1*]

June 9, 2024

Abstract

I use this article to describe Low frequency illiquidity measures concisely. These measures include: Bid-Ask Spread, Effective Spread, Roll Measure, Amihud Measure, and Pastor and Stambaugh. At the same time, I attach the corresponding MATLAB code with the document.

Measures

Bid-Ask and Effective Spreads

A bid-ask spread measures the difference between an asset's bid price and ask price. It is the most used liquidity measure and normally calculated as a percentage spread. For example ,the bid and ask prices for an targeted asset are 48 and 52. Then the absolute spread[1] is 4 and the percentage spread, aka bid-ask spread, is 4/50=8%.

Here we denote the percentage bid-ask spread as BA. This is to be consistent with the Matlab code.

$$BA_t = \frac{Ask_t - Bid_t}{Mid_t} \tag{1}$$

where $Mid_t = \frac{(Ask_t + Bid_t)}{2}$

The effective spread measures the difference between price and midpoint quotes.

$$ES_t = 2 * |ln(Price_t) - ln(Mid_t)| \tag{2}$$

Matlab

```
BA = (ASK-BID)./((ASK+BID)/2);
ES = 2 * abs(log(abs(PRC))-log((ASK+BID)/2));
```

Roll Measure

Roll (1984) uses autocovariance of price changes (bid-ask bounce). The qwll-known limitation of this measure is that it is only valid for negative autocovariance. Here we provide the final equation. Please check the paper for the intermediate derivation process.

Roll (1984) shows

$$\begin{aligned} Cov(\Delta p_t, \Delta p_{t-1}) &= -\frac{1}{4}s^2 \\ Roll_t &= 2 * \sqrt{Cov(\Delta p_t, \Delta p_{t-1})} \end{aligned} \tag{3}$$

Matlab

```
Roll_lag = Roll_raw(1:end-1, :);
Roll_cur = Roll_raw(2:end, :);
Roll = nancov(Roll_cur,Roll_lag);
```

Amihud

Amihud (2002) proposes an illiquidity measure, which is calculated as the absolute stock return divided by its dollar volume. This measure is normally rescaled by 10^6.

Amihud (1984) is

$$Amihud_t = factor * \frac{|Return_t|}{VOLD_t} \tag{4}$$

Matlab

```
Amihud_{t} = factor * abs(Ret)./(PRC.*VOL);
```

where PRC * VOL is the dollar volume

Pastor and Stambaugh

Pastor and Stambaugh (2003) propose a multifactor model. The individual stock's yearly/monthly/quarterly illiquidity is measured by $\gamma_{i,t}$, which is expected to be negative in sign.
Pastor and Stambaugh (2003) is

$$r^e_{i,t+1} = \theta_{i,t} + \phi_{i,t} * r_{i,t} + \gamma_{i,t} * sign(r^e_{i,t}) * V_{i,t} + \xi_{i,t} \tag{5}$$

where $r^e_{i,t+1}$ is the excess return (in excess of the market return) at time t+1. $V_{i,t}$ is the dollar volume at time t.

Matlab

*Corresponding author: liyifinhub@outlook.com. I thank Dmitry for making his code publicly available.

[1] The absolute bid-ask spread is widely used in CDS market, while the percentage bid-ask spread is commonly used in other financial markets.

```
Dep = Y(2:end, 1);
Ind = X(1:end-1, :);
linear_regression = regstats(Dep, Ind);
Results = linear_regression.beta(3);
```

References

Amihud, Yakov. "Illiquidity and stock returns: cross-section and time-series effects." Journal of Financial Markets 5.1 (2002): 31-56.

Borisenko, Dmitry. "Low Frequency Liquidity Proxies: a Brief Guide to the Simpliest Ones." Working Paper

Fong, Kingsley YL, Craig W. Holden, and Charles A. Trzcinka. "What are the best liquidity proxies for global research?." Review of Finance 21.4 (2017): 1355-1401.

Lubos Pastor and Robert Stambaugh. "Liquidity Risk and Expected Stock Returns" Journal of Political Economy 111.3 (2003): 642-685.

Roll, Richard. "A simple implicit measure of the effective bid-ask spread in an efficient market." The Journal of Finance 39.4 (1984): 1127-1139.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Low Frequency Illiquidity Measures%
% Author: Yi Li                      %
% Date: 09/06/2024                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%
clear all
clc
tic

%% import data from xlsx file
filename = 'D:\...\sample.xlsx';                                    % your won path here
sheetname = 'Sheet1';
range = 'A1:G18115';                                                % adjust this line, to fit ↙
your data
raw_data = readtable(filename, 'Sheet', sheetname, 'Range', range);

% delete data before 2001
raw_data.indicesToDelete = nan(numel(raw_data.DATE), 1);            % create new columne
raw_data.indicesToDelete(year(raw_data.DATE) >= 2001) = 1;          % new columne marks all ↙
data before 2001 as NaN, on and after as 1
raw_data(isnan(raw_data.indicesToDelete), :) = [];                  % delete all data before ↙
2001

% format var for calculation
VOL = raw_data.Volume;
PRC = raw_data.PRICE;
ASK = raw_data.ASK;
BID = raw_data.BID;
STKID = raw_data.TICKER;
MKTRE = raw_data.MKTRE;
date = raw_data.DATE;

% prepare Year, Mont, Quarter ID
Year = year(date);
Month = month(date);
Quarter = 1.*(Month<=3)+2.*(Month>3 & Month<=6)+3.*(Month>6 & Month<=9)+4.*(Month>=10);
[~,~,YearID] = unique(Year);
[~,~,QuarterID] = unique([Year, Quarter], 'rows');
[~,~,MonthID] = unique([Year, Month], 'rows');

% prepare log-return for calculation
Ret = accumarray(STKID, PRC, [], @(x) {diff(log (abs(x)))});
Ret = vertcat(Ret{:});

% delete the first returns for each stock
```

```
PRC = accumarray(STKID, PRC, [], @(x) {x(2:end)});
PRC = vertcat(PRC{:});
VOL = accumarray(STKID, VOL, [], @(x) {x(2:end)});
VOL = vertcat(VOL{:});
YearID = accumarray(STKID, YearID, [], @(x) {x(2:end)});
YearID = vertcat(YearID{:});
QuarterID = accumarray(STKID, QuarterID, [], @(x) {x(2:end)});
QuarterID = vertcat(QuarterID{:});
MKTRE = accumarray(STKID, MKTRE, [], @(x) {x(2:end)});
MKTRE = vertcat(MKTRE{:});
BID = accumarray(STKID, BID, [], @(x) {x(2:end)});
BID = vertcat(BID{:});
ASK = accumarray(STKID, ASK, [], @(x) {x(2:end)});
ASK = vertcat(ASK{:});
TEMP_STKID = STKID;                                          % include a temp variable ↙
to work on STKID
STKID = accumarray(STKID, TEMP_STKID, [], @(x) {x(2:end)});
STKID = vertcat(STKID{:});

% Create tructure array for storing BA, ES, Roll, Amihud, and Pastor and Stambaugh results
N = max(STKID)
Y = max(YearID);
Q = max(QuarterID);

% Year
Y_Results = struct;
Y_Results.Y_BA = NaN(Y,N);
Y_Results.Y_ES = NaN(Y,N);
Y_Results.Y_Roll = NaN(Y,N);
Y_Results.Y_Amihud = NaN(Y,N);
Y_Results.Y_PS = NaN(Y,N);

% Quarter
Q_Results = struct;
Q_Results.Q_BA = NaN(Q,N);
Q_Results.Q_ES = NaN(Q,N);
Q_Results.Q_Roll = NaN(Q,N);
Q_Results.Q_Amihud = NaN(Q,N);
Q_Results.Q_PS = NaN(Q,N);

%% Bid-Ask Spreads

% Calculate Bid-Ask Spreads
BA = (ASK-BID)./((ASK+BID)/2);

% Store Yearly Bid-Ask Results
for i = 1:Y;
    for j = 1:N
```

```
        Y_Results.Y_BA(i,j) = nanmean(BA(YearID==i & STKID==j));
    end
end

% Store Quarterly Bid-Ask Results
for i = 1:Q;
    for j = 1:N
        Q_Results.Q_BA(i,j) = nanmean(BA(QuarterID==i & STKID==j));
    end
end

%% Effective Spreads

% Calculate Effective Spreads
ES = 2*abs(log(PRC)-log((ASK+BID)/2));

% Store Yearly Effective Results
for i = 1:Y;
    for j = 1:N;
        Y_TEMP = ES(YearID==i & STKID==j);
        Y_Results.Y_ES(i,j) = nanmean(Y_TEMP);
    end
end

% Store Quarterly Effective Results
for i = 1:Q;
    for j = 1:N;
        Q_TEMP = ES(QuarterID==i & STKID==j);
        Q_Results.Q_ES(i,j) = nanmean(Q_TEMP);
    end
end

clearvars Y_TEMP Q_TEMP

%% ROLL Measure

% Yearly ROLL Measure
for i = 1:Y;
    for j = 1:N;
        Roll_raw = Ret(YearID==i & STKID==j);
        Roll_lag = Roll_raw(1:end-1, :);
        Roll_cur = Roll_raw(2:end, :);
        Roll_TEMP = nancov(Roll_cur,Roll_lag);

        if Roll_TEMP(1,2)<0
            Y_Results.Y_Roll(i,j) = 2*sqrt(-Roll_TEMP(1,2));
        else
            Y_Results.Y_Roll(i,j) = 0;
```

```
        end

    end
end

clearvars Roll_raw Roll_lag Roll_cur Roll_TEMP

% Quarterly ROLL Measure
for i = 1:Q;
    for j = 1:N;
        Roll_raw = Ret(QuarterID==i & STKID==j);
        Roll_lag = Roll_raw(1:end-1, :);
        Roll_cur = Roll_raw(2:end, :);
        Roll_TEMP = nancov(Roll_cur,Roll_lag);

        if Roll_TEMP(1,2)<0
            Q_Results.Q_Roll(i,j) = 2*sqrt(-Roll_TEMP(1,2));
        else
            Q_Results.Q_Roll(i,j) = 0;
        end

    end
end

clearvars Roll_raw Roll_lag Roll_cur Roll_TEMP

%% AMIHUD Measure

factor = 1e6                                                    % rescale factor
upper_bound = 0.99;                                             %truncat data in order to↙
reduce the power of outliers, Amihud (2002)
lower_bound = 0.01;

% Calculate Amihud Measure
Amihud = factor*abs(Ret)./(PRC.*VOL);                           % PRC * VOL is the dollar↙
volume

% Outliers identified as NaN
for i = 1:N;
    Amihud(STKID==i &(Amihud>=quantile(Amihud, upper_bound) | Amihud<=quantile(Amihud, ↙
lower_bound))) = NaN;
end

% Store Yearly Amihud Measure
for i = 1:Y;
    for j = 1:N
        Y_Results.Y_Amihud(i,j) = nanmean(Amihud(YearID==i & STKID==j));
    end
```

```
end

% Store Quarterly Amihud Measure
for i = 1:Q;
    for j = 1:N
        Q_Results.Q_Amihud(i,j) = nanmean(Amihud(QuarterID==i & STKID==j));
    end
end

%% PASTOR AND STAMBAUGH

% Calculate Excess Return, MKTRE is the S&P500 log-return
Excess_Ret=Ret-MKTRE;

% The regressors matrix
regressors = [Ret, sign(Excess_Ret).*VOL.* PRC];                    % PRC * VOL is the dollar volume

% Yearly PASTOR AND STAMBAUGH Measure
for i = 1:Y;
    for j = 1:N;

        Y_temp = Excess_Ret(YearID==i & STKID==j);
        Dep = Y_temp(2:end, 1);                                     %select 2:end as dependent variable

        X_temp = regressors(YearID==i & STKID==j,:);
        Ind = X_temp(1:end-1, :);                                   %select 1:end-1 as independent variable

        Y_linear_regression = regstats(Dep, Ind, 'linear', 'beta');   %Estimate linear regression (double check results using mdl=fitlm(X,y))
        Y_Results.Y_PS(i,j) = Y_linear_regression.beta(3);          %Store the coefficient

    end
end

clearvars Y_temp Dep X_temp X_temp Dep Ind

% Quarterly PASTOR AND STAMBAUGH Measure
for i = 1:Q;
    for j = 1:N

        Y_temp = Excess_Ret(QuarterID==i & STKID==j);
        Dep = Y_temp(2:end, 1);

        X_temp = regressors(QuarterID==i & STKID==j,:);
        Ind = X_temp(1:end-1, :);
```

```
        Q_linear_regression = regstats(Dep, Ind, 'linear', 'beta');
        Q_Results.Q_PS(i,j) = Q_linear_regression.beta(3);

    end
end

clearvars Y_temp Dep X_temp X_temp Dep Ind

%% end
toc
```