2014 年 1 月

# データ分析の前に

前回の処理内容を振り返っておきます。入手したデータ(samplecsv.csv)に欠損値が存在していたので、その欠損値の 1 期前のデータを代入するという処理を EViews プログラムで行いました。この結果、できあがったデータファイルを data01.wf1 とします。
data01.wf1 を画面に表示し、price2 のオブジェクトを開き、メニューから View/Descriptive Statistics and Tests/ Stats by classification と操作し、Series/Group for classify の項目に code と入力して OK ボタンをクリックします。銘柄ごとの平均、標準偏差を、データ数を一目で把握することができます。

| CODE | Mean | Std. Dev. | Obs. |
|---|---|---|---|
| x100 | 241.3023 | 29.59315 | 1118 |
| x200 | 373.4469 | 29.99888 | 866 |
| x300 | 181.8786 | 29.18443 | 997 |
| x400 | 517.8479 | 42.86735 | 1052 |
| x500 | 285782.9 | 3462.826 | 956 |
| x600 | 351.5190 | 32.03596 | 948 |
| x700 | 217.6525 | 29.06680 | 777 |
| x800 | 403.4112 | 29.80806 | 1138 |
| All | 35085.22 | 93356.57 | 7852 |

今回はこの表をプログラムで作成してみます。プログラミングを始める前に、データの構造について次の事柄を前提とします。

1. 上表から銘柄数は 8 だが、data01 の中に含まれる銘柄数は分からないものとする
2. シリーズ date の最大/最小値を見れば分かるが、データは 2009 年 10 月 1 日から 2009 年 10 月 30 日までのデータである。
3. 各日のデータには必ず、全銘柄(8 銘柄)のデータが入っている。

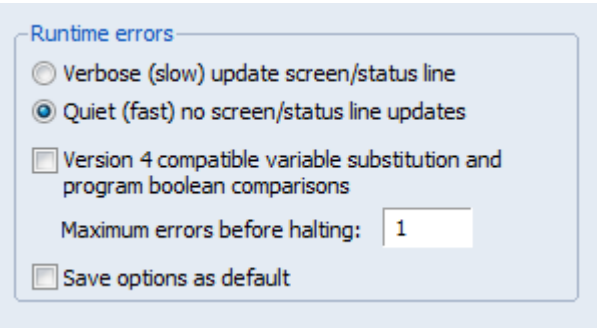
これらを前提として次のようなプログラムを作成します。

```
'data01.wf1を開いておく。
'第一日目(2009年10月1日)のデータに限定して銘柄コードを調べる。

smpl @all if date=20091001

'codeから銘柄コードを拾ってくる方法
'表(tableオブジェクト)の行数はデータを入れることで自由に拡張できるので、
'最初はとりあえず、1行4列の表ptbを作成する。
```

```
table(1,4) ptb

'10月1日のデータに限定し、1行ごとに銘柄を調べていく。
'10日のデータの個数が!obsとなる。
!obs=@obssmpl

'codeの1行目のデータを文字列変数経由でテーブルに代入する。
'もちろん、文字列変数を使わずに、ダイレクトに代入しても良い。
%code=code(1)
ptb(2,1)=%code

'ptbの行数を!jとし、alphaシリーズオブジェクトcodeの行数を!iでコントロールする
'銘柄一覧の作成(表の左側)。!jの1は見出しCODE、2は最初に登場するコード名。

!j=3

'codeの1行目はすでにテーブルに代入しているので、2行目から調べる。
for !i=2 to !obs
    %icode=code(!i)
    if %icode=%code then
        else
        ptb(!j,1)=%icode
        !nos=!j-1
        !j=!j+1
        %code=%icode
    endif
next

'テーブルptbの1行目に列名を入力する
ptb(1,1)="CODE"
ptb(1,2)="Mean"
ptb(1,3)="Std.Dev"
ptb(1,4)="Obs."
ptb(!j,1)="All"

'表ptbの左端のコード情報を利用して平均、標準偏差、データの個数を調べる。
```

```
'前段で銘柄数は!nosに入っている

for !k=2 to !nos+1
    %code=ptb(!k,1)
    smpl @all if code=%code
    ptb(!k,2)=@mean(price2)
    ptb(!k,3)=@stdev(price2)
    ptb(!k,4)=@obssmpl
next

smpl @all
ptb(!k,2)=@mean(price2)
ptb(!k,3)=@stdev(price2)
ptb(!k,4)=@obssmpl

%btm="a"+@str(!k)

'表の一番に下線を引く
ptb.setlines(a1:d1) +b

' 左端の列の右側に縦線を引く
ptb.setlines(a1:{%btm}) +r
show ptb
```

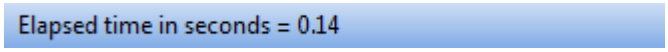
ここまで入力できたら、Run ボタンをクリックしてプログラムを実行します。プログラムを実行したときに、画面上で数値やグラフが何回も再描画される場合があります。そのような場合は右図の Runtime errors の項目で Quiet(fast)を選択します。このオプションを利用することで、処理時間を大幅に節約できます。




また、そもそもプログラムの処理にどのくらいの時間がかかっているのか簡単に調べる方法もあります。次に示すように tic と toc でプログラム全体を囲むようにします。

```
tic
   some commands
toc
```

EViews はプログラムの処理にかかった時間を画面下側のステータスラインに表示します。




## For ループ

for ループはコード名の検索部と、表の中に統計値を入力する 2 カ所で利用しています。for ループの構文を次に示します。

```
   for counter=start to end [step stepsize]
        [commands]
   next
```

counter には数値変数(!が先頭に付く)を利用します。start と end には具体的に数字、または数値変数を利用できます。ステップサイズを省略した場合、start から end まで counter は 1 単位づつ増加します。画面に表示されているワークファイルに対して、具体的には[commands]部分のコマンドを実行します。

for ループと同じように繰り返し処理を行う構文で while ループというものがあります。繰り返し回数が決まっている場合は、for ループを利用します。それに対し、ある条件文が満たされている限り、ループを繰り返すのが while ループです。
EViews のオンラインヘルプにある while ループの例を紹介します。以下は EViews オンラインヘルプにある例題ですので、今回ご紹介してる EViews データファイルで実行することはできません。

```
' 2つの数値変数を設定する
!val = 1
!a = 1

'条件が満たされる限り、wend までの処理を繰り返す。
while !val<10000 and !a<10
smpl 1950q1 1970q1+!a
series inc{!val} = income/!val
!val = !val*10
```

```
!a = !a+1
wend
```

for ループは増分(デフォルトは 1)を自動的に増やしてくれますが、wend は増やしませんので、自分で設定するところにも大きな違いがあります。

## IF 条件文

最初の for ループの中で利用して if 条件文について例題を用いて構文を説明します。

```
if !scale>0 then
   series newage = age/!scale
else
   series newage = age
endif
```

この if 条件文は数値変数!scale がゼロよりも大きな値の場合、シリーズオブジェクト newage=age/!scale とし、0 以下の場合は、series newage = age とするものです。

今回は EViews プログラミングでよく利用する for ループと if 条件文を中心に解説しました。次回は、データの分布についてグラフを用いて調べる方法と基本的な統計量についてみることにしましょう。

■