10 Cookbook: Do-file programming IV

This cookbook chapter presents a number of recipes for Stata do-file programmers using the programming features described in the previous chapter. Each recipe poses a problem and a worked solution. Although you may not encounter this precise problem, you may be able to recognize its similarities to a task that you would like to automate in a do-file.

10.1 Computing firm-level correlations with multiple indices

The problem: a user on Statalist posed a question involving a very sizable dataset of firm-level stock returns and a set of index fund returns. He wanted to calculate, for each firm, the average returns and the set of correlations with the index funds, and determine with which fund they were most highly correlated.

We illustrate this problem with some actual daily stock returns data for 291 firms, 1992–2006, from CRSP (the Center for Research on Securities Prices): 311,737 firm-daily observations in total. We have constructed nine simulated index funds' returns. The hypothetical funds, managed by a group of Greek investment specialists, are labeled the Kappa, Lambda, Nu, Xi, Tau, Upsilon, Phi, Chi and Psi funds. To solve the problem, we define a loop over firms. For each firm of the `nf` firms, we want to calculate the correlations between firm returns and the set of `nind` index returns, and find the maximum value among those correlations. The variable `hiord` takes on values 1–9, while `permno` is an integer code assigned to each firm by CRSP. We set up a Stata matrix `retcorr` to hold the correlations, with `nf` rows and `nind` columns. The number of firms and number of indices are computed by the `word count` extended macro function[1] applied to the local macro produced by `levelsof` ([R] **levelsof**).

```
. qui levelsof hiord, local(indices)
. local nind : word count `indices'
. qui levelsof permno, local(firms)
. local nf : word count `firms'
. matrix retcorr = J(`nf', `nind', .)
```

We calculate the average return for each firm with `summarize, meanonly` ([R] **summarize**). In a loop over firms, we use `correlate` ([R] **correlate**) to compute the correlation matrix of each firm's returns, `ret`, with the set of index returns. For firm n, we move

1. See Section 3.8.

the elements of the last row of the matrix corresponding to the correlations with the index returns into the n^{th} row of the `retcorr` matrix. We also place the mean for the n^{th} firm into that observation of variable `meanret`.

```
. local n 0
. qui gen meanret = .
. qui gen ndays = .
. local row = ‘nind’ + 1
. foreach f of local firms {
  2.         qui correlate index1-index‘nind’ ret if permno == ‘f’
  3.         matrix sigma = r(C)
  4.         local ++n
  5.         forvalues i = 1/‘nind’ {
  6.                 matrix retcorr[‘n’, ‘i’] = sigma[‘row’, ‘i’]
  7.         }
  8.         summarize ret if permno == ‘f’, meanonly
  9.         qui replace meanret = r(mean) in ‘n’
 10.         qui replace ndays = r(N) in ‘n’
 11. }
```

We now may use the `svmat` command ([P] **matrix mkmat**) to convert the `retcorr` matrix into a set of variables, `retcorr1-retcorr9`. The `egen` function `rowmax()` computes the maximum value for each firm. We then must determine which of the nine elements is matched by that maximum value. This number is stored in `highcorr`.

```
. svmat double retcorr
. qui egen double maxretcorr = rowmax(retcorr*)
. qui generate highcorr = .
. forvalues i = 1/‘nind’ {
  2.         qui replace highcorr = ‘i’ if maxretcorr == retcorr‘i’ ///
>         & !missing(maxretcorr)
  3. }
```

We now can sort the firm-level data in descending order of `meanret`, using `gsort` ([D] **gsort**) and list firms and their associated index fund numbers. These values show, for each firm, which index fund their returns most closely resemble. For brevity, we list only the fifty best-performing firms.

```
. gsort -meanret highcorr
. label values highcorr ind
. list permno meanret ndays highcorr in 1/50, noobs sep(0)
```

permno	meanret	ndays	highcorr
24969	.0080105	8	Nu
53575	.0037981	465	Tau
64186	.0033149	459	Upsilon
91804	.0028613	1001	Psi
86324	.0027118	1259	Chi
60090	.0026724	1259	Upsilon
88601	.0025065	1250	Chi

73940	.002376	531	Nu
84788	.0023348	945	Chi
22859	.0023073	1259	Lambda
85753	.0022981	489	Chi
39538	.0021567	1259	Nu
15667	.0019581	1259	Kappa
83674	.0019196	941	Chi
68347	.0019122	85	Kappa
81712	.0018903	1259	Chi
82686	.0017555	987	Chi
23887	.0017191	1259	Lambda
75625	.0017182	1259	Phi
24360	.0016474	1259	Lambda
68340	.0016361	1259	Upsilon
34841	.001558	1259	Nu
81055	.0015497	1259	Lambda
85631	.0015028	1259	Chi
89181	.0015013	1259	Chi
76845	.0014899	1006	Phi
48653	.0014851	1259	Xi
90879	.0014393	1259	Psi
85522	.0014366	454	Chi
80439	.0014339	1186	Chi
85073	.0014084	1259	Phi
86976	.0014042	1259	Chi
51596	.0014028	1259	Tau
77971	.0013873	1259	Xi
25487	.0013792	1259	Chi
14593	.0013747	1072	Kappa
79950	.0013615	1259	Nu
79879	.0013607	127	Phi
12236	.0012653	858	Kappa
77103	.0012513	648	Lambda
81282	.0012314	1259	Chi
75034	.0012159	1259	Phi
46922	.0012045	1259	Xi
82488	.0011911	359	Chi
75912	.0011858	1173	Phi
82307	.0011574	1259	Kappa
83985	.0011543	1259	Kappa
79328	.0011498	1259	Phi
11042	.0011436	1259	Lambda
92284	.0011411	1259	Psi

An alternative approach to the computations, taking advantage of Mata, is presented in Section 14.3.