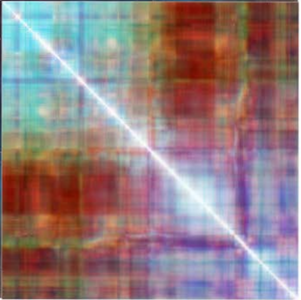
Working with Large Arrays

Data Rave Meetup
March 24, 2015

Working with Large Correlation Matrixes Using J

devonmcc@gmail.com



Questions to answer:

What are covariance and correlation?

Why is correlation useful for work in finance?

What is an array?

Why is array-orientation useful?

What techniques are useful for large arrays?

Questions to answer:

What are covariance and correlation?

Why is correlation useful for work in finance?

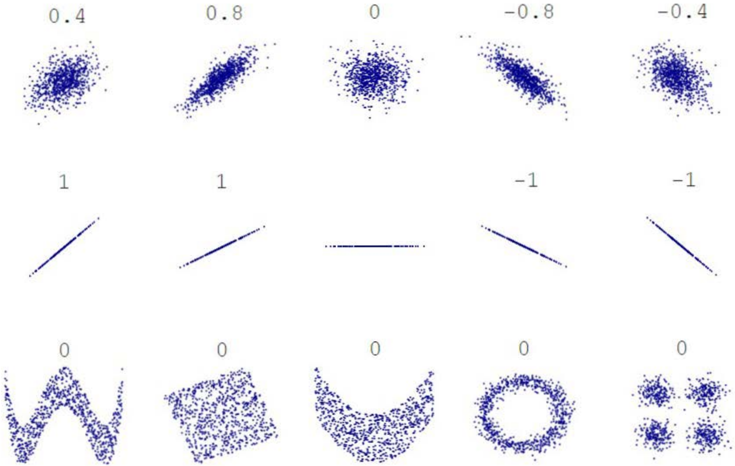
What is an array?

Why is array-orientation useful?

What techniques are useful for large arrays?

Working with Large Arrays

Covariance and correlation describe how two variables relate to each other: whether they are positively or negatively related, or un-related.



Covariance and Correlation

[From https://en.wikipedia.org/wiki/Correlation_and_dependence]

The population correlation coefficient $\rho_{X,Y}$ between two random variables X and Y with expected values μ_X and μ_Y and standard deviations σ_X and σ_Y is defined as:

$$\rho_{X,Y} = \mathrm{corr}(X,Y) = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y},$$

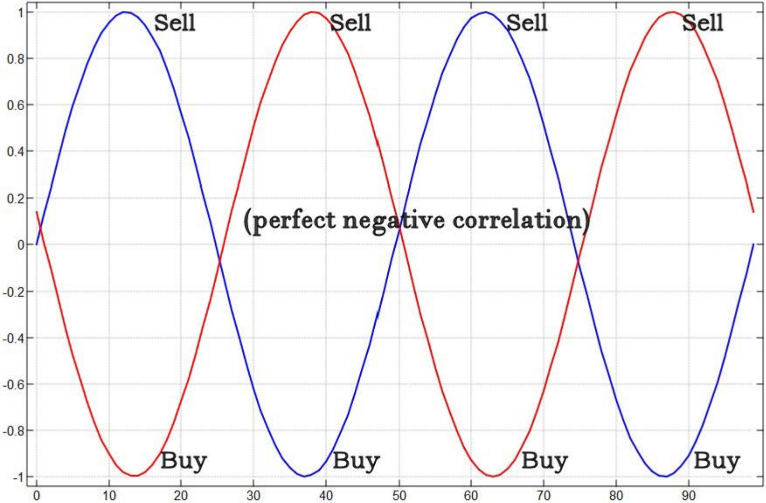
where *E* is the expected value operator, *cov* means covariance, and *corr* is a widely used alternative notation for the correlation coefficient.

If we have a series of n measurements of X and Y written as x_i and y_i where $i = 1, 2, \ldots, n$, then the *sample correlation coefficient* can be used to estimate the population Pearson correlation r between X and Y. The sample correlation coefficient is written

$$r_{xy} = \frac{\sum\limits_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{(n-1)s_x s_y} = \frac{\sum\limits_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum\limits_{i=1}^{n}(x_i-\bar{x})^2 \sum\limits_{i=1}^{n}(y_i-\bar{y})^2}},$$

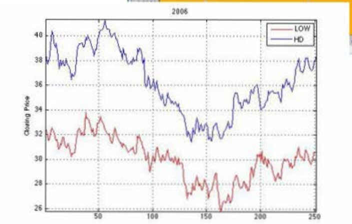
where $\bar{x}$ and $\bar{y}$ are the sample means of X and Y, and s_x and s_y are the sample standard deviations of X and Y.

Ideal Diversification

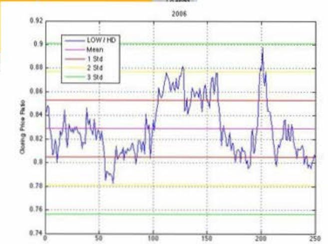


Working with Large Arrays

Pairs Trading: High Correlation



Historical correlations



Some Big-data Features of J

Extended precision numbers

Infinity and negative infinity: _ and __

Hierarchy of nouns (data), verbs (act on nouns), adverbs (combine verbs and nouns -> compound verbs), and conjunctions (verbs combined with verbs).

Simple, generalized array-handling: understanding arrays by their shapes and unexceptional rules for combining them

Open-ended iterator constructs

Memory-mapped files

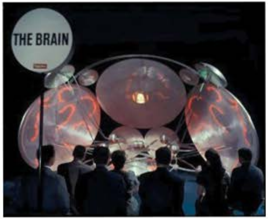
Compact notation clarifies processes and algorithms

```
   2^100   NB. Normal floating-point
1.26765e30
   2x^100  NB. Extended precision
1267650600228229401496703205376
   1r2
1r2
   2*1r2
1
   1r2^100  NB. Rationals
1r1267650600228229401496703205376


   1+2+3+4          NB. Verb "+"
10
   +/1 2 3 4        NB. Adverb "/"
10
   2 +/ \ 1 2 3 4  NB. Adverb "\"
3 5 7
   _2 +/ \ 1 2 3 4
3 7

   i. 2 2
0 1
2 3
    NB. Conjunction
    NB. (matrix multiplication)
   (i. 2 2) +/ . * i. 2 2
2  3
6 11
   -/ . % i. 2 2   NB. Determinant
_2
```

Why Notation?

A Progression...

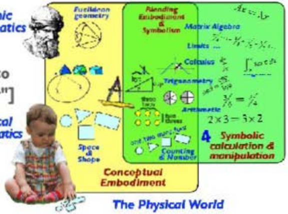
Representational



[From "Visual Design Literacy", http://f10323cdiaz-mihell1.blogspot.com/2010/12/interactions-between-3-levels.html]

Abstract

[From David Tall's "How Humans Learn to Think Mathematically"]



Symbolic

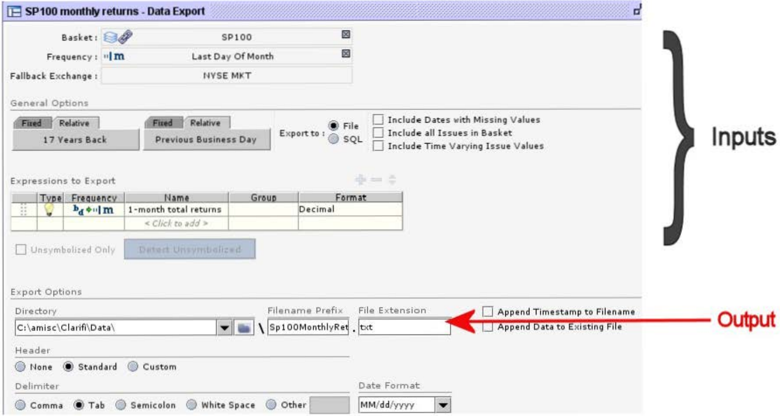
Looking at Some Data

Before we look at the large correlation arrays that reflect more useful, realistic forecasting possibilities, it may be instructive to consider small ones.

First we read a tab-delimited file of monthly returns of the S&P 100:

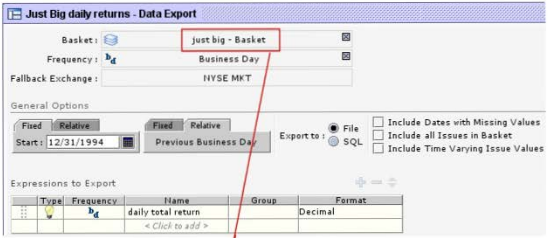
```
   'sptit sp'=. split matFromDelimitedFile 'D:\...\Sp100MonthlyRets.txt'
   sptit
+----+---------+--------------------+
|Date|$issue_id|1-month total returns|
+----+---------+--------------------+
   $sp
20354 3
   3{.sp
+----------+--------+-------------------+
|08/31/2005|00107801|-0.0401956614206721 |
+----------+--------+-------------------+
|09/30/2005|00107801|-0.04996639032041239|
+----------+--------+-------------------+
|10/31/2005|00107801|0.029489319628959176|
+----------+--------+-------------------+
```

Digression on Data Source



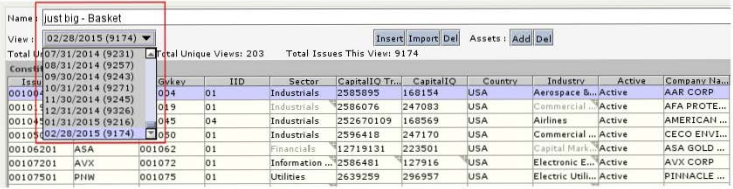
[From ClariFI]

Some Larger Datasets



Here we see an export of daily returns for more than 20 years for a basket called "just big" that has over 9,000 issues in recent time periods, as seen below.

The size of this file is over 140 MB.



Simplifying the Data

Returning to our earlier thread where we've just read in the S&P100 monthly return file as a 3-column matrix, let's look at the data.

```
   #dts=. ~. sp{"1~ sptit i. <'Date'            NB. # unique dates
207
   # ~. allids=. sp{"1~ sptit i. <'$issue_id'   NB. # unique IDs
183
```

File Example

```
Date       $issue_id       1-month total returns
08/31/2005      00107801         -0.0401956614
09/30/2005      00107801         -0.0499663903
10/31/2005      00107801         0.02948931962
11/30/2005      00107801         -0.1178947368
12/31/2005      00107801         0.03328092243
-\---  Sp100MonthlyRets.txt    Top (6,34)
```

However, not all IDs are present for all dates as the composition of the index changes over time.

```
   #/. ~ allids    NB. #s of each ID
115 183 171 195 207 44 11 22 153 95 184 16 84 48 93 28 171 186 70 92 207 207 61 36 111
207 108 207 36 146 106 132 149 115 30 171 113 91 37 207 207 120 197 34 115 36 116 92 207
207 207 112 95 48 207 207 52 28 169 27 36 207 207 207 127 21 58 207 23 186 207 1...
```

We see that some IDs are present for all 207 dates. Let's select only these as this will give us a simple, complete matrix upon which to make our preliminary studies.

```
   #unqids=. (~.allids) #~ 207= #/.~ allids
35
```

So, only 35 IDs are present for the entire time period. We'll start with only these for simplicity, so let's get the returns for only these IDs.

```
   colix=. sptit i. <'1-month total returns' NB. Look up column
   #rets=. (colix{"1 sp)#~allids e. unqids   NB. Select returns
7245
```

Preliminary Data Work

```
   colix=. sptit i. <'1-month total returns' NB. Look up column
   #rets=. (colix{"1 sp)#~allids e. unqids   NB. Select returns
7245

   3{.rets
+-----------------+-------------------+------------------+
|0.13444275892939905|-0.03281970342902496|0.07859880239520955|
+-----------------+-------------------+------------------+

   ;3{.rets                 NB. But these are char representations
0.13444275892939905-0.032819703429024960.07859880239520955

   3{.rets=. n2j&>rets   NB. So make them numeric
0.134443 _0.0328197 0.0785988

   usus rets                NB. Usual stats: min, max, mean, SD
_0.518681 1.16667 0.0102479 0.0891205

   #&>rets;dts;<unqids
7245 207 35
   207*35
7245
```

Here we have three vectors with returns, dates, and unique IDs; the sizes work out as we might expect.

Tabulating Returns

This congruence of sizes illustrates that our returns could be represented as a table with shape # unique IDs by # dates.

```
   #whComplete=. allids e. unqids   NB. Which ID sets are complete?
20641
   +/whComplete
7245
   ixs=. whComplete#(unqids i. allids),&.>dts i. alldts
   $rets=. (rets) ixs } _ $~ (#unqids),#dts
35 207
   _ e. rets
0
```

Spot-check that the table looks correct:

```
   2 5{.rets    NB. Returns for 1st 2 IDs for 1st 5 dates
0.134443   _0.0328197  0.0785988 0.0194308  0.125387
0.0019563   0.151173   0.108924 _0.0214816 _0.0231278
   3{.sp#~whComplete
+----------+--------+-------------------+
|12/31/1997|00144701|0.1344275892939905 |
+----------+--------+-------------------+
|01/31/1998|00144701|-0.0328197034290249|
+----------+--------+-------------------+
|02/28/1998|00144701|0.0785988023952095 |
+----------+--------+-------------------+
```

These first three numbers match these.

Correlation versus Covariance

We will build correlation matrixes from these returns using the "corrMat" verb.
Here's an example of "corrMat" in action:

```
   ]samp=. (i.10),(|.i.10),:10?10
0 1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 3 2 1 0
0 1 8 2 6 7 9 4 5 3

   corrMat |:samp
       1          _1  0.345455
      _1           1 _0.345455
0.345455 _0.345455         1
```

The relation between correlation and covariance:

```
   covar"1/~samp
 8.25 _8.25  2.85
_8.25  8.25 _2.85
 2.85 _2.85  8.25

   8.25%~covar"1/~samp
       1          _1  0.345455
      _1           1 _0.345455
0.345455 _0.345455         1
```

$$\frac{\mathrm{cov}(X,Y)}{\sigma_X\sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X\sigma_Y}$$

Looking at Tables

Even with our small initial sample of security returns, it becomes difficult to read the expansion of data from a correlation matrix, even the 35x35 result of looking at the correlations of the 35 members of the S&P 100 present through our entire time period:

```
   corrMat |:rets
        1   0.18442  0.230039 0.460377  0.25764   0.299866  0.280051   0.533...
  0.18442         1   0.14159 0.307184 0.327299   0.264152  0.157678  0.2374...
 0.230039   0.14159         1 0.239434 0.347943   0.163934  0.144337  0.2591...
 0.460377  0.307184  0.239434        1 0.338124   0.314311  0.191743  0.4712...
...
 0.340417  0.128661   0.35793 0.147008 0.137471   0.111099  0.189696  0.2946...
 0.172584  0.152517  0.207536 0.136011 0.106354 _0.0985161 0.0487254  0.2106...
 0.407972  0.183323  0.169678 0.267821  0.24174   0.143346  0.111134  0.4084...
```

We'll be using a standard J utility called "viewmat" to make it easier to look at large tables. This utility maps the values in a table to integers 0 to 255, corresponding to a simple color palette, as shown here. So, for this matrix,

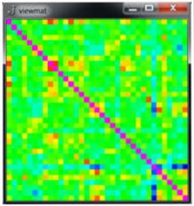
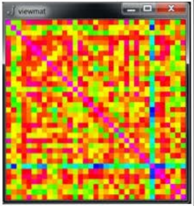
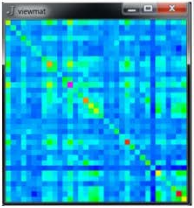
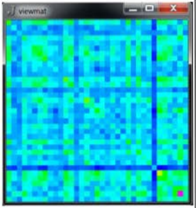
```
   i. 16 16
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
 16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
 32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47
...
224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239
240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255
```

we get this display:

The lowest values are dark blue, the highest are magenta and the "cooler" colors represent lower values whereas the "hotter" ones represent higher ones.

Choosing Correlation over Covariance

For the purpose of this study, we settled on correlation rather than covariance for a couple of reasons. One important one is that correlation values are scaled between one and negative one: this normalizes comparisons between different time periods. To illustrate another reason we might prefer correlation, consider the difference between the two for the entire time series under consideration, then for some subsets of that period.

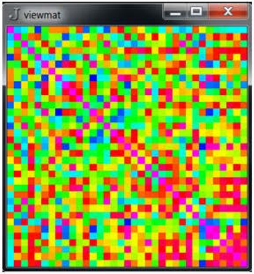
Based on Monthly Returns of 35 S&P 100 Companies from 12/31/1997 Through 2/28/2014			
Correlation		Covariance	
Whole Period	First 12 Months (1998)	Whole Period	First 12 Months (1998)
			
`corrMat |: rets`	`corrMat |:12{."1 rets`	`covar"1/~ rets`	`covar"1/~ 12{."1 rets`

Preliminary Look at the Correlation Data

We see from the observation of the first 12 months – corresponding to 1998 – that the correlations in this period appear to be much higher than across the whole period. Further breaking this down by looking at the first 6 months of 1998 compared to the last 6 months, the difference is striking.

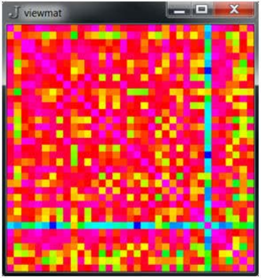
First 6 months 1998

```
corrMat |:(0+i.6){"1 rets
```



Last 6 months 1998

```
corrMat |:(6+i.6){"1 rets
```



The markedly higher correlations in the latter half of that year undoubtedly reflect the Russian debt crisis of that time: it's generally known that correlations heighten during general market downturns. Notice, too, the outlier, evident in nearly all the views we've seen so far, that's about the 7th one from the end. This is evident because of the stability of having the same population throughout the entire period of this small study.

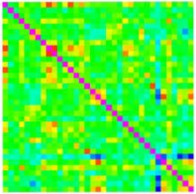
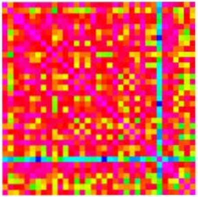
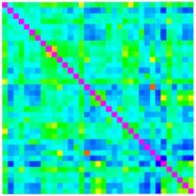
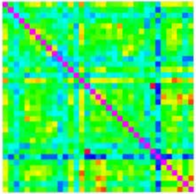
Changing Frequencies

Let's move on to a larger dataset covering the same time period and universe but at a higher frequency: daily returns instead of monthly ones.

Since we'll be re-doing the steps above, changing only the input file, we put those steps into a named function and use it as follows to build the same data structures but for daily data:

```
   'rets dts unqids ndts'=. retFl2Nums 'D:\amisc\Clarifi\Data\Sp100DailyRets.txt'
   $rets
35 4327
   6{.ndts
19971231 19980102 19980105 19980106 19980107 19980108
```

Comparing correlations for daily returns versus monthly returns over the same time period:

<table>
<tr><th colspan="4">Based on Returns of 35 S&P 100 Companies from 12/31/1997 Through 2/28/2014</th></tr>
<tr><th colspan="2">Monthly</th><th colspan="2">Daily</th></tr>
<tr><th>Whole Period</th><th>Latter 6 Months of 1998</th><th>Whole Period</th><th>Latter 6 Months of 1998</th></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td><code>|:rets_mly_</code></td><td><code>6}.12{.|:rets_mly_</code></td><td><code>|:rets_dly_{."1~>:ndts i. 20140228</code></td><td><code>((ndts>19980630) *. ndts<:19981231)#|:rets_dly_</code></td></tr>
</table>

Frequency Differences

Some summary statistics might illuminate these differences:

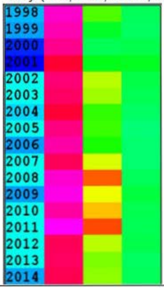
```
   usus ,corrMat |:rets_mly_    NB. Usual stats: min, max, mean, SD
_0.252005 1 0.28451 0.186815
   usus ,corrMat |:rets_dly_{."1~>:ndts_dly_ i. 20140228
0.103347 1 0.349588 0.146282
```

Looking at statistics on the annual correlations for both the monthly and daily series:

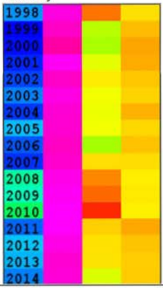
```
   (1998+}:i.+/yrptn_dly_),.usus |:,&>(<-.idy35)*&.>}:corrMat&.>yrptn_dly_<;.1
|:rets_dly_
```

```
1998   _0.077748 0.843596 0.280591 0.134529
1999  _0.0955814 0.820824 0.175058 0.125533
2000   _0.206722 0.789605 0.136329 0.144443
2001   _0.239612 0.709063 0.189883 0.182134
2002           0 0.768096 0.375963 0.146662
2003           0 0.752136 0.353495  0.13818
2004  _0.0482829 0.713288 0.256071 0.124517
2005           0 0.779641 0.263289 0.107588
2006  _0.0982202 0.812075 0.233107 0.113862
2007           0 0.742847 0.404959 0.113386
2008           0 0.847852 0.583068 0.134015
2009   _0.100137 0.870915 0.424115 0.156838
2010           0 0.806984 0.493457 0.141111
2011           0 0.889813 0.599105 0.137402
2012           0 0.743337 0.377228 0.143368
2013           0 0.749535 0.324433 0.126559
2014  _0.0902495 0.735958 0.338822 0.132964
```

Daily (min, max, mean, SD)



Monthly

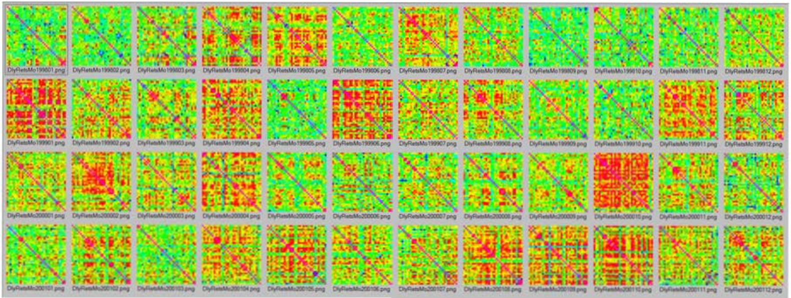


Higher Dimensions

Thinking about partitioning the daily returns into monthly groupings and taking the correlation of each month naturally leads us to working with higher dimensions.

```
   $moptn_dly_=. (1{$rets_dly_){.2~:/\<.100|1e2%~ndts_dly_
4327
   4{.moptn_dly_
1 0 0 0
   +/moptn_dly_
207
   $dlyMo=. corrMat &> moptn_dly_<;.1 |:rets_dly_
207 35 35
```

This gives us 207 35x35 correlation matrixes from which to sample to build a statistical profile of month-by-month correlation, a sample of which looks like this:

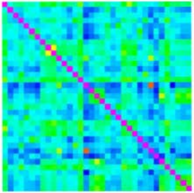


Statistics on Correlations

Even this summarized data tends to be overwhelming. We need to think about how to summarize it.

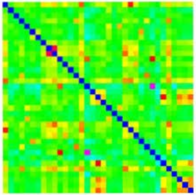
The shape of the 3D array is suggestive – what would it look like if we summed the correlations across time and what might this mean?

```
   $+/dlyMo
35 35
```



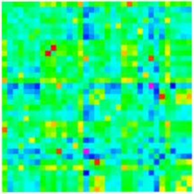
```
viewmat +/dlyMo
```

One problem with this is that the ones on the major diagonal overshadow the other values. We could zero out the major diagonal:



```
viewmat (-. idy35)* +/dlyMo
```

This distorts the non-diagonal values by the other extreme. Let's replace the diagonal by the mean of the non-diagonal values:



```
viewmat (idy35*mean ,(-. idy35)#"1 +/dlyMo)+(-. idy35)* +/dlyMo
```

Moving Up: Bigger Data

How do we handle a larger dataset? We start as before, but get an error:

```
   'rets dts unqids ndts'=. retFl2Nums 'D:\amisc\Clarifi\Data\JustBigDailyRets.txt'
|out of memory: retFl2Nums
|   'sptit sp'=.split     matFromDelimitedFile y
   fsize 'D:\amisc\Clarifi\Data\JustBigDailyRets.txt'
145262256
```

We need an adverb to apply a supplied verb to blocks of the file, aware of the its structure, i.e. tab-delimited fields in LF-delimited rows with the initial row being a header.

```
NB.* doSomething: apply verb to sequential blocks of large file, by whole lines,  Args:
NB. file current location pointer, # bytes in each chunk read, size and name of file,
NB. [any partial chunk from previous call, file header, result of previous call to be
NB. passed on to next one].
doSomething=: 1 : 0
   'curptr chsz max flnm leftover hdr passedOn'=. 7{.y
   if. curptr>:max do. ch=. curptr;chsz;max;flnm
   else. if. 0=curptr do. ch=. readChunk curptr;chsz;max;flnm
           chunk=. leftover,CR-.~>_1{ch                 NB. Last complete line.
           'chunk leftover'=. (>:chunk i: LF) split chunk   NB. LF-delimited lines
           'hdr body'=. (>:chunk i. LF) split chunk     NB. Assume 1st line is header.
           hdr=. }:hdr                                  NB. Remaining part as "leftover".
       else. chunk=. leftover,CR-.~>_1{ch=. readChunk curptr;chsz;max;flnm
           'body leftover'=. (>:chunk i: LF) split chunk
       end.
       passedOn=. u body;hdr;<passedOn              NB. Pass on u's work to next invocation
   end.
   (4{.ch),leftover;hdr;<passedOn
NB.EG ((10{a.)&(4 : '(>_1{y) + x +/ . = >0{y')) doSomething ^:_ ] 0x;1e6;(fsize
'bigFile.txt');'bigFile.txt';'';'';0  NB. Count LFs in file.
)
```

How the Adverb "doSomething" Works

The last line, starting "`NB.EG`" gives an example of how to use the adverb. We'll detail the elements of this line in the following table.

`(10{a.)&`		10th character of atomic vector – LF – bound as left argument (`&`)
`(4 : '(>_1{y) + x +/ . = >0{y')`		Anonymous dyadic function (*x f y*) to count occurrences of *x* in *y*.
`doSomething`		Name of adverb.
`^:_ ]`		Power iterator (`^:`) applied "infinitely" many times (_) to right arg (`]`).
The following right-hand arguments to "doSomething" are assigned local names on the first line – shown here in the middle column. These values are separated by a semi-colon which "boxes" each argument into its own cell.		
`0x`	currptr	Location in file at which to start reading; "x" gives extended integer.
`1e6`	chsz	"Chunk" size – maximum number of bytes to read each time.
`(fsize 'bigFile.txt')`	max	Maximum number of bytes to read; here, end of file.
`'bigFile.txt'`	flnm	Name of file.
`''`	leftover	[Any partial (incomplete) line left over from previous iteration.]
`''`	hdr	[A boxed vector of column entries from first row of file.]
`0`	passedon	[Value passed on from previous invocation of verb.]

These last three values are initialized to be empty vectors or zero, depending on the type of value to be appended or assigned for each iteration within the adverb. The power iterator (^:) repeatedly calls the adverb with its verb left argument, supplying the result of the previous call as the right argument each time.

An Example of Using the "doSomething" Adverb

In this case, we need to process a file in pieces because it's too large to process all at once. We want to pull the same values out and process them as we did for our smaller file. So, we first apply a verb to extract the "key" columns – the date and issue ID - to a file (in case the result is also too large to hold in memory).

```
accumGblKeysVar2Fl=: 3 : 0
   (unmatify 2{."1 usuMatify >0{y) fappend >_1{y
   >_1{y              NB. Pass on file name.
)

   fsize flnm=. '\amisc\Clarifi\Data\JustBigDailyRets.txt'
145262256
   initArgs=. 0x;1e6;(fsize flnm); flnm; ''; ''; 'KeyFl.txt'
   6!:2 'rr=. (accumGblKeysVar2Fl) doSomething ^:_ ] initArgs'
15.7446
```

We see that this takes about 16 seconds. This "KeyFl.txt" is small enough to read in and parse all at once:

```
   fsize 'KeyFl.txt'
69586060
   $keys=. matFromDelimitedFile 'KeyFl.txt'
3479303 2
```

File Example

```
01/05/1995          00101301
01/06/1995          00101301
01/09/1995          00101301
01/10/1995          00101301
01/11/1995          00101301
01/12/1995          00101301
-(Unix)---    KeyFl.txt
```

We get the returns by applying another verb to the file:

```
   accumRets=: 3 : '(>_1{y),n2j&>_1{"1 usuMatify >0{y'
   6!:2 'rets=. (accumRets) doSomething ^:_ ] 0x;1e6;(fsize flnm);flnm;'''';'''';'''''
17.9782
   $rets=. >_1{rets
3479303
```

This takes about 18 seconds and we now have all the data we need from the file.

Making and Using a Larger Table

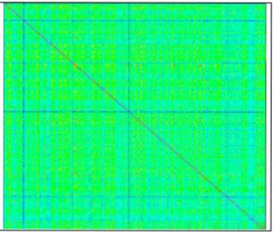
```
   'alldts allids'=. <"1 |:keys
   'dts unqids'=. ~.&.>alldts;<allids
   dts=. dts/:'/' dt2num&>dts
   $ndts=. '/' dt2num&>dts                     NB. Numeric, orderable version of dates.
5080
   $unqids=. (~.allids) #~ (#dts)= #/.~ allids   NB. Unique IDs for entire time period
400
   whComplete=. allids e. unqids
   ixs=. whComplete#(unqids i. allids),&.>dts i. alldts
   $rets=. (whComplete#rets) ixs}_$~(#unqids),#dts
400 5080
   _ e. rets
0
```

Since we have multiple versions of the same data, keep the sets straight by assigning each to its own namespace:

```
   vnms=. 'rets';'ndts';'dts';'unqids'
   #&>".&.>vnms,&.>(<'_ldly_'),&.>(<'=: '),&.>vnms
400 5080 5080 400
```

We can look at the correlations for the comparable period:

```
   selDts=. (ndts>:19971131)*.ndts<:20150228
   viewmat corrMat selDts#|:rets
```



This shows us the limitations of viewing the data once it gets large enough. In fact, we can generate a matrix far too large to be meaningfully viewable:

```
   $corrMat rets
5080 5080
```

Memory-mapped Files

Another method for working with large datasets is to use memory-mapped files. This perhaps does not scale as well as the adverb method but works well for somewhat large files:

```
   require 'jmf'
   fsize flnm=. '\amisc\Clarifi\Data\FairlyBigDailyRets.txt'
121983235
   JCHAR map_jmf_ 'bigfl';flnm
   $bigfl
121983235
   75{.bigfl
Date  $issue_id   daily total return
04/30/1998  00101301    0.006302521008403339
```

Memory-mapped files take advantage of the operating system's paging facility. With a larger address space, say with 64-bit J, larger files can also be accessed this way.

```
   JVERSION
Engine: j803/2014-10-19-11:11:11
Library: 8.03.12
Platform: Win 64
…
   JCHAR map_jmf_ 'bigfl';'\amisc\Clarifi\Data\JustBigDailyRets.txt'
   $bigfl
145262256
   6!:2 'whlf=. I. LF=bigfl'    NB. Where are line-feeds?
0.401335
   _5{.whlf
145262102 145262144 145262187 145262230 145262255

   bigfl{~145262145+ i. (#bigfl)-145262145
03/04/2015  31605601    -8.547008547008517E-4
03/05/2015  31605601    -0.005816937553464574
03/06/2015  31605601    0.0
```

Large Data in Pieces

Another method for dealing with a large dataset is to put it in pieces to files saved in the format of J variables. I used this technique for working with the Netflix Challenge dataset. Using this technique, I developed adverbs, like the following, for working with these collections.

```
NB.* getVarInfo: apply arbitrary function to each (filed) var named.
getVarInfo=: 1 : 0
   'dd varnm'=. y                    NB. Vars dir, var names.
   rc=. dd unfileVar_WS_ varnm
   if. >{.rc do. rc=. 1;u ".varnm
       [4!:55 <varnm
   end.
   rc
NB.EG ({."1,.{:"1) getVarInfo &.>(<'C:\data\');&.>'var1';'var2';'var3'
NB.EG dts=: (3&{) getVarInfo&.>(<VDIR);&.>MVN
)
```

This adverb could be applied with an arbitrary verb in the following manner:

```
meanMovieRatings getVarInfo&>(<VDIR);&.>MVN[4!:55 <'MMR'
```

where “VDIR” is a directory of variables on file and “MVN” is a list of relevant file names.

Arrays: Breaking Up is Easy To Do

Often we'll encounter a hiccup in an attempt to do something all at once as is natural with array processing.

```
   +/moptn=. (1) 0 } (1{$rets){.2~:/\<.1e2%~ndts
243
   $moptn
5080
   moptn i: 1   NB. Location of last one in partition vector.
5074
   _6{.ndts
20150227 20150302 20150303 20150304 20150305 20150306
```

But, when we try to partition our daily returns into months and make a correlation matrix for each month:

```
   $corrMat &> moptn <;.1 |:rets
|out of memory
|   $     corrMat&>moptn<;.1|:rets
   $moptn <;.1 |:rets
243
   $corrMat &> 122{.moptn <;.1 |:rets
|out of memory
|   $     corrMat&>122{.moptn<;.1|:rets
   $corrMat &> 61{.moptn <;.1 |:rets
61 400 400
```

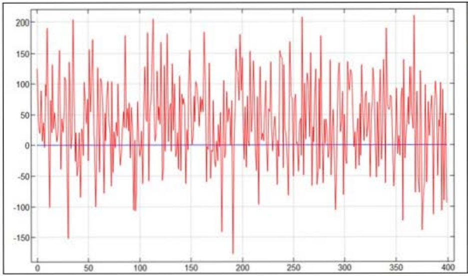
The three most common programming problems:
1) Naming things, and
2) Off-by-one errors.

Here, we ran out of memory trying to do everything at once, so we scaled back to half, then a quarter of until we are able to get a partial answer.

The Sky's the Limit

Here, I'm exploring the idea that a month's correlation matrix can be used as a system of linear equations to solve for future returns.

```
   $somRets=. 61{. (_1 |. moptn) # |:rets  NB. Start-of-(next)Month returns
61 400
   coeffs=. (0{corrMat &> 61{.moptn <;.1 |:rets) %. 0{somRets  NB. Linear eq'n coeffs
   ret2d=. 0 { (_2 |. moptn) # |:rets  NB. 2nd day of month returns
   estpx=. coeffs +/ . * corrMat &> 0{(_1 |. moptn) <;.1 |:rets  NB. Apply coefficients
   load 'plot'
   plot ret2d,:estpx
```



The plot reveals a scaling problem so we'll adjust our estimate to have the same mean and standard deviation as the actual return series.

```
   usus estpx
_177.593 209.835 37.6117 70.0167
```

```
   estret=. (mean ret2d)+(stddev ret2d)*(stddev estpx)%~estpx-mean estpx
   usus estret
_0.0543829 0.0425288 _0.000551367 0.017514
```

Trying Things on the Fly: Failing Quickly

After this adjustment, the means and standard deviations of the two series are the same.

```
   estret=. (mean ret2d)+(stddev ret2d)*(stddev estpx)%~estpx-mean estpx
   usus estret
_0.0543829 0.0425288 _0.000551367 0.017514
   usus ret2d
_0.146875 0.0588235 _0.000551367 0.017514
'title Return vs. Estimate;key Return Estimate' plot ret2d,:estret
```

It's hard to tell from this plot how much the two differ, so add in an error line:

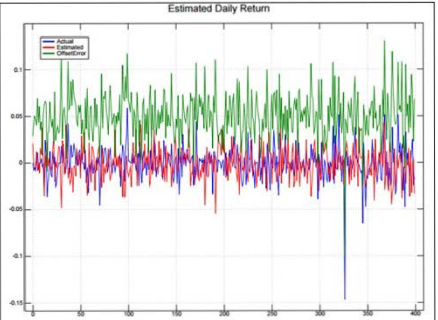
```
   plot (],-/) nn=. ret2d,:estret
```

It's hard to distinguish the error line, so offset it an arbitrary amount so it's easier to see.

```
   dat=. 0 0 0.05+"(0 1) (],-/) nn
   plarg=. 'title Estimated Daily Return;'
   plarg=. plarg,'key Actual Estimated'
   plarg=. plarg, ' OffsetError'
   plarg plot dat
```

Let's also get a numerical measure of how well our estimate fits:

```
   corrMat ret2d,.estret
         1 _0.0343239
_0.0343239          1
```



J Resources

Join the J Forums (mailing lists) –

- programming - the main forum, covering J programming from beginner to expert, and announcements
- chat - all other discussions on computer languages and J - messages welcomed from both J and non-J programmers
- general - installation, support, website and other infrastructure topics

Vocabulary - http://www.jsoftware.com/jwiki/NuVoc
A Brief J Reference - http://www.jsoftware.com/books/pdf/brief
Minimal J - http://www.jsoftware.com/jwiki/DevonMcCormick/MinimalBeginningJ
Learning J - http://www.jsoftware.com/learning/contents.htm
The J Meetup - http://www.meetup.com/J-Dynamic-Functional-Programming, also http://www.jsoftware.com/jwiki/NYCJUG
J Software - http://www.jsoftware.com/
J Wiki - http://www.jsoftware.com/jwiki/FrontPage
J in a Day - http://www.jsoftware.com/jwiki/IanClark/JinaDay
Oleg's J page - http://olegykj.sourceforge.net/
Books on J - http://www.jsoftware.com/jwiki/Books

What's this?

```
   2$<'hip'
+---+---+
|hip|hip|
+---+---+
```

?