Chapter 3
Managing and Analyzing Data

The previous chapter focused on data display, with very little or no attention paid to DataView's analysis capabilities. However, as noted before, in practice there is not a hard-and-fast separation between display and analysis. DataView lets you combine and transform variables just by typing the formula, even in the context of what is ostensibly a display command; for example the expressions:

```
100*(CONS+GOVP)/GNP
exp(0.2*ln(WPAGR)+0.2*ln(WPEN))
```

can be displayed as plots or a tabular transformation display as easily as can a single series. Moreover, the commands can be captured from one context and pasted into another in less time than it has taken for you to read this.

DataView has been specifically designed to foster this type of coordinated processing whenever possible, with the aim of permitting you to do your work without having to worry about the details. Nevertheless, there are times when it is helpful to know something about the way in which the program does its processing. In particular, the provision DataView makes for automatic data storage in a Memory File is an important issue and the subject makes a good background introduction to data analysis. In a nutshell, the issue in the abstract is that when transformations of existing data series are made and stored, they need to be put somewhere.

The Memory File

Whenever data series are created explicitly, or as a by-product of some operation, if they need to be stored longer than a few seconds DataView will usually put them into a temporary work file known as a "Memory File." If this file already contains other data series, the new series are simply added to the file. However, if the file does not exist, then it will be created automatically as needed.

Consider a simple example: In order to perform a calculation, DataView usually needs to know something about the data to be used, generally the observation frequency and the date range. Thus it is good practice to begin by setting the frequency and dates, either by clicking on the status bar or using the **Settings** menu. Alternatively, from the Central Control screen you can issue the commands:

```
SET FREQ=QUARTERLY
SET DATES=198001-199204
```

Next, to cause the creation of a Memory File, from the Central Control screen try typing the simple equality:

```
A=1
```

In response to this command, DataView will create a Memory File with suitably wide date limits, its choice of these depending on either the dates you have set or the default date limits, whichever is greater. This Memory File will contain a single series, named "A," with all its observation values equal to 1 for the date range you set. The status bar will now display "Memory File Open" to indicate that the Memory File has been created. Click on it if you wish to view the Memory File index. Or you can display the series itself by typing:

View A

But what if you would like to make changes? Once a Memory File has been created, if for instance you do not like the date limits chosen by DataView, or if you want to change the data frequency, you can type:

CLEAR MEMORY FILE

and start again by clicking on **New** from the **File** menu category, which will display the form shown in Figure 3-1. In this case, you can make sure that the Memory File is opened with the particular base frequency and date limits you want, since DataView permits you to supply directly a description and date limits; the latter do *not* have to conform with the date range set earlier for transformations. Note that you will, however, need to click the radio button labelled Memory File, in order to specify that it is a Memory File that you wish to initialize.

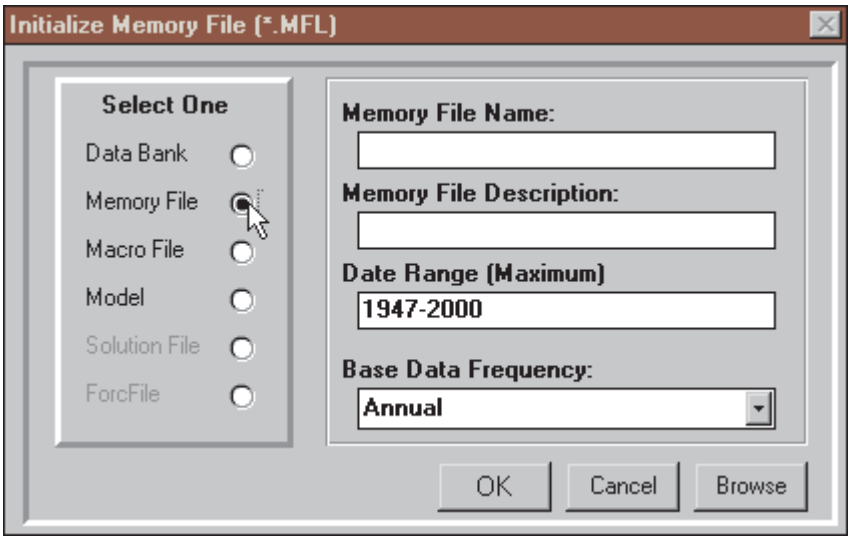

Figure 3-1. Memory File Initialization

The date limits and frequency you select will jointly determine the maximum number of observations per series for this Memory File. The choice of frequency does *not* limit you only to data of that frequency. Provided they have different names, series of different frequencies can co-reside in the same Memory File. But in order to reserve an appropriate amount of space, you should

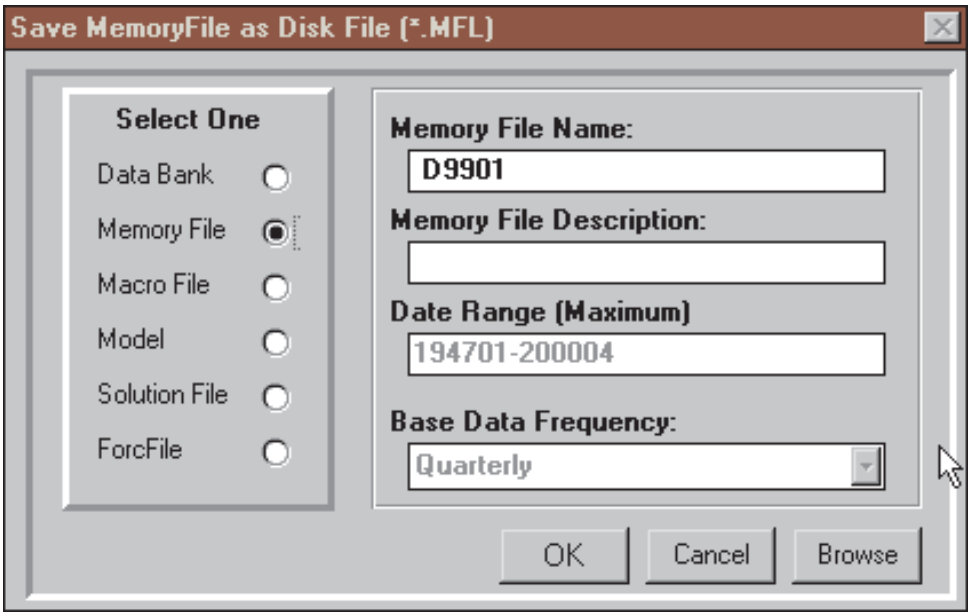

Figure 3-2. Saving a Memory File

initialize a Memory File with the frequency set to the **highest** frequency of any series you intend to store in that file.

At this point, it may be useful to pause for a moment's reflection. We have just considered two ways of creating a Memory File, the first as a by-product of creating observations on a series and the second as a conscious action. The fundamental difference between the methods is that in the first case, the date limits are set by the operation of creating a series whereas in the second we set the date limits independently of any particular transformation or other data generation operation.

In the same sense that you may want to control the file creation process, you may also want to save the Memory File from session to session. This file may be created initially in order to provide for temporary data storage. However, if you create a Memory File containing data that you want to keep, you can tell DataView to save a copy by issuing the SAVEMF (SAVE MemoryFile) command:

SAVEMF D9901

Or you can click on **File | Save Memory File As**. In either case, DataView will respond by displaying the form shown in Figure 3-2:

If saved this file will appear in your Home directory as D9901.MFL.

To reload the data later, in the same or a subsequent DataView session, the command is:

LOADMF D9301

The alternative to using these commands is to click on the relevant, rather obvious items under **File**, as noted. However, to use the menu options here is rather more long-winded than issuing the commands.

Combining Data in the Memory File

Once it exists, you can put data into the Memory File in several different ways. To copy a series from an open Data Bank, use:

FETCH GNP

or

FETCH GNP=GNP

or simply:

GNP=GNP

where in the second and third cases the name to the left of the equals sign can be different from that of the series in the Data Bank.

What is the difference between these commands? The answer is: if, previously, the Memory File series did not exist, they are equivalent. However, if it exists, then the FETCH command must be used whenever the names are the same in both data bank and Memory File; otherwise, the series will instead be retrieved from the Memory File, resulting in a do-nothing operation. The reason is that, except when the FETCH command is used, the Memory File is *always searched first, before any bank*, whenever data series are to be retrieved.

You can also copy a group of series. For instance, the command:

COPY SERIES=CON* FROM DEMOBANK TO MEMFILE

will fetch all series from DEMOBANK the names of which begin with the letters "CON". The DataView syntax obeys the standard wildcard conventions, with ? serving as a placeholder and * indicating that the remaining characters (including blanks) are to be accepted whatever they are. Or you can click on **Data** | **Copy Series** to perform the same action.

In addition, you can create a new series in the Memory File (or overwrite an existing one) using a formula or function such as:

SAVRAT=100*(1-CONS$/YPERS$)
COSTINDEX=exp(0.2*ln(WPAGR)+0.2*ln(WPEN))

The consequence of the execution of each of these commands is the retrieval of the right-hand-side series from the Memory File or a data bank, the calculation of the stated transformation and the storage of the results, under the name given on the left-hand side in the Memory File. If no series previous exists with that name, it will be created. If a series exists, its observations will be over-written for the date range currently in force, *provided that the frequency of the existing series is the same*; if it is not, no change will occur.

Managing The Memory File

The Memory File has an index very much like that of a Data Bank. You can search this index using the INDEX command; for example:

INDEX
INDEX (CON-CP)

Note that the word INDEX is used on its own, rather than in combination with any sort of file name. Both these commands produce brief indexes, but the second displays only series the names of which fall into the name range from CON* to CP*. Alternatively, if you wish to find the names of series that have particular characteristics, and display a standard index, you can use the expanded form of this command:

INDEX (CON-CP, STD)

The same commands that are used to build and update series documentation in a Data Bank can also be used for Memory File series. However, the Data Bank (rather than the Memory File) is intended to be used for archival storage; if you go to the trouble

Figure 3.3

DataView Functions

LN(expr)	natural logarithm
EXP(expr)	exponential
LOG10(expr)	logarithm to base 10
SIN/COS/TAN(expr)	trigonometric functions
ABS(expr)	absolute value
ROUND(expr)	round to whole number
TRUNC(expr)	truncate to whole number
MOD(expr1,expr2)	remainder of expr1 on division by expr2
PDIFF(expr1,expr2)	absolute difference expr1 minus expr2
SIGN(expr1,expr2)	absolute value of expr1 with sign of expr2
APCT(expr)	annualized one period percentage change
APCT(j,expr)	annualized percent change in series compared with n periods before (j=1,2,4,12 or 52)
CUMS(C,expr)	Cumulative Sum: x(t=x(t-1)+expr(t); x(0)=C
DIF(expr)	first difference: expr(t) - expr(t-1)
DIF(n,expr)	expr(t) - expr(t-n)
GROW(C,expr)	Grow Function: x(t)=x(t-1){1+expr(t)*0.01};x(0)=C
LAG(n,expr)	expr(t-n)
LEAD(n,expr)	expr(t+n)
LGT(expr)	logit: ln[expr/(1-expr)]
LN(n,expr)	ln(expr(t-n))
MAVG(n,expr)	Sum[expr(t),...,expr(t-n+1)]/n
TRND(expr)	Time trend regression values of series
MAX(expr)	maximum value in current date range
MIN(expr)	minimum value in current date range
MEAN(expr)	mean value over current date range
SDEV(expr)	standard deviation over current date range
VAR(expr)	variance of current date range
SELECT(i/j,series)	ith observation out of every j
FTRAN(CODE,series)	higher to lower frequency conversion
DIS(CODE,series)	lower to higher frequency conversion
TIME(value,date)	time variable taking set value at given date
DUM(date=value)	all observations zero except at given date
DUM(Fk=value)	seasonal dummy; frequency F=S,Q,M, or W

of documenting your series it is usually best to set up your own Data Bank, as described in the next chapter of this User Guide.

A useful Memory File facility is the RENAME command:

RENAME CPI CPIX

Here it is used to rename CPI to CPIX. You may want to rename a series before displaying it in a hard-copy plot, in order to group series alphabetically, or to distinguish variants from different sources (as, for example, in the case of results from different simulations of a model).

Sometimes you may also wish to delete one or more Memory File series. The command is:

DELETE seriesname

Similarly to the case when you delete a DOS or Windows file, the delete command does not actually remove the series from the Memory File; instead, the series is "turned off." Consequently a series can be restored as well:

RESTORE seriesname (k)

where k is the location number ("series number") provided when a series is deleted.

The menu analogues to these commands can be found under the **Data** category of the Central Control screen menu or, in the case of the index command, by clicking on the status bar item "Memory File Open."

Operators and Functions

In general, expressions are written using the normal mathematical conventions: In particular, lagged observations may be indicated by putting (-k) after the series name, where k is some integer number: for example

DGNP1=GNP-GNP(-1)
DGNP2=GNP-0.5*GNP(-1)-0.2*GNP(-3)-0.2*GNP(-4)

Most of the functions available are listed in Figure 3.3 below. They include the TIME function and dummies, for example.

TIME(1,198001)
DUM(Q3=1)
DUM(197401=1.0)

The arithmetic operators recognized by DataView are:

****	Power
*	Multiplication
/	Division
+	Summation
-	Subtraction

These operators are given in hierarchical order. Power first, Multiplication and Division (co-equal) second, and Summation and Subtraction (co-equal) lowest. As usual, parentheses may be used to override this ordering. Otherwise, as is normal, processing proceeds from left to right whenever terms of comparable order are involved.

The logical and relational operators are:

.AND.
.OR.
.NOT.
.LT.
.LE.
.EQ.
.GE.
.GT.

Mathematical, logical, and relational may be mixed in expressions. Mathmatical operators are highest order, Relational are second, and Logical lowest. In this case also, parentheses may be used to override the implicit ordering conventions. As before, whenever comparable terms are involved, processing proceeds from left to right in the normal manner.

Seasonal Adjustment

In addition to permitting you to make arithmetic and functional tranformations, DataView allows you to perform both monthly and quarterly seasonal adjustment, using the Census X-11 method; as an option, weekly seasonal adjustment is also supported.

At the most basic level, to seasonally adjust a series, simply execute the DataView command:

newseries=**SEAS**(oldseries)

where “new series” is the name under which you wish the seasonally adjusted results to be stored and “oldseries” is the name of the original unadjusted series. Depending upon the frequency set,

DataView will automatically execute the appropriate version of the seasonal adjustment procedure. The oldseries observations will be retrieved from an open Data Bank or Memory File. The results will be automatically stored in the Memory File or designated Data Bank, depending upon whether or not an UPDATE command is in force; this command is described in Chapter 4.

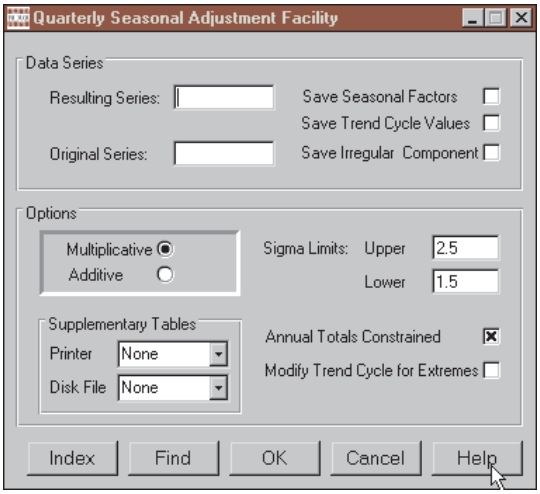

Figure 3-4. Quarterly Seasonal Adjustment

The alternative to issuing such commands is to click on the **Data** category of the Central Control Screen menu, then **Seasonal Adjustment**. If the observation frequency is set to Quarterly, the form shown as Figure 3.4 will be displayed. This form indicates the full range of options available. It also provides immediate access to context sensitive help.

Seasonal adjustment is interesting here for two reasons. The first is that the procedure allows you to adjust for seasonal variation, which in itself is often desirable. The second is that this procedure is performed using a powerful DataView facility: the capability to execute other programs concurrently as subtasks, including sharing data. To perform seasonal adjustment, DataView will actually use a separate program, which one depending upon the frequency of the data: MODX11Q (quarterly), MODX11M

(monthly), and MODSAWK (weekly). But while you need to be aware that these programs exist—since you must have them in the PATH or in the main Windows directory—as they execute you will be only vaguely, if at all, conscious that you are not using just another DataView internal command, so transparent is their operation.

Typing In New Data

Analysis can also involve updating series. You can use ordinary in-line commands to enter series observations, either to create new series or to revise or extend existing series. One method can be demonstrated by example:

CONS$,198402:2330.1,2334.5,2339.6,2348.1

This particular command enters values for the variable CONS$ for the second quarter of 1984 through the first quarter of 1985, overwriting any data already in the Memory File for the same series and dates; however, outside these dates any existing values are left undisturbed. Note that the series name is followed by a comma, a date and a colon, and that the observations follow after that colon, with single commas in between.

Commands providing observations on a particular series may run over several lines provided each line-to-be-continued ends with a comma to tell DataView that there is more to come. As indicated, the observations should generally be separated by single comas. Comas used as line continuation characters must be placed at the end of an observation value; individual numbers should never be split between two lines.

DataView also recognizes the * sign in a data entry command as indicating multiple occurances of the same value. For example:

DUMMY01,194701:0*40,1*4

directs that 40 zeros should be entered, beginning in the first period of 1947, followed by 4 ones.

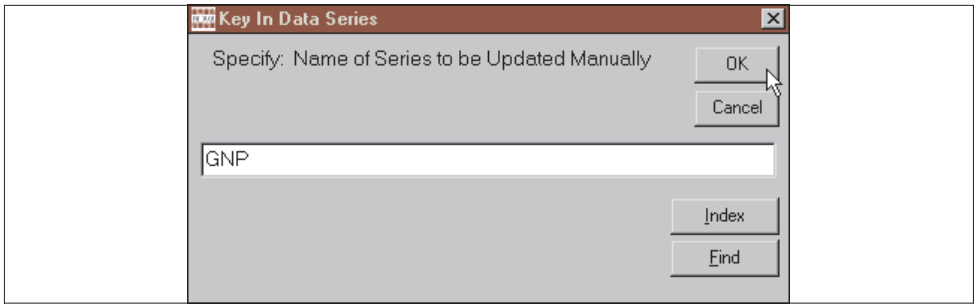

Figure 3.5 Series Update Selection Form

Series Update Command

Another way to enter or edit series observations is to click on **Data** followed by **Key In Series**, which causes the form shown in Figure 3-5 to be displayed. The series specified may be either an existing or a new one. Once the series mnemonic is specified, and you have clicked OK or pressed the <Enter> key a form similar to that shown on the next page as Figure 3-6 should be displayed.

The form shown in Figure 3-6 permits you to edit the series documentation as well as to add and revise observations. An aspect of this form that should be noted particularly is that you can use cut, copy, and paste to enter either text or observations, as appropriate. If, for instance, you happen to be using your Web browser in conjunction with DataView and wish to transfer an observation from a Web page to a field on this form, all that you

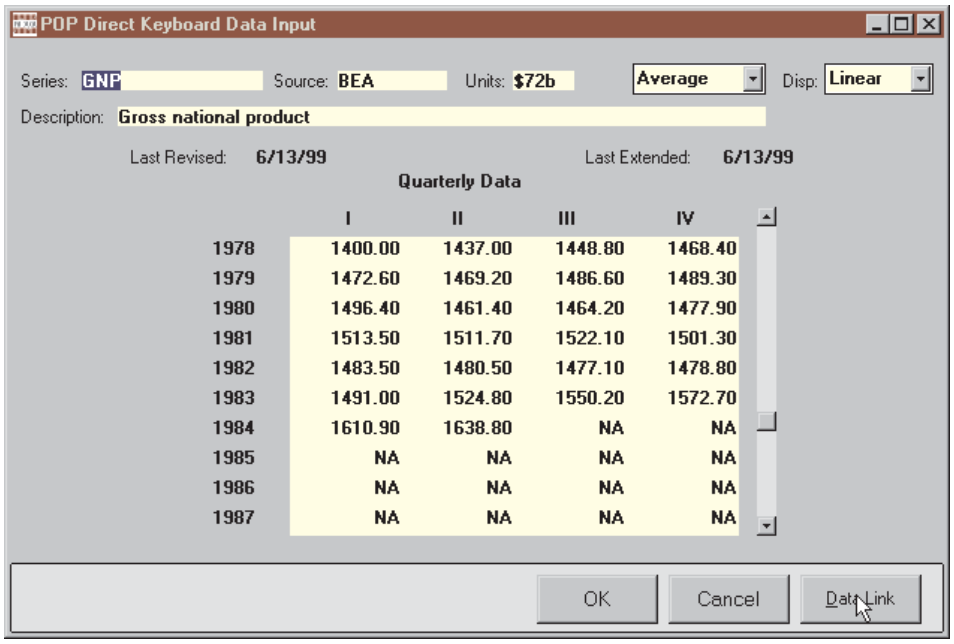

Figure 3-6. Series Update Form

need do is to place the point of your mouse cursor on the value you wish to capture, doubleclick, then right-click to cause a floating edit menu to appear, then choose **copy**. Next, place your mouse point at the location you wish to put this value, right-click your mouse once again, and then select **paste**.

Alternatively, this form can be used to move observervations on a series from a spreadsheet to DataView. Click the button **Data Link** and when the data link form appears, click on its help button for full details.

Data Entry Methods Compared

Which method of data input to use in a particular case depends upon a number of considerations, including the number of observations and the number of series. For a single series or if you would like to change series documentation, the onscreen editor shown as Figure 3-6 offers a good choice. Alternatively, if you

wish to enter a few observations each on multiple hardcopy series, the best choice is often to type these into an ASCII text file using the data entry format described on page 82 above. You can use DataView's internal text editor (click on **View | Macro**) to enter the values and then use the Run command to execute the macro. For example:

RUN NEWDATA

where NEWDATA.MAC contains the data. Among its virtues, this approach allows recovery from data file creation errors, since the data will be in a macro saved on your hard disk. Note, however, that in all cases the lines must be a **maximum of 80 characters**; a macro file simply simulates keyboard input.

In addition, if you will be revising data series, or adding observations at either end, it is possible to text edit your file, and then simply re-execute the original RUN command. Alternatively, you can create a new text file, containing only the revised and new observations. As long as you have adhered to the DataView ASCII input file conventions, you have a choice.

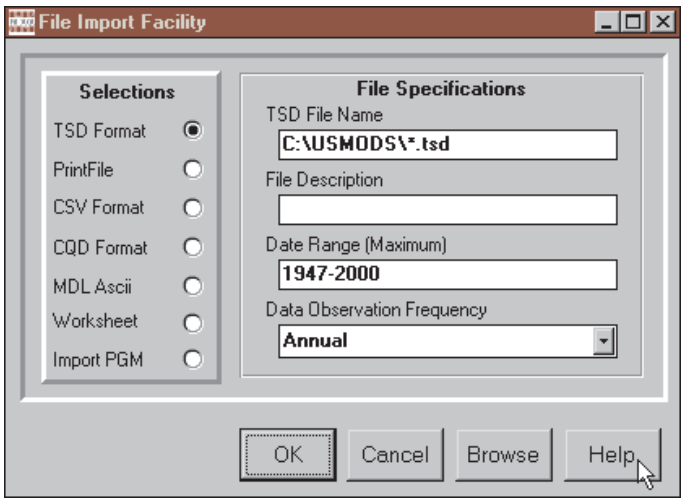

Figure 3-7. File Conversion Facility

DataView Import Facilities

But what if the data are already in machine-readable form? You may have a data file that is not in the DataView format or you may wish to download such data files from the Internet or some other source. In order to allow for these possibilities, DataView incorporates extensive file conversion facilities. From the Central Control screen click on Data then Import. You should next see the form shown in Figure 3-7. Click the help button for details.

Displaying Groups of Series

The TAB SERIES command allows you to produce a basic table that displays series in columns. One option allows you to select the series to be displayed by declaring the initial letters of their names. For example, if the series with names beginning with the letters D or E are to be retrieved from a Data Bank, use a command of the form:

TABULATE SERIES=D-E* FROM DEMOBANK

where the Data Bank name (here DEMOBANK) appears in the command; the Data Bank must already be open for access for this command to execute. If the series are to be retrieved from the Memory File, use a command of the form:

TABULATE SERIES=D-E*

Sometimes, you may wish to collect and display series from one or more Data Banks—or selected series from a particular bank that do not group alphabetically. In this case, one option is to use the Memory File as a collection point. The following sequence of commands illustrates the process.

First load the Memory File:

CLEAR MEMORY FILE
ACCESS DEMOBANK
SET FREQ=Q
SET DATES=7801-7902
FETCH POP
FETCH POPWA
FETCH LABF
FETCH EMPL

using the FETCH command described earlier. Then display the series as a column table by typing:

TAB SERIES=A-Z

As before, the absence of any bank name implies that the series will be retrieved from the Memory File.

Context-Related Groups

The TAB command just described references groups that are defined by series name or location, where the names are alphabetically contiguous and the locations follow one another. Alternatively, DataView permits groups to be defined by a **GROUPFILE**, which is an ASCII file containing the names of series; it must have the extent **.GRP**. In this case, the TAB command will take the form:

TAB GROUP=groupname

The characteristics of a groupfile are as follows: It must be a pure ASCII file, containing no control characters, except Carriage Return Line Feed. Each line must be 80 characters or less and

contain series names, separated by blanks. It can contain a maximum of 3500 series names. It must have the extent **.GRP**.

Groupfiles can also be referenced in COPY commands, EXPORT commands, and PRINT commands:

COPY GROUP=MYGROUP **FROM** memoryfile **TO** bank

EXPORT GROUP=MYGROUP **FROM** bank **VIA** TSD

PRINT GROUP=MYGROUP

where syntactically the principal distinction is the replacement of the word **SERIES** by the word **GROUP**.

The menu analogues of these command are all found under the **Data** category of the Central Control screen menu.

Some Important Data Management Issues

DataView is designed to work specifically with time series data. As a result, it naturally supports data of many observation frequencies, including weekly, biweekly, ten day, monthly, quarterly, semi-annual, and annual data; the annual data frequencies include both calendar and fiscal year.

The number of observations stored per series can be 1600 maximum; of these as many as 800 can be used at any one time. In the case of annual or quarterly data, the time span that 800 observations implies is too great to become an issue for many people—since very few time series span more than 100 years. However, 800 weeks is less than sixteen years and 800 months is less than 70 years. As you progress to higher data frequencies, you need to begin to consider exactly what is implied.

The first thing you need to take into account is disk space. If you allocate 1600 observations per series record, then data banks and memory files will tend to become large even if they contain only a moderate number of series. Physically, each series takes up a record and 1600 observations implies a record length of over 6K of storage space; this means that a 100 series data bank will only with difficulty fit on a low density floppy disk. A 1000 series data bank will not fit on any standard size diskette, although it will easily fit on a Zip disk, Clik disk, or any such storage medium that has the capacity these do.

The second thing you should take into account is that a given date range implies many fewer observations at an annual frequency than at a monthly or weekly frequency. If you mix data series of different frequencies in the same data banks or memory files, you should create each data bank or memory file in terms of the highest frequency data it will ever contain: store first at least one high frequency series. In contrast, if you begin by storing annual data, you are likely to find that you cannot later store all the quarterly, monthly, or weekly data that you would wish.

The third consideration is that the way that you specify frequency and dates depends upon the particular frequency. In the case of annual or quarterly data, it is necessary to specify only:

SET FREQ=ANNUAL

or

SET FREQ=QUARTERLY

however, in the case of weekly data you must specify both the frequency and the day of the week. Weekly data are associated with particular days of the week. Moreover, some years will contain 53 weeks while others will contain only 52 weeks; for instance, the first day of the year occurs 53 times in that year and

this must be taken into account. If January 1st falls on a Friday, then there will be 53 Friday weeks that year. In the case of weekly data, the SET FREQUENCY command is:

SET FREQ=WEEKLY(kk)

where kk designates the first two letters of the day (MO, TU, WE, TH, etc.). You must include both the parentheses and the day designation. Note also that you generally will not be able to use weekly data of one day with weekly data of another day in the same transformation—technically these are two different frequencies.

When working with weekly data, you may either specify dates or the year and week. That is, the dates command can take the form:

SET DATES=mm/dd/yy-mm/dd/yy

or

SET DATES=yyww-yyww

where, in the first case, mm is the month, dd the day, and yy the year. In the second case, yy is the year and ww is the week of the year. Thus 12/31/89 is December 31, 1989 and 8822 is the 22nd week of 88. It does not matter which form you use, but you cannot mix them in a single command; for example, it is **not** valid to specify:

SET DATES=1/1/70-8842

Frequency Conversion

DataView is not only designed to handle data of different fre-

quencies, but as noted in passing earlier to make it easy for you to convert from one frequency to another. You can, for example, automatically convert quarterly data retrieved from DEMOBANK into annual averages by typing:

```
AUTOFREQ AVG
CLEAR MEMORY FILE
SET FREQUENCY=ANNUAL
SET DATES=1965-1984
ACCESS DEMOBANK
```

Because the SET FREQUENCY command specifies annual, all observations retrieved will be converted as they are retrieved, allowing series from DEMOBANK (and any other open Data Banks or Memory Files) to be treated as if they were annual—printing or plotting them or applying any function or transformation. For example, once having issued the above commands, to display annual average GNP growth and inflation, type:

```
PLOT APCT(GNP),APCT(CPI)
```

AUTOFREQ AVG is often appropriate when the original data are expressed in levels (such as millions of people or index base 1967=100) or when they are measured at annual-equivalent rates (for example, $billion per year). However, in other circumstances, other types of conversion are needed. DataView's alternatives include AUTOFREQ SUM (adding up weeks, months or quarters) and AUTOFREQ FIRST or LAST (for example, for an opening or closing stock). The HELP file and System Reference Manual describe yet other options, including the SELF option, which allows the conversion method to be determined series-by-series as they are retrieved, provided that the Frequency Transformation Criterion (FTC) is included in the series' documentation.

Otherwise, if different conversions are required for different

variables series may be loaded into the Memory File individually using explicit conversion commands: for example

```
MIDYRM1$=SELECT(2/4,M1$)
FINALM1$=SELECT(4/4,M1$)
AVRGEM1$=FTRAN(AVG,M1$)
GNP=FTRAN(SUM,GNP)
EMPL=FTRAN(AVG,EMPL)
```

The SELECT(i/j,seriesname) command shown here, incidentally, is used to select the ith of every jth observation, and is described fully in the relevant section of DataView's online help facility.