UsingMcalc64 and the Dyad Ratios Algorithm

James A. Stimson

Updated September, 2021

1 What's New in Mcalc64

Most important, reflected in the name, Mcalc64 is a Mac 64 bit application. That means that it will run on Mac 64 bit operating systems 10.15 (Catalina), 11 (Big Sur) and 12 (Monterey, not yet released) and yet newer versions to come in the future.. Mcalc, the still available 32 bit legacy version, will run on Mac OS X 10.4 to OS X 10.14 systems. These systems, released from 2004 through 2018 were 64 bit systems that also supported older 32 bit products. The newest systems, in contrast, are 64 bit only.

Also new in Mcalc64 is the ability to read the most recent Stata "dta" data files (through the most recent versions of Stata, including the current version 17). The older version read dta files through the Stata release 12 version.

2 Mcalc and Wcalc Versions and Flavors

Mcalc64 runs in the Mac console. That text interface consists of prompts to the user for choices and information and user responses from the keyboard. The console has no point and click functionality. This current version updates Mcalc, a version written for the Mac OS X operating system. In contrast, Wcalc5 and Wcalc6 have a graphical interface for various versions of the Windows operating system. The two different applications require the same data and choices and yield the same outcomes. It is the look and feel that are different.

A Beta Version This code has been extensively tested in house and has no known problems. My testing staff consists of one person, myself (also known as the programming staff). But users will expose the code to a vastly more varied set of circumstances than I can envision in my testing and will find problems I have not encountered. Going back three versions,

about 15 years, no bugs have ever affected the correctness of results. The Dyad ratios algorithm code is rock solid and largely identical to all previous versions. The problems that are encountered are typically that something in the user's data cause the program to hang up and not run at all—and with a cryptic error message that is not helpful.

Notifying me that it won't run is a waste of time, because I can no more diagnose those cryptic messages than you can. But if you notify me and send me the data file that produced the problem, I can quickly diagnose and fix the issue.

2.1 Other Supported versions

1. Wcalc6, a product of the Microsoft .net (dot-net) programming system, runs on the newest versions of Windows.

2. Wcalc5 is still supported for use with older Windows versions, Windows 95 to XP, inclusive. It is written in Visual Basic 6.

3. Mcalc is a non-graphical version for the Macintosh. It runs in a DOS-like console window and is equivalent, except for the user interface. It is written in c++.

4. extract.r is a function for the R platform. Except that it is limited to estimating a single dimension, it has equivalent functionality to the other versions. It is written in R.

   All versions return the same results on test data within the limits of the rounding error inherent in iterative estimation.

3 What is Mcalc64?

Mcalc64 implements the Dyad Ratios algorithm (Stimson 2018) for building a continuous regular time series from the scraps of dated survey results that are typically available for public opinion analysis. It emulates the logic of principal components analysis in most regards but, unlike principal components, it does not require that all variables have a complete set of cases or, indeed, anything close to it. Input is survey data expressed in a summary score, e.g., percent liberal responses, which is dated and includes a number of cases (although the program will accept 0 for N when it is unknown). The use of variable names indicates which survey results are based on comparable questions and sampling techniques and which are different. Mcalc64 is written for Mac OS in c++ using standard library 11 extensions.

In order to estimate the underlying tendency as a time series the program needs four things for each survey result:

Variable Name A common variable name (case sensitive) indicates to the program which survey results are produced by the same questions at multiple times. There is no issue of degree here. The results are either comparable—meaning that it is meaningful to compare the different results at different times—or not. Stretching a metaphor, if apples and oranges have something in common, that can be determined by their empirical correlation. But you would call one APPLES and the other ORANGES in order not to falsely assume that they are the same thing.

Date The program expects to read a date which decodes to a full month, day, and year specification. That is best achieved in binary input files, such as the Stata dta file. Nearly all software treats dates as serial counts of the number of days that have transpired since some base date. Much of the Windows world is based on a system in which the zero day is December 31, 1899 and the current date is the number of days that has transpired since then. This is a foolproof system that does not depend upon culture. So for example, 40390 decodes to August 1, 2014 and it doesn't matter which of the dozen or so common ways of writing dates are used. A European, for example, would commonly write 1/8/2014 and his or her software would understand that it means August 1, not January 8, because it would always be 40390.

The program supports text file input. But one of the reasons—and there are several others—that such usage is discouraged is that string representations of dates, e.g., "8/1/2014", are culturally ambiguous. Is that August 1 (in the U.S.) or January 8 (in France?) For text file input the program correctly translates ONLY U.S. dates written in the Month/Day/Year format.

And full dates are required. Even though you may only care about, say month and year, in order to build serial dates all three pieces are required. The day portion clearly doesn't matter in this example, but the program does not know how to translate a partial date, say "5/2014". It requires all three pieces.

Summary Score Survey marginals come with as many categories as proposed answers to a closed ended question plus the usual assortment of missing data categories. For analysis those multiple responses need to be collapsed into a single score that is an indicator of the concept in question. For my original implementation for Public Policy Mood, for example the summary is:

$$Percent\ Liberal = 100 * [\frac{liberal\ responses}{liberal\ responses + conservative\ responses}] \qquad (1)$$

where neutral and missing categories do not enter in the computation.

The summary score, which will typically have a fractional component, raises a cultural issue. Numbers with fractions are written using a decimal point in the U.S., as in 43.9. Whereas usage in Europe uses the comma, as in 43,9. The program expects the American format for text input. (For binary input there is no problem because the system reads floating point numbers, something like 439e02 (except binary rather than decimal) with no decimal point or comma.)

Number of Cases Why number of cases? Mcalc64 uses number of cases for one limited purpose, which often doesn't even arise. The case is where more than one survey result is available within an aggregation period. The limited purpose is to compute a weighted average of the two or more results. Sampling theory tells us that the best estimate from two or more surveys is the one that weights by the number of respondents in each. So estimates of say 45 and 55 from samples of 500 and 1000, are computed as (500*45 + 1000*55)/ 1500 = 51.67, rather than the simple average of the two scores, 50, effectively weighting all respondents equally. Sometimes N's are unknown, in which case entering a 0 arbitrarily produces an estimated N of 1000. Or entering zero for all cases effectively does unweighted averages.

Although entering zero's where number of cases is unknown is OK, simply omitting the information from the file is not. The program expects to find a number of cases variable and will crash if it does not.

For text input do NOT include commas in the numbers, because 1,000 does not read as one thousand; it reads as two numbers, 1 and zero. There are no commas in binary representations of numbers and so that issue does not arise.

4 Usage Choices: Input Data

Mcalc64 supports two kinds of input data, text and binary. The latter is strongly preferred. The problem of text input—and this is true of all software—is that the format is ignorant. That is, the machine needs to read it blindly, according to some prior expectations and restrictions because the text format does not permit instructions in the data stream which inform the software of how to read it correctly.

4.1 Text Input

Although deprecated, text input is still supported for downward compatibility. Text input must be prepared specifically for the demands of Mcalc64. It wants an input file with exactly four columns of data, variable name, date, index, and number of cases in that order. All four must be present for every line as a missing item causes the program to misread the whole file after the missing item. The missingness issue most often arises with N. 0 is an acceptable input for N, which is interpreted as unknown and arbitrarily assumed to be 1000. Blank, however, is not acceptable.

The program does not require a header line and does not use it, if present. But it will tolerate (and just skip over) the line if it is present as the first line of the file. Detecting the header line in order to skip it can be difficult, so it is best not to have one.

Free format data input (the common file types are ".txt," ".prn," and ".csv") does not tolerate stray characters that may be (mis)interpreted as delimiters. In particular, that means that variable names may *not* contain embedded blanks. Because the space character is a common delimiter, a variable name like "my var"would be misread as two items, "my" and "var" and is not allowed. ("my_var" would work fine.) Similarly, input numerics may not a contain a comma, as in "1,000". Common delimiters are the "space" character (Ascii(32)), the tab, (Ascii(9)), and comma, (Ascii(44)).

The input data file is a space-, comma-, or tab-delimited ASCII file with one line for each set of survey marginals. That line contains, in order, VARNAME, DATE, VALUE, N.

Example line:

AIDMIN, 5-01-1968, 37,1479

Note that blanks between items are allowed, but not required. Excel tab-delimited txt files will be correctly formatted, but aren't the only means of producing the file.

VARNAME is an alphanumeric string identifying a series. If it complies with the normal rules for computer variable naming, it requires no punctuation except a following comma or tab to delimit it from DATE.

DATE is a string in one of the common forms: mm/dd/yy, mm/dd/yyyy, mm-dd-yy. These have in common that the order is always month, day, year. Also, they always contain two delimiters, either "/" or "-". Leading zeros on month and day are allowable, but not required: 05/ and 5/ are both "May." Two digit year fields are presumed to be 20th Century dates, and 1900 is added to them.

Full dates must always be present, even if they are more specific than needed (e.g. in annual data the user doesn't care about month or day). Where specifics are unknown (and insignificant) any legitimate date will do. Dates must be legitimate dates, i.e., those which are actually found on historic calendars. February 29, for example, is only acceptable if the year is actually a leap year.

VALUE is a single indicator of the concept to be measured. It could be one of the response categories from a survey item, e.g. percent approve, or an index of multiple responses. In any case, the value must NOT include zero or negative values, as these violate the ratio-based structure of the algorithm. Near zero values ought also to be avoided. This is a rounding error issue. 90/1 is twice as big as 90/2, for example, but a difference that could easily occur from sampling error and results reported rounded off to whole numbers. Value can be represented in any standard numeric format.

As in principal components analysis, the polarity of the resulting dimension measure is determined by the dominant direction of coding of the raw items. Coding all in the same direction, e.g. liberal, approving, or whatever, will insure against a solution opposite

of what is intended. Negative covariance between item and scale, when observed, is treated as meaningful (a revision from earlier versions of the algorithm.)

N is the number of cases in the survey. It is used by the program only for the limited purpose of weighting in the case of multiple instances of a variable within an aggregation interval. When either (a) N is unknown, or (b) the user prefers unweighted averages, enter 0. A zero is replaced by 1000 for the averaging process, a fact which matters only if N is known for some instances of a variable and unknown for others within an aggregation interval. Even if N is unknown or weighting is not desired, some value must be present or the file will be misread.

Case Order: Text input files must be sorted by (1) VARNAME and (within that) DATE. The program searches for break points, both between variables and between aggregation intervals within variables, as it reads. In the current version sorting is automatic and you need not concern yourself with it. VARNAME is case specific, e.g., "Clinton" and "clinton" are assumed to be different series. The order of VARNAME's within the file has no consequence.

Here is a small piece of an input data file, illustrating the format. (This is a tab-delimited file, which is why the columns align visually. Space delimited or comma-delimited work the same, except for appearance.)

```
...
ABCWP 1/30/2008 33 1249
ABCWP 1/30/2008 33 0
ABCWP 2/28/2008 32 0
ABCWP 4/10/2008 33 1197
CBSNYT 2/10/2001 53 1124
CBSNYT 3/8/2001 60 1105
CBSNYT 4/4/2001 53 660
CBSNYT 4/23/2001 56 921
CBSNYT 5/10/2001 57 1063
CBSNYT 6/14/2001 53 1050
CBSNYT 8/28/2001 50 850
CBSNYT 9/11/2001 72 1041
...
```

Note these characteristics: (1) the variable name has no embedded blanks, (2) the date is a Month/Day/Year format, (3) the index variable just happens to be whole numbers (because that is what is usually reported for presidential approval), but numbers with decimal points and fractional components are allowed, and (4) the number of cases is always present, but the zeros for two cases indicate that it is unknown. This file was actually written by Excel, using the tab-delimited txt format.

4.2 Binary (.dta) Input

ASCII text input is an extremely limiting format. To read correctly it requires that the user prepare a file containing exactly four items per line in the required order. This process is both tedious and error prone.

The logic of binary input is to allow any number of variables to be read, from which the four to be used are selected at run time. Rather than specify the order of input variables, as in the text version, for binary files the variable name defines contents. The program looks for "variable" or "varname" to define the string variable which holds variable names. A variable named "date" (and which is a Stata data type Date) defines the date item. "Value" or "index" define the column of numeric values, and "n" or "ncases" defines the column with sample size information. Header names are not case sensitive, so varname = VARNAME = Varname all point to the same variable.

If any of the four standard variable names is not found at run time, the program provides a numbered list of variable names in the file and prompts the user to select the appropriate ones. It is time saving to just use the standard variable names.

Stata Dates When you create a Stata file by any of several means, if you explicitly inform Stata that a particular variable is a date, if will translate it from the form you have, often a string variable like "8/1/2014", into its own internal representation of dates. Mcalc64 knows how to translate Stata dates, so you do not need to understand how it is done. But for the technically inclined, a Stata date is a count of the number of days which have expired since January 1, 1960. Any date which ever did occur or ever will can be translated into a count (positive or negative) relative to 1/1/1960. So that single number is readily translated into month, day, and year components or displayed with a familiar string representation of dates. You never see the number which is actually stored. Stata determines how the date is represented internally and it varies, depending upon the size of the number.

So in order to be read correctly, it must be an actual Stata date, and not a numeric such as year or a string representation of dates. So if you have a variable, say myyear, standing for the year and you don't care about month or day, then:

gen date=mdy(1,1,myyear)

will produce a Stata date that is January 1 of the given year. If you translate from one kind of file, say a spreadsheet, into Stata using file translation software, anything recorded as a date will be correctly translated. But if you do the same thing by cutting and pasting, Stata will record the date variable as a string, and *not* a date. (If you display the variable, Stata will report its data type as "str" and a number, meaning string.) To translate a string date into a Stata date takes two Stata commands. (Assume that you have a string version of a

date named mydate.)

gen newdate=date(mydate,"MDY")
format newdate %td

and you have a real Stata date which will be read correctly by Mcalc64. (You could name the new variable date, but I chose newdate instead so as not to confuse the variable name date with the function of the same name.) The second parameter to the date function informs Stata what order a string date uses for month, day, and year. A European would enter "DMY".

5 Topic Query

The query facility permits selecting a subset of a data file defined by content at run time. (This operates independently from selection by date, which has always been part of Mcalc64. Thus one may select by topic, by date, by both, or by neither.) This permits selection from data files containing materials not wanted for a particular analysis.

Topic query is performed first. The program then updates beginning and ending date screens based upon the cases actually selected, rather than the whole file. Then it is appropriate to narrow the date range, if desired. (It isn't possible to expand it, because there are no selected cases outside the computed range.)

Query assumes that a file contains one or more numeric variables which are content codes.[1] And so the task of the user is specify which variable that is and then what the selection criterion is.

Example: mycode=13 (where mycode is a variable name in the file).

The query prompt is below as it appears on the screen:

```
Case Selection:
Enter a selection criterion of the form VARNAME OPERATOR VALUE,
where (1) VARNAME is one of the column names above,
(2) OPERATOR is one these 6:  <, >, <=, >=, =, or <>
(3) VALUE is a number.

Example: topic<10
```

[1] String variables are not supported for query selection.

```
Conditions may also be joined with the boolean operators  AND  and OR as in:
topic=1 and subtopic<3

Specifications are not case sensitive.

Specify a condition for selection:
```

[End of screen display]

Parentheses are useful to eliminate ambiguity about the order of evaluation. For example:

topic>11 and topic<21 or subtopic=17

is ambiguous about order of evaluation (and the ambiguity is resolved by order of evaluation rules that are pretty technical.) Whereas:

(topic>11 and topic<21) or subtopic=17

resolves the ambiguity and puts you in charge.

The query specification can be as complicated as is required, with any number of variables, parentheses, and Boolean operators ("and" and "or").

Query is not available for text file input because the same data selection for text files can be carried out in the file preparation process.

6 Auto Sorting

The logic of Mcalc64 requires that records be sorted by variable name, and within that, by date (ascending). (Variables can be in any order.) In older versions of Mcalc and Wcalc that required that the user perform the sort as part of file preparation. Failure to sort correctly produced nonsense analyses. On input Mcalc64 detects the actual sort ordering of the file. If it does not comply with the dates within variables requisite, the file is automatic resorted to be in the correct oder. Variables are sorted in alphabetical order and dates within variables are ascending.

Beware that variable names are case sensitive in Mcalc64. Thus two names which are identical except for capitalization are treated as two different variables for analysis.

wk1 Previous versions of Mcalc have supported input from old .wk1 spreadsheet files. After Microsoft Excel dropped the wk1 file as an optional file format, wk1 support has been dropped on the assumption that it is not being used. I would like to support Excel .xls and .xlsx files, but the format is actually a dozen or so variations, each of which quickly becomes obsolete with the newest release of Excel. Keeping up requires a programming staff that doesn't exist.

7 What Is a Case? It Depends

Input to the dyad ratios algorithm consists of cases, of which there are two kinds. Cases may be thought of as lines in an input file. But the act of aggregation into regular periods, years, months, quarters, days, etc. changes the definition of cases, yielding a smaller number of cases actually used. Whenever two or more cases come from the same aggregation period, their average becomes a single estimate for that period and the number of effective cases is reduced. Orphan cases—those that have only a single time point after aggregation into periods—contribute nothing to the solution. In order to observe change, there must be data for at least two periods. Having orphans in the input file is harmless. Lacking the ability to produce dyad ratios, they simply contribute nothing to the solution. The effect is exactly the same as if they were not present.

How does aggregation affect the numbers of cases? Imagine the following example. Let's say that our data for a particular variable consists of survey results on five dates, 11-1-2020, 11-17-2020, 12-13-2020, 12-27-2020, and 1-21-2021. If our aggregation period is daily, then none of these results fall on the same days so we have five cases after aggregation. If aggregation is monthly, then we have three cases, 11/2020, 12/2020, and 1/2021. If it is quarterly we have two cases, 2020:4 and 2021:1. And it is is annual, we also have two, 2020 and 2021. If it is biennial, it depends whether the initial year of the biennium is even or odd. If it is even (2020 that is) then we have an orphan case. If odd, we have two cases.

The N's that appear in printed output are numbers of cases *after* aggregation into periods, so they will not correspond to numbers of cases in the file.

8 Running Mcalc64: Program/User Interaction

1. Click on Mcalc64 on a Mac that supports 64 bit applications. (That is versions OS 10.4 and newer.)

2. Mcalc64 will open a console window on your desktop and prompt you to enter a file name. This is an example of what you will see:

```
Mcalc64 for Macintosh

Reuse last file: /users/jamesstimson/dropbox/public/BushJob.txt
Enter y to reuse:
```

The very first time you use it it will not prompt you to reuse a file. After that it will. If you are entering a file name for the first time, it will need to be in the format illustrated from my machine, that is a full path specification beginning with /users. And yes, entering file names in console text mode is tedious.

If you are reusing the file, just enter y (or Y) for yes. (Entering n or N will produce a prompt for the new file name.)

3. If you enter a text input file, you will see a dump of its contents. If you enter a dta file, you will see miscellaneous information from the file header. You need not do anything in either case. These are printed largely for diagnostic use by me in case of a file read failure. Then you will be prompted:

```
Choose aggregation interval
Daily (d)
Monthly (m)
Quarterly (q)
Annual (a)
Multiple Years (o)
Input d, m, q, a, or o:
```

and you just choose the letter for the right aggregation interval. In the case of option "o" (other) you will then be prompted to enter the multiple.

4. then (Only for dta file input)

```
Select input records by topic codes? (y/n):
```

and you enter y or n. If "y" you will see a full screen of documentation about how to enter the selection criterion.

5. Next the app will inform you of the earliest data point found in the file. This will be expressed in the aggregation units you have chosen.

```
Earliest Date from input file:
Year: 2001
Month: 1

You may use the earliest date or enter a newer one. Enter y to accept date:
```

Normally you will accept the app's recommendation. The exception that could occur is if that earliest item is an orphan, in which case it is not suitable for the beginning point because you need to start with a usable item. You could also choose a later starting date for substantive reasons.

6. Then the parallel choice of ending period:

```
Latest date from input file:
Year: 2005
Month: 4

You may use the latest date or enter an older one. Enter y to accept date:
```

7. Then a choice of how many dimensions to estimate:

```
Number of dimensions to extract. (Default = 1). Enter 1 or 2.
```

Estimation of a second dimension is performed by first changing the input matrix x(i,k) into a residuals matrix by first regressing all the x's—treated as a dependent variable—on C_1, the estimated latent first dimension. An intercept shift is then performed to remove zero or negative values. And then the estimation procedure is repeated for the now residualized data. The result is thus interpretable as an orthogonal process. Whether or not a second dimension of public opinion *should* be orthogonal is a theoretical matter. Some skepticism is appropriate.

8. Then you have a choice of whether or not to smooth intermediate results. I recommend opting for smoothing on.

```
Smoothing on or off (Default=on). Enter y/n for smoothing on :
```

9. The final choice is how to name your output files. The default is to use the filename portion (of filename.ext). The app will overwrite files on subsequent runs without warning, so If you wish to retain multiple files, you need to rename files.

```
Use [YOUR FILE NAME] as root name for output file? Enter y to accept or another
name to change.
```

9 The Dyad Ratios Algorithm

What follows is a documentation of the logic of the dyad ratios algorithm. It is written for the technically curious and not at all requisite for using the software successfully.

Data Begin with data. After aggregation into T regular time periods, we have a matrix of N items for T periods, x(i,t) where i indicates variables and t indicates period. Because no survey item is ever posed at every consecutive time point in the sample most of the matrix is missing data, represented as zero. We assume that items are positive numbers scored to represent the concept in question. It eases exposition to assume that all items are scored in the same direction, that higher scores indicate more of the concept and lower scores less of it (regardless of the polarity of the underlying survey question).

Our definition of a variable or item is that it is the same question, the same response options, the same sampling design and typically posed by the same organization (although that requirement can sometimes be waived). Thus a common variable name implies that all of the cases of that variable may be meaningfully compared.

Ratios Define a dyad as values for the same variable, x(i,k) and x(i,l) for any two time points, k and l, where k is not equal to l. Then as a starting point we can say that the ratio

$$r_{ikl} = \frac{x(i,k)}{x(i,l)} \tag{2}$$

is a meaningful indicator of the concept C to the degree that item i is a valid indicator of C. (The validity issue is taken up below.) This assumption of meaningful dyad ratios is the foundation of the algorithm and the source of its name.

Thinking of data as ratios rather than variable scores has one major advantage. Whereas in general scores are not comparable across items, ratios are. Two variables will, in general, produce different scores. Because there is no science of question wording, we do not know what level of support or opposition each item should draw. If we had a full set of cases for each variable (as in principal components analysis) we could estimate the variable means and use that knowledge to compare across items. But we do not. Because of the missing values issue, we have neither a full set of cases nor a representative sample of them. Thus we cannot know item expectations. But ratios, r, have a known expected value across cases, 1.0. That common expectation justifies comparing across items.

But how to combine items? We have a multitude of dyad ratios, with typically large numbers for each time point in the series to be estimated. There are multiple ratios for each item. Time k, that is, has a ratio for every other time point that is available for item i. And then there a multiple variables as well. So information exists in abundance and the problem becomes simply how to combine it sensibly.

The problematic aspect of ratios is that they are relative information. What we want to know is the absolute level of our concept for some time t. But what we have tells us instead how t is related to t+k. *If* all variables were available for one or more common times, then the problem is easy. We make the ratios relative to those times and then we have absolute

information. But in general our real world data do not provide the convenience of a time point available for all variables. The easy option is precluded by the data we have.

Backward recursion There is no global solution for combining information across items. We could just ignore comparability issues and average across all the different ratio estimates for each case. That would probably produce a decent approximation of a good measure. But it depends upon an assumption, that we have a representative sample of time points for each item, that is known to be false. Recursion is a second best approach. Begin with the final point of the series, T. Having lost our metric information in the computation of ratios, we can assign an arbitrary value to this one time, say 100.

Now there exists a subset of items which include an available value for time T. For those items (only) we can estimate absolute values for each item and each t by simply projecting the known value at T (100) onto the ratio of T to other time points. If, for example, item i has a T to T-1 ratio of .92, then the estimated score for that item at t-1 is 92. Then we can average across all existing cases that have non-missing values for T and T-1 to get an estimate of t-1. (We are still assuming perfect validity for items here. That issue will be dealt with below.) All earlier times are also projected for later use.

For the latent concept C denote our estimated value for case t-1 as $\hat{C}_{t-1}$. So now we know two values, $C_T = 100$ and a data determined value for $\hat{C}_{T-1}$. The data determined value reflects the true ratio of T and T-1 estimated from all of the data which exists. No missing values enter the computation and no existing values are ignored.

Now we can repeat the process for comparing T-1 to T-2, but this time using the estimate for T-1, $\hat{C}_{T-1}$, rather than an arbitrary number, for the value at T-1. Following this procedure we eventually work back to time 1, the beginning of the series to be estimated.

When using backward recursion later periods tend to dominate the solution. Each later value has influence on the values of early items, but not the reverse. Also the estimates are not unique. Reversing the order and starting with time 1 and working forward – forward recursion – produces a similar, but not identical, set of estimates. Forward recursion has the reverse weighting of backward, early items contribute more to the solution than do later ones.

So we end up with two estimates of the latent concept C, C forward (C_F) and backward(C_B), which are equally valid time series. Averaging the two (for each time point) accomplishes two things, (1) it uses all available information for the solution rather than using one and ignoring one, and (2) it corrects the weighting effects to produce a summary score in which all items weight equally in the solution.

Smoothing Now we face a choice. We could just average the computed values of C_F and C_B. But we know something from sampling theory which suggests that we could do better. Sampling theory tells us that if nature produced smooth outcomes—i.e., if opinion change were gradual and regular rather than abrupt and jumpy—then our observed estimates of it would be noisy. Because the data points are the result of survey samples, they will capture the true level of the phenomenon while adding or subtracting a small error to each due to sampling. So if nature were smooth, we would still not observe smoothness due to sampling errors.

Thus the choice: in estimating C_F and C_B do we prefer those which are strictly data determined (and therefore also sampling error determined) or a smooth approximation based upon the prior knowledge that nature is smoother than empirical estimates of it? In my view, which you need not share, the smoothed approximation is superior to the data-driven estimates.

The particular smoothing model chosen is exponential smoothing. It has the virtue that it is senstive to how much noise in present in the data series (and will not alter an already-smooth series). The exponential smoothing model is:

$$y_t = \alpha x_t + (1 - \alpha) x_{t-1} \tag{3}$$

where y is the smoothed version of x. The intuition is that if the past, x_{t-1}, provides any useful information for predicting y_t, then some portion of the variation in x_t is a deviation from the smooth path of x. This is seen in zig-zag behavior, where the series tends to return to normal levels after extreme movements away from them.

The α parameter is estimated by minimizing within sample forecast error. Thus it is fully determined by the data. Exponential smoothing has the desirable property that it will not oversmooth. If the data are already smooth, a situation that often occurs with annual aggregation levels, then α converges on 1.0 and $y_t = x_t$. Smoothing occurs in both forward and backward directions in time, the result of which is that the raw data series become exponentially weighted moving averages of past and future values.

Smoothing operates on the raw values of C_F and C_B during estimation. That means, in effect that the smoothed value is presumed to be a better measure of the true level of the series than is the original, and that it is the smoothed values of the series that drive the ultimate measure. The impact of smoothing varies in direct proportion to the apparent randomness of the series. Where the original series are highly patterned, the impact of smoothing is rarely discernible. Where, in contrast, they exhibit a good deal of period-to-period zig-zag fluctuation, the effect of smoothing is large.

What consequence? Smoothing has a big (and helpful) effect on periods in which data are

relatively thin and usually modest effects when data are rich. This is to be expected because having multiple estimates of a quantity averaged together (when data are rich) produces natural smoothing, the Central Limit Theorem in action.

Validity Estimation The issues that arise is validity estimation in the dyad ratios algorithm are essentially the same as the validity issues in principal components analysis. Stata, for example, offers three options for validity estimation, (1) assuming perfect validity—essentially ignoring the issue, (2) estimating from the R^2 of multiple regressions of item i as dependent on all other items, and (3) iterative estimation. These amount in PC to treatment of the main diagonal of the input matrix, that it is (1) 1.0 for all items, (2) R_i^2, or (3) a convergence result where μ_i^2 (validity assumed for item i) becomes equal to $\hat{\mu}_i^2$ (validity estimated from the squared loading of $\hat{C}$ on x(i).) It is this third approach, iterative estimation, that is implemented in Mcalc64.

In the theory of vector decomposition mathematics if you could somehow "know" the right values of the validity estimates for each item, μ_i^2 (the proportion of all variance in the item i that is shared with the concept C) then the estimate produced after estimation, the squared correlation between the latent factor $\hat{C}$ and the raw item would be the same value, $\mu_i^2 = \hat{\mu}_i^2$. That theory provides a solution criterion. Iterative solution is reached with μ_i^2 estimated from the previous iteration differs by less than a trivial amount (.001) from μ_i^2 estimated from the current iteration for all i. There are N such estimates and the solution requires that all N be less than .001 different for solution.

Where validity comes into play in dyad ratios is that the estimate for each time point is, instead of a simple average of ratios, r_i,

$$\hat{C}_t = \frac{\sum_{i=1}^{N} r_i}{N} \tag{4}$$

a weighted average of ratios weighted by item validity.

$$\hat{C}_t = \frac{\sum_{i=1}^{N} \mu_i^2 r_i}{\sum_{i=1}^{N} \mu_i^2} \tag{5}$$

10 Output

Output I: rootname.csv The csv file contains date information as appropriate for the type of aggregation selected and the value of the estimated latent variable, a "factor score"

in common usage. On many systems the csv file type is "owned" by Microsoft Excel, so that double clicking this file will display it in an Excel sheet.

For daily estimation, the output includes a 5 digit integer date, followed by a text string date, followed by the estimated value for that date. The integer date will convert to a date variable in spreadsheets and database managers, which will be more convenient for some purposes than the text string. After input (e.g., to Excel) select the date column and format it as a date, and the integer values will be replaced by a date representation of your choice.

Output II: rootname.log The log file consists of four sections, (1) a brief header documenting the estimation data and options, (2) an iteration history, (3) item information (name dimension loadings, means and standard deviations), and (4) an Eigenvalue estimate of model fit.

Iteration history includes the convergence estimate, the cutoff criterion, number of series in the analysis. Also "F-B Reliability" is a product moment correlation of the two independent estimations (forward and backward recursion) of the latent series. AlphaF and AlphaB are separately estimated exponential smoothing α's for forward and backward recursions.

The item information includes the number of time points for which the item is available (after aggregation into intervals) and the item-scale correlation at solution. This information is interpretable as a factor loading. Its square is interpretable as a validity estimate.

The Eigenvalue estimates emulate Eigenvalue analysis from principal components analysis, but differs because here each item contributes only as much variance to the solution as it has cases. So an item which has non-missing values for k time points where there are T in the analysis contributes a variance of k/T (rather than the 1.0 of PC). Thus the total Eigenvalue is a small fraction of N and not a whole number. Interpretation of explained and unexplained is the usual.

11 Problems and Pitfalls

Mcalc64 always produces correct results when it runs, but sometimes it doesn't run at all. Problems are always related to the input data file. The program is extremely sensitive to data errors. It wants to read "N" cases each exactly of the structure outlined above. When it does not, it crashes and usually produces no useful diagnostic.

Nearly all problems are associated with text input. The program just doesn't tolerate what may seem to the user to be small mistakes, a missing number of cases, a comma where it doesn't belong (e.g., in number of cases) and so forth. Tongue in cheek advice: Take the

MMPI and if you score low on obsessive compulsive behavior deviation, do not attempt text input. If your standard of data preparation is "good enough," you will suffer countless frustrations.

Or, if your response is "Why can't my computer figure out what I want to do?" Remember that your computer is not as smart as the family dog.

With dta file input most issues arise over the treatment of dates. Read carefully what I have written about handling dates. Another variation is the perfectly reasonable decision to use numeric year for annual analyses. Logically that ought to work. Here it doesn't. Read what I have written about transforming year to date in Stata. It's easy.

And when you can't figure out what to do, send me a message (stimson@unc.edu) which includes your data file as an attachment. Please do not send a message without data, because it is frustrating to respond that there is nothing I can do without the data. I can usually find the issue quickly and usually it is the data file that comes back changed, not the software. (But gold stars may be awarded to users who locate actual bugs.)

Citation Stimson, James A."The Dyad ratios algorithm for estimating latent public opinion: estimation, testing, and comparison to other approaches," *Bulletin of Sociological Methodology/Bulletin de Méthodologie Sociologique*, SAGE Publications Sage UK: London, England, 137(1), 201-218.