SPREADSHEET LOGOLOGY ON THE PC

ANTHONY SEBASTIAN
San Francisco, California

Tapping the Computer's Power by the Non-Computer-Expert

Logologists who are computer programmers can exploit the computer's power to assist them in wordplay activities. Logologists who are not computer experts can share in that power by using commercially available, inexpensive, user-friendly spreadsheet programs designed for personal computers (PCs). Such programs, initially developed for manipulating numbers (e.g., financial modelling), have evolved the capability to manipulate non-numeric data--strings of characters other than integers exclusively--according to string-specific formulas.

The purpose of this article is to introduce the concept of spreadsheet logology to non-computer-experts, and to demonstrate the hospitality of the spreadsheet environment for logology by exemplifying a few specific applications. The article provides just enough details of both spreadsheet basics and how-to-do-it particulars (for the exemplified applications) to enable a motivated novice PC user to begin experimenting.

Spreadsheet Basics for Wordplay

On activating a spreadsheet, the computer screen displays a matrix of initially empty "cells" at the intersection of columns (labelled A,B,C,...Z,AA,BB,AC,...AZ,BA,BB,...) and rows (consecutively numbered 1 to thousands). Each cell's "address" is its unique column-row coordinates (e.g., A1, B9, AX45, IV666).

The user determines a cell's display-width and its contents, which can be a "string" (a character sequence such as a word) or a number. To determine a cell's content, the user moves a visible cell-pointer to the cell (via the keyboard's arrow-keys) and then types in (a) a string or number, (b) a formula (e.g., +"word"& "play" displays "wordplay"), or (c) a socalled @Function, which performs specialized manipulations (e.g., @LEFT("wordplay",4) displays "word", the leftmost four characters of "wordplay").

In the following sample spreadsheet, the words or numbers displayed in columns A and B were entered as such. Column C displays the "values" resulting from the computation prescribed by formulas entered into column C's cells. For expository purposes, those formulas are listed to the right. Careful study of those formulas reveals much about the way spreadsheets work.

Formulas can reference another cell's address, computing string or numeric values that depend on the referenced cell's value. Al-

	A	B	C	
1			wordplay	<-- +"word"&"play"
2	word	play	wordplay	<-- +A2&B2
3			word	<-- @LEFT("wordplay",4)
4		layout	out	<-- @RIGHT(B4,3)
5	spreadsheet		spread	<-- @LEFT(A5,6)
6		beer	reader	<-- @RIGHT(C5,4)&@RIGHT(B6,2)
7	wordplay		8	<-- @LENGTH(A7)
8	10	5	15	<-- +A8+B8

tering a referenced cell's value initiates recalculation of dependent cells automatically; the screen immediately updates. The "&" operator in string formulas is analogous to the "+" in numeric formulas; it concatenates strings. Cell formulas always begin with a + or @ (or similar operator).

Using @REPLACE

@Functions provide many ways to manipulate words. Consider @REPLACE. @REPLACE("word",0,1,"fj") yields "fjord". That is, @REPLACE replaces part of a word with specified letters as follows: @REPLACE(string1,num1,num2,string2), where string2 stands for the string of characters that are to replace a specified number of characters (specified as num2) in string1 beginning with the character at position num1 of string1. Note that positions in strings are numbered left-to-right beginning with zero; i.e., the first position in a string is position 0.

The following sample spreadsheet illustrates some of @REPLACE's word-manipulating capabilities:

	A	B	
38	bore		
39		bare	<-- @REPLACE(A38,1,1,"a")
40	n	born	<-- @REPLACE(A38,3,1,A40)
41	st	store	<-- @REPLACE(A38,0,1,A41)
42	sta	stare	<-- @REPLACE(A38,0,2,A42)
43	ingo	bingo	<-- @REPLACE(A38,1,3,A43)
44		bingo	<-- @REPLACE(A38,0,4,B43)
45	n	borne	<-- @REPLACE(A38,3,0,A45)

The strings displayed in column A were entered as such. The strings displayed in column B are the values computed by the formulas in column B's cells (listed at right for expository purposes). The formulas are the cells' contents, yielding the values displayed. Formula-lengths can exceed the cell's display-width.

Note that if num2 is set to 0 in a @REPLACE formula (see cell B45), no letters of string1 are deleted; string2 is inserted at the num1-specified position, in effect replacing nothing. That means @REPLACE formulas can be used also to insert letters into test words (e.g., converting "lien" to "linen"). By using "" (the so-called null-string) as string2, @REPLACE can be used to delete letters (quantity specified by num2).

@REPLACE Template for Letter-Insertions

The following describes how to use @REPLACE to construct reusable spreadsheet templates for generating insertion-networks (lien to linen) and letter-substitution networks (trope to tripe).

We want to test a word for all 26 possible letter additions at all possible positions in the test word (including the positions before the first and after the last letter). To accomplish that, we create a reusable template such that, each time the user enters a test word into a reusable test cell, the spreadsheet promptly displays parallel columns of 26 cells each, one column for every possible insertion position; the 26 cells of each column will show each letter of the alphabet inserted into the test word.

The following segment of such a template shows the results when "lien" is entered into the reusable test cell A141. Hits (actual words) are underlined for illustrative purposes.

	A	B	C	D	E	F
141	lien					
142	a	alien	laien	liaen	liean	liena
143	b	blien	lbien	liben	liebn	lienb
144	c	clien	lcien	licen	liecn	lienc
145	d	dlien	ldien	liden	liedn	liend
	.	.	.	.	.	.
	.	.	.	.	.	.
152	k	klien	lkien	liken	liekn	lienk
153	l	llien	llien	lilen	lieln	lienl
154	m	mlien	lmien	limen	liemn	lienm
155	n	nlien	lnien	linen	lienn	lienn
	.	.	.	.	.	.
160	s	slien	lsien	lisen	liesn	liens
	.	.	.	.	.	.
163	v	vlien	lvien	liven	lievn	lienv
	.	.	.	.	.	.
	.	.	.	.	.	.

Each of the 26 cells of a column contains the appropriate @REPLACE formula that inserts a letter into a given position in the

test word. Each position of the test word (0,1,2,3,...) has its own column (position numbers always begin with 0); each of the 26 insertion letters (a,b,c,d,...) has its own row. The number of columns in the user's template will depend on the maximal length of words one wishes to be able to examine.

The first few cell formulas of the template's first two rows are listed below; the remaining formulas are readily deducible.

```
B142: @REPLACE(A141,0,0,"a")     B143: @REPLACE(A141,0,0,"b")
C142: @REPLACE(A141,1,0,"a")     C143: @REPLACE(A141,1,0,"b")
D141: @REPLACE(A141,2,0,"a")     D143: @REPLACE(A141,2,0,"b")
```

@REPLACE Template for Letter-Substitutions

A similar spreadsheet template can be constructed for one-on-one letter replacements in a word, as in constructing word ladders. Again the template has one row for each letter of the alphabet, and one column for each letter of the test word, each cell having the appropriate @REPLACE formula. String1 is the same for every cell's formula (specify the cell address of the test word), and num2 is the same for every cell's formula (num2=1, so that one letter of string1 gets replaced). String2 is the same for every cell's formula in a given row, with a different letter of the alphabet for each row; num1 is the same for every cell's formula in a given column, with a different letter-position of string1 for each column.

After entering a test word, the legitimate derived words are noted, and each successively entered into the test cell to search for additional rungs of the ladder. An unused portion of the spreadsheet can be set aside for recording the results of each pass of a test word, and the accumulated record subsequently printed for a permanent record.

The ladder-building template also facilitates searching for hospitable words--words each of whose letters accepts a replacement that generates a legitimate word. Hospitable words are readily spotted, since they must generate at least one legitimate word in every column. Such a template could facilitate searching for the most hospitable word of a given word-length in a list of words.

Both ladder and insertion templates constructed to test longer words work for shorter words as well. If you enter, say, a four-letter word into the test cell of a template constructed for longer words, ignore the results in the columns beyond the first five (insertion template) or four (ladder template).

Beyond @REPLACE: Spreadsheet Templates for Words-Within-a-Word

There are many ways to use @REPLACE formulas to manipulate words and phrases in systematic and repetitive ways. In addition, there are many other string-manipulating @Functions. Furthermore, more than one @Function can be used in a single cell formula, adding to the manipulative power of spreadsheets for complex logological studies.

The spreadsheet's @MID function can be used to construct a template for generating words-within-a-word (pastille: paste, pill, etc.). A cell containing one @MID function extracts a specific letter or consecutive letter-sequence from a test word, as follows: @MID(string1,num1,num2), where num2 specifies how many letters from string1 to extract beginning at string1's letter position designated by num1. As before, the first character in a string is position 0. A cell containing @MID("logological",4,5) will display the string "logic".

To extract noncontiguous letters, combine @MIDs, one for each letter, using "&" as coupler: @MID("attend",0,1)&@MID("attend",5,1) extracts "ad" from "attend", for example.

The following sample spreadsheet illustrates ways @MID can be used. Columns A through C display strings or numbers entered as such. Column D displays the string values computed by formulas entered into Column D's cells (listed, for expository purposes, to the right).

	A	B	C	D	
1				play	<-- @MID("wordplay",4,4)
2				word	<-- @MID("wordplay",0,4)
3			wordplay	play	<-- @MID(C3,4,4)
4				word	<-- @MID(C3,0,4)
5	4	4		play	<-- @MID(C3,A5,B5)
6	0	4		word	<-- @MID(C3,A6,B6)
7				pa	<-- @MID(C3,4,1)&@MID(C3,6,1)
8	4	1		pa	<-- @MID(C3,A8,B8)&@MID(C3,A8+2,1)
9	4	1		pay	<-- @MID(C3,A9,B9)&@MID(C3,A9+2,B9+1)
10	4	1		pay	<-- @MID(C3,A10,B10)&@MID(C3,A10+2,B10)&@MID(C3,A10+3,B10)

With that background, we can construct a template that displays all possible interior same-order letter-sequences of a test word (i.e., where the letter-sequences are in the same order as those of the test word). For illustration, we construct a template for eight-letter test words (see illustration on the next page). The eight-letter template works also for shorter words.

We will use "pastille" as a sample test word (see Word Ways, May 1988, page 85), and set cell H1 as the test cell to receive it. Below H1, starting at H3 and ending at H247, we will enter the cell-formulas needed to generate every same-order letter sequence within the test word.

Before we can enter the formulas in Column H, we will first compose a separate @MID formula to extract each of the eight letters of the test word (see below). String1 is the same for every letter, and is specified by the address of the cell containing the test word (viz., H1). Num2 also is always the same, namely 1, the number of letters to be extracted from string1. Num1 specifies which letter to extract, position 0 for "p", position 1 for "a", etc. To extract p, use @MID(H1,0,1); to extract a, use @MID(H1,1,1); to extract

s, use @MID(H1,2,1); and so on.

Knowing the @MID function for each letter of the test cell's word,

	A	B	C	D	E	F	G	H	
1								pastille	<-- test word
2									
3	0	1	2	3	4	5	6	pastill	
4	0	1	2	3	4	6	7	pastile	
5	0	1	2	3	5	6	7	pastlle	
6	0	1	2	4	5	6	7	pasille	
7	0	1	3	4	5	6	7	patille	
8	0	2	3	4	5	6	7	pstille	
9	1	2	3	4	5	6	7	astille	
10	0	1	2	3	4	5		pastil	
11	0	1	2	3	4	6		pastil	
12	0	1	2	3	4	7		pastie	
.								.	
40	0	1	2	3	6			pastl	
41	0	1	2	3	7			paste	
42	0	1	2	4	5			pasil	
.								.	
102	0	1	3	7				pate	
103	0	1	4	5				pail	
104	0	1	4	6				pail	
105	0	1	4	7				paie	
106	0	1	5	6				pall	
107	0	1	5	7				pale	
.								.	
168	0	1	6					pal	
.								.	
195	1	4	6					ail	
196	1	4	7					aie	
197	1	5	6					all	
198	1	5	7					ale	
.								.	
247	6	7						le	

H4: @MID(H1,A4,1)&@MID(H1,B4,1)&@MID(H1,C4,1)&@MID(H1,D4,1)&@MID(H1,E4,1)&@MID(H1,F4,1)&@MID(H1,G4,1)

H6: @MID(H1,A6,1)&@MID(H1,B6,1)&@MID(H1,C6,1)&@MID(H1,D6,1)&@MID(H1,E6,1)&@MID(H1,F6,1)&@MID(H1,G6,1)

H8: @MID(H1,A8,1)&@MID(H1,B8,1)&@MID(H1,C8,1)&@MID(H1,D8,1)&@MID(H1,E8,1)&@MID(H1,F8,1)&@MID(H1,G8,1)

H10: @MID(H1,A10,1)&@MID(H1,B10,1)&@MID(H1,C10,1)&@MID(H1,D10,1)&@MID(H1,E10,1)&@MID(H1,F10,1)

H12: @MID(H1,A12,1)&@MID(H1,B12,1)&@MID(H1,C12,1)&@MID(H1,D12,1)&@MID(H1,E12,1)&@MID(H1,F12,1)

we can appropriately concatenate them to generate in each cell of Column H a formula for one of the possible interior letter sequences.

To facilitate that one-time-only formula entry process, first list the sequence of letter position-numbers (numls) needed for a given cell's formula, listing those numls in separate narrow-width cells in Columns A through G to the left of the formula cell in Column H (see spreadsheet on preceding page). Then, in composing and entering the required formula of concatenated @MID functions for that cell, the num1 values for each @MID function are easily available, and can be entered as cell addresses.

Working left to right, the first interior same-order letter sequence in "pastille" is "pastill", whose letter positions are 0123456. With those numls in cells A3 through G3, the required formula in H3 comprises the following seven concatenated @MID functions:

```
@MID(H1,A3,1)&@MID(H1,B3,1)&@MID(H1,C3,1)&@MID(H1,D3,1)&
 @MID(H1,E3,1)&@MID(H1,F3,1)&@MID(H1,G3,1)
```

Note that the coupled @MID functions are identical except for the num2 values specified as A3, B3, C3, ... A cell formula can exceed the display width of a cell; typically it can comprise 240 characters.

There are eight possible seven-letter interior same-order letter sequences for an eight-letter word: 0123456, 0123457, 0123467, 01235-67, 0124567, 0134567, 0234567, and 1234567. There are 28 possible six-letter interior same-order sequences. For five-, four-, three- and two-letter sequences, there are 56, 70, 56 and 28 possible combinations, respectively. The cell-formulas for each set require one fewer @MID function than the previous set requires. The last cell in column H, H247, displays "le" from "pastille", representing the num1 sequence 67, and requires only two @MID functions: @MID(H1, A247,1)&@MID(H1,B247,1), where A247 and B247 contain the numbers 6 and 7, respectively.

Sections of the template are shown on the preceding page. Missing rows, evident from the missing row numbers at the left margin, are represented by a blank row with a dot in the Column H cell. Examples of some of the Column H formulas are also shown. The formulas in the remainder of the cells of Column H are deducible from the pattern shown.

The template provides an interactive computerized method for searching for words-within-a-word. To test multiple words per session, create multiple parallel columns to the right of Column H, each with a test cell in Row 1, e.g., at cells I1, J1, K1,... Repeat the Column H formulas in those columns, modifying the formulas to change the stringl cell address from H1 to that of each column's appropriate test cell (I1, J1, K1,...). If desired, the results obtained after each session can be printed before the template is reused, and the printouts accumulated for later analysis and permanent record-keeping.

Limitations and Possibilities of Spreadsheet Logology

The spreadsheet applications described were designed to operate primarily on single words one at a time in contradistinction to all words in a list at once. Yet even that limited application can be useful, in particular to logologists who are not computer experts and therefore would otherwise not have computer assistance of any kind. While it is possible, for example, to visualize mentally all possible words generated by letter-insertion into a single five-letter word, the spreadsheet template for that activity eliminates the tedium, accelerates the process, and ensures against missing any of the 156 sequences possible with a 26-letter insertion-set.

Clearly, one expects more from a computer, however. How much of that "more" the spreadsheet environment can provide remains to be explored. The virtue of the spreadsheet environment is that it permits the exploration by the non-computer-expert--the average PC user or motivated novice with minimal computer literacy or typing skill.

Spreadsheet models can be designed to operate on all words in a moderate sized list (e.g., up to a few thousand). Most spreadsheets for the PC enable importation of so-called text-files consisting of a columnar word list, whereupon each word appears in a cell in one column of the spreadsheet. Indeed, many spreadsheets have built in features for sequentially operating on the entries of such a list; those features simplify constructing reusable templates for specific applications.

For example, one can easily construct a template to compute the sum of the numbers corresponding to the alphabetical position of the letters in each word in a list arbitrarily of 200 words, enough to examine the words in a Shakespeare sonnet, for example. Where "a"=1, "b"=2, "c"=3,..., the template computes the sum of the digitized letters of each word in the list (e.g., "bad"=2+1+4=7); call them Borgmann letter-sums. The results appear in a column adjacent to the word list column. In two additional adjacent columns, the template computes for each word the word's letter-count, and the ratio of the word's Borgmann letter-sum to its letter-count. If the digitized letters are viewed as "weights", "z" being the heaviest, "a" being the lightest, the ratio of a word's Borgmann letter-sum to its letter-count gives a kind of letter-density: in a sea of "the"s (33/3=11), "dizzy" (90/5=18) is a sinker, "bead" (12/4=3) is a floater.

For each parameter (Borgmann letter-sum, letter-count, letter-density), the template could also be designed to compute the average value of the parameter for all the words in the list, yielding a unique set of numeric specifications for the word list. One might use such a template, for example, to analyze the list of words comprising a Shakespeare sonnet, comparing the values with those of the author's other sonnets or with those of other authors' sonnets.

One can imagine any number of other uses for such a template

in analyzing word lists of textual origin. One could also include in the template additional sections that compute other numeric characteristics of the word list, such as frequency of use of each letter of the alphabet, the number of unique words, the number of words total, and the frequency of use of each word in the text. Those particular applications of spreadsheet analysis of word lists are cited only to illustrate that a spreadsheet capability exists for word list analysis, and to emphasize that the capability is available to the non-computer-expert.

Whether the spreadsheet environment can accommodate very long lists of words (tens of thousands), such as a dictionary list, remains to be explored. Realization of this capability may be difficult, owing to current spreadsheet size limitations.

The editor asked whether spreadsheets can have more than two dimensions, a feature that might facilitate exploring complex word networks. Several multidimentional spreadsheets are, in fact, available, but I have had no opportunity to test them (or reason to, until now). Both are purportedly user-friendly, i.e., accessible to the non-computer-expert, especially to someone already familiar with the more common two-dimensional spreadsheet.

Getting Started

Learning to use a PC spreadsheet program for logological studies requires no more computer literacy than that required to use a simple word processor and no more skill than that required to use a typewriter. With their string-manipulative functions, PC spreadsheets not only provide computer power to the logologist, but also invite new forms of wordplay specific to the spreadsheet environment.

An excellent way to learn how to use the electronic spreadsheet is by experimentation; in a few hours, a motivated novice will have surpassed the elementary skill required to set up the templates described above. There are many commercially available spreadsheet programs, some under $100, and several public-domain or so-called shareware programs available for minimal cost.