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 modeling), 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 so-called @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-
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:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>wordplay</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>word</td>
<td>play</td>
</tr>
<tr>
<td>3</td>
<td>layout</td>
<td>out</td>
</tr>
<tr>
<td>4</td>
<td>spreadsheet</td>
<td>spread</td>
</tr>
<tr>
<td>5</td>
<td>wordplay</td>
<td>beer</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
dependent

"&"

oper-
c formulas;

ith a + or

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.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>141</td>
<td>lien</td>
<td>alien</td>
<td>lien</td>
<td>lien</td>
<td>lien</td>
</tr>
<tr>
<td>142</td>
<td>a</td>
<td>alien</td>
<td>lien</td>
<td>lien</td>
<td>lien</td>
</tr>
<tr>
<td>143</td>
<td>b</td>
<td>bien</td>
<td>lien</td>
<td>lien</td>
<td>lienb</td>
</tr>
<tr>
<td>144</td>
<td>c</td>
<td>cien</td>
<td>cien</td>
<td>lien</td>
<td>lienc</td>
</tr>
<tr>
<td>145</td>
<td>d</td>
<td>dlien</td>
<td>dlien</td>
<td>dlien</td>
<td>liend</td>
</tr>
<tr>
<td>152</td>
<td>k</td>
<td>klien</td>
<td>klien</td>
<td>liken</td>
<td>lienk</td>
</tr>
<tr>
<td>153</td>
<td>l</td>
<td>lien</td>
<td>lien</td>
<td>liien</td>
<td>liien</td>
</tr>
<tr>
<td>154</td>
<td>m</td>
<td>miien</td>
<td>miien</td>
<td>miien</td>
<td>miien</td>
</tr>
<tr>
<td>155</td>
<td>n</td>
<td>niien</td>
<td>niien</td>
<td>niien</td>
<td>niien</td>
</tr>
<tr>
<td>160</td>
<td>s</td>
<td>silien</td>
<td>silien</td>
<td>silien</td>
<td>silien</td>
</tr>
<tr>
<td>163</td>
<td>v</td>
<td>viien</td>
<td>viien</td>
<td>vien</td>
<td>vien</td>
</tr>
</tbody>
</table>

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 for­
mula 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 addi­
tional 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 per­
manent record.

The ladder-building template also facilitates searching for hos­
pitable words—words each of whose letters accepts a replacement
that generates a legitimate word. Hospitable words are readily spot­
ted, 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 (in­
sertion 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 log­
istical 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-sequency 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("logical",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).

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>play</td>
<td></td>
<td>=@MID(&quot;wordplay&quot;,4,4)</td>
</tr>
<tr>
<td>2</td>
<td>word</td>
<td></td>
<td>=@MID(&quot;wordplay&quot;,0,4)</td>
</tr>
<tr>
<td>3</td>
<td>wordplay</td>
<td>play</td>
<td>=@MID(C3,4,4)</td>
</tr>
<tr>
<td>4</td>
<td>play</td>
<td></td>
<td>=@MID(C3,0,4)</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>4</td>
<td>=@MID(C3,4,5)</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>4</td>
<td>=@MID(C3,0,5)</td>
</tr>
<tr>
<td>7</td>
<td>pa</td>
<td></td>
<td>=@MID(C3,4,1)&amp;@MID(C3,6,1)</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>1</td>
<td>=@MID(C3,4,5)&amp;@MID(C3,6,2)</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>1</td>
<td>=@MID(C3,4,5)&amp;@MID(C3,6,2)&amp;@MID(C3,6,3)</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>1</td>
<td>=@MID(C3,4,5)&amp;@MID(C3,6,2)&amp;@MID(C3,6,3)</td>
</tr>
</tbody>
</table>

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

```
A    B    C    D    E    F    G    H
---  ---  ---  ---  ---  ---  ---  ---
 1   pastille
 2   pastille
 3   pastille
 4   pastille
 5   pastille
 6   pastille
 7   pastille
 8   pastille
 9   pastille
10  pastille
11  pastille
12  pastille
13  pastille
14  pastille
15  pastille
16  pastille
17  pastille
18  pastille
19  pastille
20  pastille
21  pastille
22  pastille
23  pastille
24  pastille
25  pastille
26  pastille
27  pastille
28  pastille
29  pastille
30  pastille
31  pastille
32  pastille
33  pastille
34  pastille
35  pastille
36  pastille
37  pastille
38  pastille
39  pastille
40  pastille
41  pastille
42  pastille
43  pastille
44  pastille
45  pastille
46  pastille
47  pastille
48  pastille
49  pastille
50  pastille
51  pastille
52  pastille
53  pastille
54  pastille
55  pastille
56  pastille
57  pastille
58  pastille
59  pastille
60  pastille
61  pastille
62  pastille
63  pastille
64  pastille
65  pastille
66  pastille
67  pastille
68  pastille
69  pastille
70  pastille
71  pastille
72  pastille
73  pastille
74  pastille
75  pastille
76  pastille
77  pastille
78  pastille
79  pastille
80  pastille
81  pastille
82  pastille
83  pastille
84  pastille
85  pastille
86  pastille
87  pastille
88  pastille
89  pastille
90  pastille
91  pastille
92  pastille
93  pastille
94  pastille
95  pastille
96  pastille
97  pastille
98  pastille
99  pastille
100 pastille
101 pastille
102 pastille
103 pastille
104 pastille
105 pastille
106 pastille
107 pastille
108 pastille
109 pastille
110 pastille
111 pastille
112 pastille
113 pastille
114 pastille
115 pastille
116 pastille
117 pastille
118 pastille
119 pastille
120 pastille
121 pastille
122 pastille
123 pastille
124 pastille
125 pastille
126 pastille
127 pastille
128 pastille
129 pastille
130 pastille
131 pastille
132 pastille
133 pastille
134 pastille
135 pastille
136 pastille
137 pastille
138 pastille
139 pastille
140 pastille
141 pastille
142 pastille
143 pastille
144 pastille
145 pastille
146 pastille
147 pastille
148 pastille
149 pastille
150 pastille
151 pastille
152 pastille
153 pastille
154 pastille
155 pastille
156 pastille
157 pastille
158 pastille
159 pastille
160 pastille
161 pastille
162 pastille
163 pastille
164 pastille
165 pastille
166 pastille
167 pastille
168 pastille
169 pastille
170 pastille
171 pastille
172 pastille
173 pastille
174 pastille
175 pastille
176 pastille
177 pastille
178 pastille
179 pastille
180 pastille
181 pastille
182 pastille
183 pastille
184 pastille
185 pastille
186 pastille
187 pastille
188 pastille
189 pastille
190 pastille
191 pastille
192 pastille
193 pastille
194 pastille
195 pastille
196 pastille
197 pastille
198 pastille
199 pastille
200 pastille
201 pastille
202 pastille
203 pastille
204 pastille
205 pastille
206 pastille
207 pastille
208 pastille
209 pastille
210 pastille
211 pastille
212 pastille
213 pastille
214 pastille
215 pastille
216 pastille
217 pastille
218 pastille
219 pastille
220 pastille
221 pastille
222 pastille
223 pastille
224 pastille
225 pastille
226 pastille
227 pastille
228 pastille
229 pastille
230 pastille
231 pastille
232 pastille
233 pastille
234 pastille
235 pastille
236 pastille
237 pastille
238 pastille
239 pastille
240 pastille
241 pastille
242 pastille
243 pastille
244 pastille
245 pastille
246 pastille
247 pastille
```

we can apply the formulas of Column H.

To facilitate the sequence of Column H (see spr; entering the formula in that cell, the table, and cell J cell. Entering the formula in "pastille" comprises the

@MID(H1,A2,1) &@MID(H1,B2,1) &@MID(H1,C2,1) &@MID(H1,D2,1) &@MID(H1,E2,1) &@MID(H1,F2,1) &@MID(H1,G2,1)

Note that the num2 values comprise the display value.

There are 26 sequences for 67, 0124567, six-letter i, and two-letter combinations, and fewer @MID values in column H, num1 sequences (A247,1) &@MID(H1,G6,1) and 7, respectively.

Sections of columns, or entire rows, may be reloaded, and the Column H cell. Examples of Column H cell. Examples of Column H cell.

The temporary searching function, create each with a proper sequence to change the appropriate temporary record, and permanent record.
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 numl 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, 0123567, 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 numl 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 the characteristics of the letters of a word or words total. Those particular words are cited only for word lists available to the non-computer-expert.

Whether the lists of words remain to be explored. The editor of the word lists remains to be explored, however, in the spreadsheet environment.

The editor and reader of the word lists remains to be explored, however, in the spreadsheet environment.

Getting Started

Learning spreadsheets requires no typewriter skills, a simple word list, or so-called text-file format. Spreadsheets not only invite new ways of looking at words, but also the new ways of looking at words that spreadsheets permit.

An excellent program for spreadsheets is by exception for the PC. Learning how to use the templates described is by exception for the PC. Learning how to use the templates described is by exception for the PC. Learning how to use the templates described is by exception for the PC. Learning how to use the templates described is by exception for the PC. Learning how to use the templates described is by exception for the PC. Learning how to use the templates described is by exception for the PC.
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 multidimensional 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.