The PowerChecker: A Visual Basic program for ensuring data integrity

JAMES C. BEATY, JR.

Midwest Research Institute, Kansas City, Missouri

Data entry and encoding errors can jeopardize the integrity of data sets generated in a variety of research settings. Despite researchers' pursuits of more accurate entry methods, data entry errors persist. Although techniques exist for identifying such errors, the PowerChecker program described here provides a more efficient method of data set validation. Rather than enter data twice and then manually search for the correct values when there are discrepancies between the two sets, with PowerChecker the user can correct entry errors as the data are entered the second time. In addition, the time-stamped record of changes to the original data set aids in meeting quality assurance requirements of the Good Clinical Practice standards.

When written data are transferred to electronic format via typing, there is a potential for errors in data entry. Various strategies have been developed for dealing with such errors. The most obvious and laborious technique is to double check the data by visually comparing each written value with the corresponding value in the electronic file, and then make any necessary correction. It takes time to do this, however, even when a data set is modest in size. Some statistical software packages (e.g., SAS) include functions that can reduce the time spent on double checking, but a significant investment in time is still required. Other attempts to ensure data integrity have ranged from techniques designed to improve the accuracy of keyboard entry (Galinsky, Schleifer, & Pan, 1995) to alternate methods such as verbal data entry (Frankish & Noyes, 1990). This line of research has yielded interesting techniques, but none of them substantially reduce data entry errors.

The PowerChecker was developed in response to a need for a more efficient method of identifying and correcting data entry errors. The program works in conjunction with Microsoft Access and simultaneously checks the values entered into the PowerChecker with the corresponding values in a database. If an error is detected, the program makes the appropriate change in the database at

that moment and keeps a time-stamped record of the change that has been made.

Reducing Data Entry Errors

Statistical software package functions for reducing errors. Data are usually imported into statistical software packages (e.g., SPSS, SAS, BMDP) from ASCII, spreadsheet, database, or word-processing files. After data are collected on paper-and-pencil forms, it is common to enter the data into electronic data files via a computer keyboard. With each keystroke, however, there is an opportunity for error. Such data entry and encoding errors have been noted as a source of error variance in scientific research (Hunter & Schmidt, 1990). Therefore, overlooking this type of error unnecessarily inflates error terms and can cause effects to be missed.

Some statistical software packages include program functions that aid in the reduction of data entry errors. For example, the manual for SAS, Version 6.12 (1990), describes the "COMPARE" procedure. When used to compare two data sets, this procedure generates a report of the differences between two data sets. The logic behind such a procedure is that if the same data are entered into the computer twice, the files should be exactly the same. If the values of corresponding data points are different, one of the two values represents an entry error.

Although not as stringent a validity check as the COMPARE procedure, BMDP, Version 7.0, includes the "CHECK" command (BMDP, 1993), which flags values that are outside a range limit set by the user. For example, if an incorrect value of 55 is entered when the correct value is 5 and the range is limited from 1 to 10, the program flags the incorrect value. SPSS (1996) likewise allows the user to restrict the range of data values, although the setting is designed to be used when data are keyed directly into the SPSS Data Editor. The fact that most statistical software packages address the problem of data entry errors attests to the serious concern about data integrity. However, range limits for data values, along with procedures such as COMPARE, do not adequately and efficiently address this problem. Although range limits can prevent some entry errors, they do not preclude others (e.g., entering a 2 instead of a 1 in the example above). In addition, procedures such as COMPARE can consume time as the user sifts through the original data to confirm the correct values for flagged differences between data sets.

Alternatives to keyboard entry. A strategy for reducing the time involved in double checking data as well as the reliance on range limits is to decrease the number of errors that need to be identified. Researchers have investigated techniques for improving the accuracy of keyboard entry (Galinsky et al., 1995; Sears, Revis, Swatski, & Crittenden, 1993). Although error rates have been reduced in some instances, many researchers have taken different routes in search of improved accuracy and have

This work was supported in part by DOE Contract 85X-SN602C, EPRI Contract WO8021-12, NIEHS Grant ES07053, and NINR Grant 01675. The author acknowledges Michael Gibertini's commitment to quality scientific research, which was the inspiration for the development of the PowerChecker. In addition, the author thanks Darrell Butler and an anonymous reviewer for their helpful comments on this article. Correspondence concerning this article should be addressed to J. C. Beaty, Jr., who is now at the Department of Psychology, Colorado State University, Fort Collins, CO 80523 (e-mail: jbeatyjr@lamar.colostate.edu).

The screenshot shows a Windows-style application window titled "The Powercheck Data Integrity System 2.0". The interface is designed for data verification. At the top, there are two text input fields: "Database Name:" containing "C:\Db1.mdb" and "Table Name:" containing "Survey 1". Below these, the interface is divided into sections. On the left, there is a "Subject" field with the value "1" and a "DOUBLE CHECKED" button. To the right of the subject field, there are two rows of input fields. The first row has "Response1" with the value "5" and an "Accept" button. The second row has "Double Check Value" with the value "55". At the bottom of the main form area, there are five buttons: "First Record", "Previous Record", "Next Record", "Last Record", and "Next Unchecked Record". An "Exit" button is located at the bottom right of the window. The Windows taskbar at the bottom shows the Start button, several icons, and the taskbar itself with the active window "The Powercheck Dat..." and the system clock showing "3:54 PM".

Figure 1. Screen display for double checking data.

developed alternate methods of transferring data from paper-and-pencil forms to electronic data files. For example, data collected on Scantron forms can be transferred to an ASCII file by passing the forms through Scantron optical scanning hardware, which reads the darkened answer spaces (Barton et al., 1991). Another recent development is verbal data entry (Frankish & Noyes, 1990; Washburn & Putney, 1990). Speech sensitive software allows one to read data into the computer with a microphone.

Although these two techniques are appealing, they have drawbacks. Scantron hardware and software are expensive, making this method too costly for many research projects. And speech-mediated data entry is often no better, and sometimes worse, in speed and accuracy than keyboarding (Damper, Tranchant, & Lewis, 1996). Most individuals can key in numbers faster than they can talk; and errors are often caused by speech recognition software. Verbal data entry remains promising, however, and is a focus of current research (Baber & Hone, 1993; Frankish, Jones, & Hapeshi, 1992; Murray, Jones, & Frankish, 1996; Noyes & Frankish, 1994).

Description of the PowerChecker

Since entry errors in data sets persist, the PowerChecker has been designed to provide a more efficient method of double checking the accuracy of data that have first been entered into a database program. The PowerChecker is a

Visual Basic (Version 4.0), Windows 95 compatible program designed to work in conjunction with Microsoft Access 7 (with minor code revisions and inclusion of different data access engines, it can be configured to work with other database programs as well). The user begins using the PowerChecker by selecting a database to access from the program display of available Microsoft Access database objects. After a database object has been chosen, the user is prompted to select a table containing data to be double checked. When a table is selected, the program accesses it as a dynaset object.

Once a table has been chosen, the program displays the screen for double checking data previously entered into a table (Figure 1). Each line of data in a table is referred to as a record. The program presents a subject identification number for the current record, along with the variable (field) name being double checked. The user then enters values for each variable. To note missing data, the user presses the F12 key. The words Missing Data are printed on the screen as the PowerChecker verifies that the corresponding value in the database is null.

Any value entered into the PowerChecker is simultaneously compared with the value first entered into the database. If the values match, the PowerChecker moves on to the next field after a press of the enter key. However, if the values do not match, the PowerChecker displays the first entered value along with the value currently being

Table 1
PowerChecker and Conventional Double-Checking Method
Group Means (in Minutes), Standard Deviations,
Significance Test, and Effect Size

Access Query		PowerChecker		<i>t</i> (7)	<i>d</i>
<i>M</i>	<i>SD</i>	<i>M</i>	<i>SD</i>		
38.4	4.0	23.5	3.9	15.92	.97

entered into PowerChecker (Figure 1). The user must either agree with the first entry value or click the Accept button to change the value in the database to the value in the double-check box. If the double-check value is accepted, a dialog box prompts the user for the user's name and an identification number. In addition, the user is prompted for the location of a log file in which the user's name and identification number are automatically entered, along with the field name, the original and double-check values, and the date and time the correction is made. If the user enters a name and an identification number, the value in the dynaset object is changed. If the user chooses not to enter a name and identification number, the change is not made to the dynaset. This time-stamped record of changes is consistent with such quality assurance standards as the Food and Drug Administration's (1997) Good Clinical Practices and ISO 9000, which are rapidly becoming requirements in grants and contracts issued by federal agencies.

Once all variables in a record have been double checked, the program moves on to the next record. Although the program automatically moves on to the next record when the previous one has been double checked, the user can also navigate through the records by using a series of buttons: First Record, Previous Record, Next Record, Last Record, and Next Unchecked Record (see Figure 1). The Next Unchecked Record button moves to the next record in the table that has not been double checked. This feature is useful when there is a large number of records in a table and a great many of them have already been double checked. The user can move directly to the records that need to be double checked. All of these buttons make use of a series of "move" methods (e.g., MoveNext) in Visual Basic that are applicable to dynasets.

A record is not flagged as double checked until all of the variables for a record have been double checked. The first field in every table is vital for tracking whether or not a record has been double checked. That is, the first field in any table formatted to work with the PowerChecker is designated a "double-check" field. This field contains a "0" as its default value, indicating that the current record has not been double checked. When a record has been double checked, the PowerChecker enters a "1" in the first field. Although the double-check field in the table is not made visible by the PowerChecker, the user is made aware of which records have been double checked via a "double-check" icon displayed on the screen (Figure 1). The icon is light gray if the record has not been double checked and bright red if it has been double checked. In this way, the

user knows immediately which records have been double checked.

Efficiency Analysis of the PowerChecker

The PowerChecker saves time during the double checking process because it has the ability to make "real-time" checks on the accuracy of data. Rather than going back through stacks of data to find the correct value for an entry error, the user can make corrections as he/she enters the data a second time. A quantitative analysis of the PowerChecker's time saving features has revealed that double checking of data with the PowerChecker can result in as much as a 39% gain in efficiency over conventional double-checking methods. In a research institution where the PowerChecker is currently used, eight individuals with extensive data entry experience double checked an average of 1,238 data points using both a conventional double-checking procedure similar to those described earlier (a Microsoft Access Query) and the PowerChecker. The order of double-checking procedures was counter-balanced. Participants in this study committed an average of 16 data entry errors. The Access Query alerted participants to the presence of data entry errors, which they then tracked down and corrected. Errors detected by the PowerChecker were corrected as the data were entered a second time. The results of this study, summarized in Table 1, reveal a significant difference between the average time it took participants to double check the data with the conventional method (38.4 min)¹ and with the PowerChecker (23.5 min). The results of this study suggest that the PowerChecker, in comparison with conventional double-checking methods, does indeed represent a more efficient method of double checking data for entry errors.

Conclusions

The PowerChecker represents two important advancements in ensuring data integrity. First, not only does the PowerChecker include most features of currently available double-checking methods, it is also more efficient, because users are able to do real-time double checking. Second, the PowerChecker maintains a log file, which contains a record of changes made to the database. This is helpful for tracking the number and type of data entry errors. More importantly, this feature is a requirement of Good Clinical Practices quality assurance standards and represents an effort to comply with professional standards of scientific research.

Availability

To obtain an executable version of the PowerChecker and the source code, send two 3.5-in. IBM-compatible disks and a self-addressed, postage paid disk mailer to Section Head, Biobehavioral Sciences Section, Midwest Research Institute, 425 Volker Blvd., Kansas City, MO 64110. The author welcomes correspondence concerning the PowerChecker. However, the PowerChecker is distributed as developmental software and is not accompanied

by technical support from Midwest Research Institute. A readme.txt file included with the executable version provides detailed instructions for operating the PowerChecker and formatting Microsoft Access tables for use with the program. Although the PowerChecker is a helpful tool for ensuring data integrity, it should be viewed as a starting point for the development of data integrity software that minimizes the time investment necessary for adhering to professional research standards.

REFERENCES

- BABER, C., & HONE, K. S. (1993). Modeling error recovery and repair in automatic speech recognition. *International Journal of Man-Machine Studies*, **39**, 495-515.
- BARTON, C., HATCHER, C., SCHURIG, K., MARCIANO, P., WILCOX, K., & BROOKS, L. (1991). Managing data entry of a large-scale interview project with optical scanning hardware and software. *Behavior Research Methods, Instruments, & Computers*, **23**, 214-218.
- BMDP (1993). Los Angeles, CA: BMDP Statistical Software, Inc.
- DAMPER, R. I., TRANCHANT, M. A., & LEWIS, S. M. (1996). Speech versus keying in command and control: Effect of concurrent tasking. *International Journal of Human-Computer Studies*, **45**, 337-348.
- FOOD AND DRUG ADMINISTRATION (1997). *Good clinical practice: Consolidated guideline* [On-line]. Available: <http://www.fda.gov/cder>
- FRANKISH, C. R., & NOYES, J. (1990). Sources of human error in data entry tasks using speech input. *Human Factors*, **32**, 697-716.
- FRANKISH, C. [R.], JONES, D. [M.], & HAPESHI, K. (1992). Decline in accuracy of automatic speech recognition as a function of time on task: Fatigue or voice drift? *International Journal of Man-Machine Studies*, **36**, 797-816.
- GALINSKY, T. L., SCHLEIFER, L. M., & PAN, C. S. (1995). The influence of performance standards feedback on speed and accuracy in an electronically monitored data-entry task. *International Journal of Human-Computer Interaction*, **7**, 25-36.
- HUNTER, J. E., & SCHMIDT, F. L. (1990). *Methods of meta-analysis: Correcting error and bias in research findings*. Newbury Park, NJ: Sage.
- MURRAY, A. C., JONES, D. M., & FRANKISH, C. R. (1996). Dialogue design in speech-mediated data-entry: The role of syntactic constraints and feedback. *International Journal of Human-Computer Studies*, **45**, 263-286.
- NOYES, J. M., & FRANKISH, C. R. (1994). Errors and error correction in automatic speech recognition systems. *Ergonomics*, **37**, 1943-1957.
- SAS (1990). Cary, NC: SAS Institute, Inc.
- SEARS, A., REVIS, D., SWATSKI, J., & CRITTENDEN, R. (1993). Investigated touchscreen typing: The effect of keyboard size on typing speed. *Behaviour & Information Technology*, **12**, 17-22.
- SPSS (1996). Chicago, IL: SPSS Inc.
- WASHBURN, D. A., & PUTNEY, R. T. (1990). Voice reaction times with recognition for Commodore computers. *Behavior Research Methods, Instruments, & Computers*, **22**, 253-255.

NOTE

1. This is the average time required for entering the data a second time and tracking down the correct values for entry errors highlighted by the Access Query. The time required to set up the Access Query is not included in this average. Set-up time is not included because it varies among conventional double-checking procedures (e.g., the COMPARE procedure in SAS), some of which take only a couple of minutes to set up. Hence, in cases in which setting up a comparison procedure is time consuming, the time savings estimate in this study will likely be an underestimate.

(Manuscript received October 5, 1998;
revision accepted for publication February 3, 1999.)