Journal of Statistical Software

June 2014, Volume 58, Software Review 1. *http://www.jstatsoft.org/*

Reviewers: A. Jonathan R. Godfrey
Massey University
and
M. Theodor Loots
University of Pretoria

Statistical Software (R, SAS, SPSS, and Minitab) for Blind Students and Practitioners

R Foundation, Vienna, Austria. `http://www.R-project.org/`
SAS Institute Inc., Cary, United States of America. `http://www.sas.com/`
SPSS Inc., Chicago, United States of America. `http://www.spss.com/`
Minitab Inc., State College, United States of America. `http://www.minitab.com/`

Introduction

Access to information is crucial for the blind person's success in education, but transferring knowledge about the existence of techniques into actually being able to complete those tasks is what will ultimately improve the blind person's employment prospects. This paper is based on the experiences of the two authors; as blind academics in statistics, we are dependent on the usefulness of statistical software for blind users more than most blind people. The use of the "we" throughout this article is intentionally meant to be personal in terms of our own experiences but more importantly, also reflects the needs of the blind community as a whole.

Blind students often benefit from one-to-one teaching resources which can aid in their uptake of statistical thinking and practice, but this additional service is only a temporary solution. Once the student has completed their first course in statistics, they may embark on research at a university, or head out into industry to apply their knowledge. Irrespective of the direction they choose, they will need certainty in being able to independently create graphs for the sighted readers of their work.

At the 2009 Workshop on E-Inclusion in Mathematics and Sciences, the first author was able to meet other researchers who are concerned about the low rate of blind people entering the sciences in a broad sense and the mathematical sciences in particular. Godfrey (2009) presents what we believe is the first formalized presentation (written by a blind person) of the current state of affairs for blind people taking statistics courses. Much of the material covered in that work still holds true today, although there have been some technological changes that have altered the landscape a little. The four main considerations of Godfrey (2009) were graphics, software, statistical tables, and mathematical formulae. Although software was just one element discussed, graphics and mathematical formulae are playing an increasing role in the usefulness of statistical software, especially with respect to the accessibility of support documentation.

We have reviewed four statistical software packages that blind people might want to use in their university education. Our review is restricted to the Windows operating system because this is the predominant environment in which blind people are working. Before we review R, SAS, SPSS, and Minitab, we outline our expectations of statistical software, describe a simple task used to evaluate some practical experiences, and describe some issues with certain file formats and graphics. Following the software-specific sections there is a general discussion of pertinent issues for software developers, including the relevant details of the legislative environment in the United States of America. The article closes with a simplified set of criteria and our overall assessment of the current state of the usefulness of statistical software for blind users.

What do we expect of statistical software?

Everyone who regularly performs statistical analyses has a preferred software option. It may of course be a combination of tools but nonetheless, we all have preferences. For the sighted user the preference may be based on the types of activity required of the software, and once a set of options that have the right mix of features are found, preference may well then come down to the user's idea of usefulness. The authors would like to emphasize that the exercise of determining a software preference is very different for blind users. We first ask if the software is accessible, by which we mean that we can make sufficient use of the software to do the job required. Only then do we get to ask about the comparative merits of the accessible options. At the time of writing, the first and second authors are regular users of R and SAS respectively. Perhaps this contributes to our findings about other software, but we feel that the common opinion reached about gauging the level of accessibility of any statistical software transcends any personal biases we may hold.

Highly interactive data exploration is promoted by some software packages. This sort of terminology often points towards a style of interaction with the software that is beyond the blind user to manage successfully, but it also reflects a style of working that is not appropriate for a blind user because it is dominated by being able to interpret content visually.

For the purposes of this article, *accessible* will be defined as:

> The software can be used to complete the same tasks as a sighted user through use of the adaptive technology blind people use in everyday life.

Adaptive technology includes software commonly known as screen readers, refreshable braille displays, and braille embossers. We note that the technological landscape is changing with the improving accessibility of smart phones and tablets, and that in time our definition may need to be updated.

Screen readers attempt to read the text on screen as well as interpret some graphical features. They do not interpret pictures or graphs, although some can try to read the text embedded within graphics. Moreover, they can have difficulty reading uncommon symbols such as Greek, and as a consequence, access to mathematical symbols is a challenge that most screen reading software is not capable of meeting.

Screen readers are operated in two ways. The passive operation of screen readers relies on the actions of the user in the application being used. This includes typing and moving the cursor in the same way as would a sighted user. It should be noted that most screen reader users do not use the mouse but replace mouse actions with keystrokes. Audible feedback is given in real time through synthetic speech tailored to the user's preferences, which for example might be that every key press is announced versus only reading whole words. The active use of the screen reader occurs when the user needs to temporarily suspend work in the application in order to review content displayed on screen. Active use of the screen reader software in this manner is hugely inefficient as it means time and energy is spent on meeting the limitations of blindness when we would much rather be doing the same tasks as our sighted peers.

Refreshable braille displays deliver text in a tactile form, but rely on screen reading software to send the right text to the hardware. The failings of the screen reading software available to blind users limits the practical use of braille for accessing software. Refreshable braille displays are also very expensive and beyond the reach of many blind people especially in the developing world. Those blind people that do have a braille display often prefer to use braille over speech for reading and editing documents; this is especially true when handling case sensitive content, or applications that do not commonly incorporate spell checking functionality. At the time of writing, braille displays were restricted to linear output, whether that be in one or two lines of braille. Two-dimensional tablet-like displays are not yet a reality. If and when these are developed they will most certainly open up a whole new world of tactile representations for blind people.

In our view, the ability of statistical software to work with screen reading software is the primary measure of its accessibility. This, however, is not just a simple matter of yes or no. Some aspects of a software solution might be inaccessible but be of little importance to the user. Some software might have totally accessible input, but totally inaccessible output; for example, a LaTeX document is prepared in plain text which is accessible, but the mathematical content in the print-ready files is not readable by screen reading software so the outcome is inaccessible.

Minimum criteria for accessibility

The main focus for creating minimum standards for accessibility are based on the question, "Can we…?" even if that means we need to install additional software or find alternative work practices to complete the same tasks done by our sighted peers. We offer the following as a list of criteria by which the accessibility of statistical software will be judged in this article:

- Can we install the software without the assistance of a sighted colleague?
- Can we import data?
- Can we review the raw data?
- Can we edit the data?
- Can we create simple exploratory graphics and numeric summaries, i.e., perform simple exploratory data analysis (EDA)?

- Can we export the output from any summaries into a file for inclusion in a report? This avoids the difficulty of copying and pasting material.
- Can we know anything about the actual visual presentation of a graph?
- Can we update the graph if minor alterations in presentation are required?
- Can we read the help documentation that is part of the standard installation of the software?
- Can we access all features available to the sighted user? That is, are there any features that require a mouse to complete?

In Table 1 we present a quick reference guide with the answers to these questions for the four software applications we consider. More detailed findings are given in the software-specific sections later in this article.

Desirable features

The blind user is not different to the sighted user in terms of having a wish list for the user-friendliness of software, although perhaps we do have a few extra or different questions to ask of it. The short assessments of the following desirable criteria are also given in Table 1.

- Can the software offer the blind user a level of comfort about the presentability of graphs they create?
- How easily can we find out what capacity the software has to offer users? That is, can we find out any information about the software and blind or vision-impaired users?
- Can we efficiently deal with minor errors that we might not spot at first due to a lack of sight?
- How easily can we transfer knowledge from other users? That is, do we work in the same way as our sighted peers?
- If we cannot use an application in the same way as our sighted peers, is there another way that can provide us with partial access?
- Can the software save graphics in a more accessible format? This must include the formats needed for LaTeX documents but may also include scalable vector graphics (SVG) for insertion in fully-accessible document formats.
- Can the output be saved in a range of formats in order to maximize accessibility?

A practical experiment

In order to test our ability to use the software applications, the authors attempted the following exercise by themselves. The experiences are discussed in the subsequent sections. A sighted research assistant oversaw the first author's interactions with the software less familiar to him (SPSS and Minitab) so that any inadequacies could be documented.

Minimum criterion	R	SAS	SPSS	Minitab
Independent installation	yes	no	not easily	yes
Import data	yes	yes	yes, but wizard is challenging	yes
Review data	yes	yes	yes	no
Edit data	yes	yes	yes	no
Simple EDA	yes	yes	yes	yes
Exporting output	yes	yes	yes	yes
Update graph	start from scratch	start from scratch	start from scratch	start from scratch
Help documentation	yes	yes	limited	limited
Access to all features	yes	no	not determined conclusively	no
Desirable criterion	**R**	**SAS**	**SPSS**	**Minitab**
Graph presentation	yes	no	no	no
Information for blind users	third party	yes	yes	none found
Minor errors	yes	yes	yes, if command language is used	yes, if command language is used
Knowledge transfer	yes	yes	not guaranteed	not guaranteed
Flexibility	yes	yes	yes, but seldom used	no
Graphic formats	yes	not SVG	not SVG	cannot directly save postscript files, no SVG
Output formats	yes, but add-on packages required	yes, including both HTML and $\LaTeX$	yes, including HTML, but not $\LaTeX$	rich text or HTML, but not $\LaTeX$

Table 1: Minimum criteria and desirable features for assessing the accessibility of statistical software.

A dataset consisting of 100 observations of two variables was constructed in Microsoft Excel and converted to comma separated values. Both authors then used these files to:

1. Import the data from the more convenient file format.
2. Save the worksheet in the software's format.
3. Create some basic numeric summary statistics of the two variables. These needed to be possible to be read by the blind user as well as to be included in a report if desired.
4. Create a scatter plot of the two variables, and add a best fit straight line to this plot.
5. Extract the details of the straight line fitted, including its coefficients and fit statistics.
6. Use the model created to find predictions for a small set of values of the predictor.
7. Assess ability to share our findings with others.

We need to note the specifications of our software and hardware. All too often we have observed (and later document) the importance of these specifications when using findings from experiments like this one; use of the same screen reader is a crucial point to note in particular. Unless otherwise stated, the first author used a Windows 7 32-bit desktop running version 14 of **JAWS** while the second author used a Windows 7 64-bit laptop, also running version 14 of **JAWS**. **JAWS** (Job Applications With Speech; Freedom Scientific 2012) is a commercial product marketed by Freedom Scientific; it is a dominant player in the screen reader software industry, but there are many alternatives for the Windows operating system. **NVDA** (Non-Visual Desktop Access; **NVDA** Team 2014) is a free screen reader that is under active development by an open source community. Users of other operating systems have screen readers built into their operating systems that can deliver access, notably **VoiceOver** (Apple Inc. 2011) for the Macintosh and **ORCA** (**ORCA** Team 2006) for Linux.

File formats and accessibility

In order to prevent duplication in the sections that follow, we first discuss the issues that affect all software used by blind people. Use of inferior tools will limit the accessibility and usefulness of any software application; most statistical software interacts with various tools for importing and presenting information, especially for saving numerical or graphical output.

Accessibility issues exist for certain file formats and are well-documented elsewhere. Some file types such as plain text are readily accessible, while others rely heavily on the screen reader's ability to extract meaningful results by rendering the content.

Of course, some file types are not specifically a single file, but incorporate other files in them. For example, an HTML document is a document that links to various graphics or audio-visual material to make it complete. The accessibility of an HTML document therefore depends on the accessibility of the subordinate content. Accessibility of audio-visual material is not relevant for this article but how graphics are dealt with can affect the usefulness of support documentation that includes mathematical content.

Screen reader software does deal quite well with HTML content in general, but there are some limitations worthy of mention. Two key elements that improve the accessibility are the use of proper (plain) HTML tags for styles, and the use of alternate text for graphics. Web designers often create style tags that are specific to their documents instead of updating the tags that exist in HTML already; the most common example we find is the inappropriate use of heading style creation instead of the `<h1>`, `<h2>`, etc. commands. Use of these header tags enables the screen reader user to skip over material in less-interesting sections of a document using keystroke presses to find the next heading of each level. An HTML document that is long and does not have any standard headings needs to be read from top to bottom unless the reader is willing to take a punt on getting past the undesired material with other cursor movement keystrokes.

The use of `alt` attributes in conjunction with graphics offers the blind user a chance to find out what was intended for display in a graphic. For example, the mathematical symbols presented in many Wikipedia pages use the LaTeX code that created the content; while the sighted user sees $\alpha + \beta x + \epsilon$, the blind user will hear "backslash alpha plus backslash beta x plus backslash epsilon" which is quite intelligible in most situations.

Perhaps the most dreaded file format is the portable document format, more commonly known as PDF. Some PDF documents are readable, but much content presented in this format is not. The best screen readers can read the text content in PDFs but only a very small number of Greek symbols are handled well. Some axis labels and the like are read by screen readers which are ultimately little more than a distraction from the more readable sentences surrounding the graphics. In most situations the blind user can do little with a PDF beyond getting the gist of the material. Current attempts for optical character recognition of these documents is a work in progress, but currently the results do not leave the authors with certainty about the mathematical content in PDF documents. The worst PDFs are the scanned pages of journal articles and the like, as these are treated as if the whole page is a single graphic. The interested reader could see how much of a PDF is readable by a blind person by passing a few PDFs through a pdf-to-text conversion tool of their choosing.

Graphics

It is the view of the authors that blind users need to be able to create and interpret statistical graphs. At present, there is little opportunity for blind users to interpret graphs as part of an EDA. We can however create graphs for the sighted audiences we must interact with.

The reality of graph creation is that we will need to re-use code (after minor modification) instead of the graph-editing facilities that the software may have. Some software encourages a sighted user to point and click to achieve the modifications required; in such circumstances we will need to work differently. Blind users will always need to re-create the graphics from scratch.

On the whole, graphics are not accessible to the blind user, no matter which file format is chosen. There is one file format that stands out as having the potential to deliver added information to the blind user, but even scalable vector graphics (SVG) files rely on human intervention to manually add the necessary detail that make the graphic informative.

Both authors have experimented with embossed graphs in the past. Some braille embossers can produce graphics in tactile form, but the resulting images are of quite a low resolution. (They are certainly not as low-resolution as old-fashioned ASCII character representations.) These tactile images can only have practical relevance if they can be produced by the blind user on demand. Loots and van Staden (2007) for instance illustrated the importance of having access to graphics generated by statistical software, in applying time series techniques. We know that statistical software can be used with an embosser to generate tactile images on demand, but we currently cannot guarantee that every blind user will ever possess an embosser of their own. It may be more realistic to think that temporary access to an embosser could be arranged for the duration of a blind student's attendance in a statistics course. The problem is slightly circular however; until such time as there are more embossers in use, blind students will not get the most out of this technology, and the value of them will not become apparent to the next generation of blind students. Many universities do have access to embossers; we just need to ask that blind students and their teachers use the technology.

If we do not have access to an embosser, the blind practitioner must use alternative methods of working. Consider for example, the need to perform a residual analysis after fitting a linear model which is a simple exercise if the user can interact with a number of graphs. A blind user could complement the normal plot of residuals generated for their sighted audience, by performing a normality test that generates output in plain text.

R

No one can question the rise and rise of R (R Core Team 2013) since its initial development. Like many academics, the first author has been using R as his primary statistical software of choice since moving away from S-PLUS (Insightful Corp. 2003) in 2006. Many of the valued features of R were documented in Godfrey (2013) and in the first author's opinion sets the benchmark for statistical software from a blind user's perspective. In the present analysis, we look at R without the added functionality that can be gained through add-on packages. We therefore look at R straight out of the box so to speak, as this is how we will also gauge the other software applications that follow.

Getting started

Installation of R for the blind user is as simple as it is for sighted users. As we will discuss, this is not common, as most other statistical software needs the blind user to make adjustments to the default installation before they can get started doing some work.

The current investigation was done using version 3.0.2 of R, but R's performance has been consistent across versions since version 2.11.0; in contrast, performance has depended on the operating system being employed by the user. Those users of either Linux or Macintosh operating systems work in exactly the same way as their sighted colleagues. Windows users since Vista were not able to use the terminal mode of operation until a solution had quite literally been stumbled upon. A sighted reader of Godfrey (2013) reported his "solution" was to hit the `Alt` key once the cursor had locked up; the standard procedure is then returned.

Use of the R console under Windows has only one small drawback; once the output has scrolled off the screen, the blind user cannot get back to it. The simplest solution is to re-route all output to a text file using the `sink` command. The technique of viewing this text file in a browser and using the `Refresh` button after new work is done, means we can keep abreast of what is happening in our interactive session. Aside from plain text, there are options for converting the output to other formats including LaTeX, HTML, Microsoft Word, and the OpenOffice document formats to name a few. This functionality is only possible through use of add-on packages available from the Comprehensive R Archive Network (CRAN, `http://CRAN.R-project.org/`).

The script window is a very useful and accessible tool for the blind user. The batch mode of operation is another. In either case, all R commands can be entered in plain text, just as done by our sighted colleagues.

Working with data

Neither the data window nor the internal pager for displaying text files are accessible, but seldom does the first author see his sighted colleagues use either functionality nowadays. Commands such as `head`, `tail`, and `str` provide the user with a quick means of understanding the content of their dataset. Data cleaning is commonly done using commands by R users; it might even be fair to say that direct editing of the raw data is very un-R-like. Any blind user needing to directly edit the source data file will have to use alternative spreadsheet software to complete this task.

The fact that data import and export are easily achieved using the `read.table` / `write.table` families of commands, by saving individual datasets with the `save` command and by saving the entire workspace on-exit means the user (blind or sighted) can easily return to the point where they left their work. The existence of the command history file created using the `savehistory` command or when the workspace is saved on exit is another advantage worthy of note. If the user types out a set of tasks (some of which might be repeated) into a script window, they can issue the commands in any order they choose. If commands are single-lined, this is a trivial exercise, but if there are code chunks that will be re-issued, perhaps after a modification to an intermediate step is made, then the efficiency gains become more evident. This flexibility is what gives the various R users, blind or sighted, the options to work in the way that best meets their current needs.

Working with graphics

The handling of graphs in R is different to the other applications considered in this work, as in many instances the user can retrieve text details of the material being graphed. It is common for data or model constructs to be assigned a class attribute which allows the creation of methods for functions such as `print` or `plot`. All information required to turn these objects into graphs is stored when these objects are created, sometimes implicitly but often using explicit assignment to an object. Key examples include the `hist` and `boxplot` functions where the returned object in addition to the plot can be read by the blind user to understand what information is to be presented in the associated graph. At the time of writing, the most notable exception to this useful functionality is the family of commands that create scatter plots.

Getting help

The "help" functionality of R is now a very useful tool as it provides information in consistent, well-formatted HTML. Searching for assistance for a keyword or command name is also easily managed using the `?` or `??` commands. One source of assistance for R users are the extended examples offered in package vignettes. These are normally provided in both raw (un-`Sweave`d) format and as fully processed PDFs (Leisch 2002, 2003). As previously stated, the PDF is not accessible, but the raw (un-`Sweave`d) file is just plain text. The first author uses `Sweave` extensively for report generation, and preparation of study material for his students. The advantage of this method of working is that the output needed for the audience can be created without the difficulties of copying and pasting material between documents, and reduction of the need to re-calculate or look up the smallest of details such as the mean of a variable or the size of a dataset. The intermediate LaTeX file is accessible, and can be processed into HTML if desired. This means his research reports are both reproducible and accessible. Package vignettes can be processed into HTML using the **knitr** package (Xie 2013b,a); there is the potential therefore for all vignettes to be in HTML one day.

Mouse, GUI and dialogue boxes

The graphical user interface known as the R console has pull down menus and for some menu items, hot keys are available. This is most evident in the script window where running single lines or chunks of selected lines can be achieved by the `Ctrl+R` key combination. Of particular note is that R does not actually provide a graphical user interface (GUI) for any of its users that gives access to the vast array of functionality available. There are various front ends for R being developed as add-on packages, but the most widely used of these is not accessible.

The blind user will find it easier to avoid the menus whenever possible by using the R commands that achieve the desired outcome. For example, choosing a CRAN mirror from the list offered through the packages menu is difficult as the contents of the various dialogue boxes are not read by screen reading software. The `chooseCRANmirror(graphics = FALSE)` command provides a text-based solution for this task.

The practical experiment

The experimental exercise was completed using the following code:

```
R> OurData <- read.csv("DataFile.csv")
R> save(OurData, file = "OurData.RData")
R> sink("OutputFile.txt")
R> print(summary(OurData))
R> OurModel <-  lm(y ~ x, data = OurData)
R> plot(y ~ x, data = OurData)
R> abline(OurModel)
R> savePlot("OurGraph.eps", type = "eps")
R> savePlot("OurGraph.pdf", type = "pdf")
R> dev.off()
R> print(summary(OurModel))
R> OurNewData <- data.frame(x = c(10, 30, 50, 70, 90))
R> print(predict(OurModel, newdata = OurNewData))
R> sink()
R> savehistory(file = "OurWork.R")
```

This code was issued through the script window, but could have been piped into R using a command line in Windows or with the addition of a shabang line by Linux/Macintosh users, which tells the operating system that the file is to be executed using R. The addition of the explicit `print` commands is necessary while `sink` is in progress; they are implicitly called if using the script window or the command line mode of operation for review in the console or terminal windows. No matter which mode of operation is chosen, the ability to re-run code chunks makes R a very efficient and effective tool for sighted and blind users alike.

Closing remarks

Sighted R users share code files all the time; this means there are plenty of code examples available for blind users to read. The difficulty is to know which resources reinforce good coding practice. In this and many other respects, the blind user of R has the same challenges as their sighted colleagues.

SAS

SAS (SAS Institute Inc. 2010) is often used in industry, and as a consequence, the blind user should not only be able to "get by", but has to deliver output similar (or in many cases over-and-above) to that of their sighted peers. This may be achieved with great success because of our greater reliance on the ability to automate tasks. The current investigation was carried out using SAS 9.3.

Getting started

Independent installation of SAS has become entirely impossible to the blind user. The dialogue boxes of the SAS deployment manager (which is used for installation) provide no feedback with any of the available screen reading cursors. At times, only the progress bar is spoken. Once the installation is complete there are other tasks required to gain full access to the SAS environment.

In order to view information in the log and output windows, the "Hide cursor in non-input windows" checkbox (checked by default) should be unchecked. This is found in the `Tools > Options > Preferences` dialogue box under the "Advanced" tab. Blind users also need to be sure that the cursor is not blinking, which could leave a screen reader repeating the current character continuously. Note that the "Create listing" checkbox in the "Results" tab (in the mentioned dialogue box) should be checked if any output is to be generated in the output window.

Although this causes the log window to be readable, the output window is only partially so. When the output spans multiple pages (which is often the case), the following steps are recommended.

Programatically clear the log and output windows: Although this can be done from the "Edit" menu (when the focus is on the particular window), this action could cause SAS to become unresponsive when used with a screen reader. The following code could for instance be included in achieving a similar result:

```
dm log 'clear';
dm output 'clear';
```

Print the contents of the log and output windows to text files:

```
%let MyLib = "C:\MySASFiles";
```

```
proc printto
log = "&MyLib.\log.txt" new
print = "&MyFile.\output.txt" new;
run;
```

This creates the two text files, corresponding to the respective windows, in the specified path. The `new` option may be omitted if the user wishes to have additional information appended, rather than creating a file containing only the results from the latest run.

Using HTML output: Because of the great amount of output generated by SAS procedures, and the difficulty blind users could have navigating through it (see Godfrey 2009), we recommend that output also be sent to the "ods HTML" destination.

Ensure that the "Create HTML" checkbox, in the "Results" tab of the "Preferences" dialogue box, is checked. This is the default behavior from version 9.3 onwards. Select the preferred web browser in the "View results using" combo box. To ensure that new output is not appended to the HTML file, include `ods preferences;` at the top of a program, and `ods _all_ close;` or `ods html close;` at the bottom (depending on which ods destinations are to be kept alive). The `ods html newfile = output;` statement would also do this, but only run this after the first HTML output file has been created. More options, such as `proc` and `table` are available to `newfile`, and could further assist in the navigation of output.

As mentioned earlier, HTML output is very easy to navigate, since the usual shortcut keys used in the preferred browser are available, e.g., jumping to different tables, or navigating them.

In addition creation of LaTeX output is easily possible. Although not an absolute necessity, the creation of LaTeX output directly from SAS, is very handy. This is done by

```
ods latex path = "C:\MySASFiles" file = "test.tex";
```

This actually forms part of the LaTeX "tagsets" provided in SAS. The current "tagsets" may be modified, or new ones created, in rendering the desired output, and options such as (`notop nobot`) may be specified to prevent the creation of a preamble or document ending respectively.

Working with data

The data import wizard in the file menu, provides various levels of accessibility. However, since this wizard creates `PROC IMPORT` code anyway, it is better for the blind user to use this (with the appropriate options) to get their data into SAS.

`PROC IMPORT` often creates datastep code in the log file/window, that could easily be formatted in a spreadsheet application for reuse in code. Copying the appropriate portion into the spreadsheet application, and specifying that spaces should for instance be used as column separators, results in a useful list of variables and corresponding formats for use in an input statement, for example.

A sighted user will most often make use of the "Table viewer" to view and edit the contents of a SAS dataset. This viewer/editor has limited accessibility and other work practices have to be followed by the blind user. The first step is to run a `PROC CONTENTS` on the dataset, to confirm its properties. Various options are available for viewing and editing its contents.

The `PROC PRINT` or `PROC SQL` procedures could be used to display the data. If the HTML options are set up as described earlier, the data should be in an easy to navigate table format. If the dataset is very large, request only a partial output of either observations, or variables, depending on the shape of the data. From the menus, the data may also be exported into various formats, for viewing the data in a spreadsheet application, for instance. The same task may be achieved by submitting `PROC EXPORT` with the appropriate commands. By pressing `Ctrl+Tab` twice from the editor window, and selecting the context menu (right mouse click) on the dataset to be viewed, the "View in Excel" option will also open the dataset directly in Excel (not advised for large datasets).

Various datastep options are available for updating and modifying datasets, including again `PROC SQL`. The SAS Add-In for Microsoft Office, allows the user to view and edit a dataset directly in Excel. This could have performance implications, depending on the host system's configuration, but is quite useful.

A subsequent step in data validation is the calculation of descriptive statistics and simple data summaries. All the SAS procedures that may be executed from the editor window, e.g. `PROC MEANS` and `PROC UNIVARIATE`, are available to the blind user. By using the LaTeX tagsets described above, the generated results, along with graphs, are easily included in a report.

Many SAS procedures include the option for storing their output in a dataset. These dataset variables are very useful when assigned to macro variables, and may then be reused in other procedures, or directing program flow. We therefore recommend the use of macro variables wherever possible for increased productivity. As macro variables are not associated with a particular dataset, they can be used to carry information between procedures, or in directing program flow. The SAS documentation may be consulted for more on this. Apart from the ods LaTeX tagsets, many others are available for exporting results. If tagsets for Excel are for instance setup in the correct way, the blind user can export results in the correct formats, e.g. fonts and colors, without sighted assistance. This is especially useful where routine tasks are to be performed. Dynamic data exchange (dde) options may also be used with great success, especially when combined with macro variables, in dropping SAS output in a specified spot in a spreadsheet for instance.

Working with graphics

From the SAS graph window (which pops up after a graph has been requested), the graph may be saved in various formats. Options for "ods graphics" (like those for the LaTeX tagsets described earlier) also allow the creation of graphics in a variety of formats. No information may be gathered by the blind user from these however. Alternative workarounds such as those mentioned in Calder, Cohen, Lanzoni, and Xu (2006) or Loots and van Staden (2007) have to be employed. Only once the graph has been assessed through means such as these, can the user respond accordingly, e.g., adjust parameters, or data values etc.

Getting help

The help documentation that is part of the standard SAS installation is very useful and accessible because it uses HTML. In particular, all mathematical formulae are marked up using LaTeX in the `alt` attributes for the graphic images.

Mouse, GUI and dialogue boxes

The SAS interface most often encountered is the GUI, and may be navigated using standard windowing commands, e.g., `Ctrl+Tab` / `Shift+Ctrl+Tab`, or built-in shortcut keys for particular windows (e.g. `F5`, `F6` and `F7` commands for the editor, log and output windows respectively). Keyboard macros are another tool that improves the effectiveness and efficiency of blind users; they can be recorded, and shortcut keys assigned for frequently-used code. This facility is found in the "Tools" menu.

Apart from the limitations already pointed out, the standard menus are mostly accessible. The main window contains tabs at the bottom, displaying which windows are open, and which are for instance running. The context menus for these are only accessible using the mouse, e.g., right clicking on one of these presents the user with options; the blind user will use `Ctrl+F6` to cycle between them.

Many tools in the "Solutions" menu are unfortunately not useable, since they rely on dialogue boxes and drag and drop techniques. The accessibility of these could be improved by labeling buttons. SAS does provide an "accessibility" system option, which attempts to bring greater access to some of these dialogues.

The practical experiment

We called the following SAS code:

```
%let MyPath = C:\MySASFiles;

ods graphics on/imagename = "OurGraph";
ods html file = "&MyPath.\SAS\OutputFile.htm";
ods tagsets.simplelatex file = "&MyPath.\SAS\OutputFile.tex";

proc import datafile = "&MyPath.\Data\DataFile.csv"
out = OurData
dbms = csv
replace;
getnames = yes;
run;

proc univariate data = OurData; run;

proc reg data = OurData simple
outest = OurModel(keep = Intercept X)
plots(only stats = none) = (FitPlot(nolimits));
model y = x;
run;

proc iml;
use OurModel;
read all into Model;
OurNewData = {10, 30, 50, 70, 90};
Predict = Model[1] + Model[2]#OurNewData;
```

```
print OurNewData Predict;
close OurModel;
quit;
```

using the following batch file:

```
set SASPath = C:\Program Files\SASHome\x86\SASFoundation\9.3
set MyPath = C:\...\Practical\SAS
"%SASPath%\SAS.exe" -config "%SASPath%\SASv9.cfg"
-sysin "%MyPath%\Regress.sas" -nosplash -icon
-log "%MyPath%\log.txt" -print "%MyPath%\OutputFile.txt"
```

N.B. The last lines should not have a line break from the third line onwards of this display. The line breaks were inserted here for presentation purposes.

The destination and names of the log and output files are supplied on the command line. The SAS documentation contains further commands and options when dealing with this type of situation.

As well as the "OutputFile" specified on the command line of the batch file, we have also routed the output to a HTML file, and a LaTeX document within the code.

Closing remarks

In short, if a task can be done using the appropriate SAS procedure, and rerouting the output to the desired destination, the greatest accessibility and independence will be attained.

Given that all SAS programs are plain text files, saved with a ".sas" extension, and can be called from a command line means we can avoid the GUI in most cases. This is known as using SAS in batch mode. The required task was performed using this technique.

When considering reproducible and accessible research, it is worth mentioning the **StatRep** package (Arnold and Kuhfeld 2012) available to LaTeX users. It is similar to `Sweave` for R as mentioned previously in that it allows inclusion of SAS code and corresponding output directly in a LaTeX document. Unlike `Sweave` which is used inside R, **StatRep** is a LaTeX package that links to SAS to obtain the required output and graphics files.

SPSS

SPSS (SPSS Inc. 2012) has received plenty of attention from blind users as it is used in many disciplines for undergraduate and postgraduate coursework. Attempts to get SPSS working well with screen readers have been made for a number of years (Orme, Dimigen, and Roy 1999). It is unfortunate that some of the efforts made for earlier versions of SPSS did not survive from one version to the next. One key problem when discussing the accessibility of SPSS is that the information in previous reviews such as those appearing by Orme *et al.* (1999) in Nature have gone stale, and therefore can offer new users unrealistic hopes of using SPSS as it is today. We found documents that refer to SPSS versions as old as version 8 that hold next to no value for users of versions of SPSS since version 20. These concerns are not unique to SPSS of course.

Getting started

The greatest challenge for blind users of SPSS is the installation process. We installed version 21 of the software with the assistance of IT support staff. It would have been possible to achieve this on our own, but the number of steps and quite specific instructions that need to be followed makes this task daunting for less-experienced computer users.

Installation is made arduous because SPSS uses Java for much of its functionality. Java applications can often fail unless the Java Access Bridge and Java Run-time Environment are both installed. Detailed instructions are available on the SPSS website to assist the additional work required for the installation process. Some of the tasks required to get this software up and running are done as part of the standard installation of SPSS. This software stands out as the only software we tested that explicitly refers to the needs of blind users when it asks if the user intends to use the commercial screen reader known as **JAWS**. While this is commendable, there are many other screen readers in use. **JAWS** may well have market dominance in the English-speaking world, but we feel that the developers of SPSS should not be so closely tied to just one screen reader.

Once the Java Access Bridge is installed, the menus become accessible; without it, this application is utterly unusable. We commend IBM for the plan to develop the installation process even better for version 22; blind users should soon be able to complete all necessary installation steps without sighted assistance even if we are the most novice of computer users.

The practical experiment

When performing the set tasks, we found the following points to note:

1. The text data import wizard had dialogue boxes that were not read properly. The options were read, but we could not tell what the options were for. This was crucial for the presence of the header row of our data set saved in csv format.

2. The menu for creating graphs was not accessible. This is because the particular graph type to be chosen is done using icons that look like the graph to be created. This problem is avoided if we use the syntax window instead of the menus.

3. The dialogue for creating a simple regression model was accessible. The use of the `Tab` and `Shift+Tab` keys moved the cursor around the elements; the screen reader read out the necessary information for each element. The `Paste` feature for sending the relevant code to the syntax window means that a blind user could be given code chunks to keep as templates for later use.

4. The help functionality did not interact well with the **JAWS** screen reader. The command syntax for some examples was given which is helpful, but the command syntax guide included in the help menu is a PDF with more than 2500 pages. Searching this document is not practical for the blind user.

5. We did not find out how to create a scatter plot with the fitted line added using the command syntax or menus within a predetermined time frame. This illustrates the need for better quality support documentation, including command syntax, especially if SPSS is to be used by blind students.

6. We did not find out how to save an individual graph as it was created. We have saved the entire output window twice; the first time in accessible HTML with the JPG graph type and a second time so that we could save the graphs as encapsulated postscript graphics for inclusion in LaTeX documents. Automatic filename generation was used to save the individual graphs. If the authors seriously contemplated using SPSS for their own work, they would need to know how to save individual graphs in the right format and using self-selected filenames.

7. We did not find out how to make the predictions using commands. We could do this by adding rows to the data as this window was read well enough by **JAWS**. The only disappointment about the accessibility of the data window was that the row numbers were not read aloud automatically as were the column (variable) names.

8. We could not find the correct syntax to generate the fitted line for addition to the scatter plot. This was completed using the GUI but no code was able to be generated in this dialogue box.

The code saved from the syntax window is as follows:

```
GET DATA  /TYPE = TXT
 /FILE = "ToUse.csv"
 /ENCODING = 'Locale'
 /DELCASE = LINE
 /DELIMITERS = ","
 /ARRANGEMENT = DELIMITED
 /FIRSTCASE = 2
 /IMPORTCASE = ALL
 /VARIABLES =
 x F5.2
 y F6.2.
CACHE.
EXECUTE.
DATASET NAME DataSet1 WINDOW = FRONT.

SAVE OUTFILE = 'Attempt.sav'
 /COMPRESSED.

GET
 FILE = 'Attempt.sav'.
DATASET NAME DataSet1 WINDOW = FRONT.
DATASET ACTIVATE DataSet1.
DESCRIPTIVES VARIABLES = x y
 /STATISTICS = MEAN STDDEV MIN MAX.

DATASET ACTIVATE DataSet1.
GRAPH
  /SCATTERPLOT(BIVAR) = x WITH y
  /MISSING = LISTWISE.
```

```
REGRESSION
 /MISSING LISTWISE
 /STATISTICS COEFF OUTS R ANOVA
 /CRITERIA = PIN(.05) POUT(.10)
 /NOORIGIN
 /DEPENDENT y
 /METHOD = ENTER x.

* Export Output.
OUTPUT EXPORT
  /CONTENTS  EXPORT = ALL  LAYERS = PRINTSETTING  MODELVIEWS = PRINTSETTING
  /HTML  DOCUMENTFILE = 'OUTPUT.htm'
     NOTESCAPTIONS = YES STYLING = YES  IMAGEFORMAT = JPG
  /JPG  PERCENTSIZE = 100  GRAYSCALE = NO.

* Export Output.
OUTPUT EXPORT
  /CONTENTS  EXPORT = ALL  MODELVIEWS = PRINTSETTING
  /EPS  IMAGEROOT = 'OUTPUT.eps'
     WIDTH = PERCENT(100)  PREVIEW = YES  FONTHANDLING = REFERENCES.
```

Closing remarks

We must question the portability of SPSS practices between users. While blind users will almost certainly need to work using the text commands in the syntax window, their sighted colleagues may not be doing so. Working environments need to accommodate the blind user by ensuring that the working practices always keep a log of the actions taken by the users. In this regard, SPSS is not alone and our point is not meant as criticism. If it were impossible to keep a log of the text commands that would re-create all actions taken using the menus and dialogue boxes then we would be critical.

Minitab

The last fifteen years have been hard on blind Minitab (Minitab Inc. 2012) users, including the first author whose experience with the software dates back to 1994. When blind people reflect on our ability to use the same software as our sighted peers, we can look to several major turning points that have widened the gap between us. The widespread introduction of the Windows operating system is the one worth the most mention for the Minitab story as it relates to blind people.

Under DOS versions of Minitab, blind and sighted users were on par with one another. Everyone needed to use typed out commands and a small array of menu items; everyone had the same help facilities and manuals; and everyone read their output and graphs that were printed out in plain text characters.

The introduction of high resolution graphics signaled the start of the divergence blind and sighted users would then take with Minitab from that point onwards. One good initiative in

Minitab is the ability to switch to low and high resolution graphics using the `GSTD` and `GPRO` commands respectively. Unfortunately, some commands (such as `acf` for an autocorrelation function) automatically revert to a high-resolution format with no apparent low-resolution equivalent.

Even when the predominant method of working changed to the GUI under Windows versions of Minitab, the output has generally been printed as text in the session window. Some functions did start to embed text in an inaccessible graphic form. A key example was the autocorrelation function which is very readable in the plain text form, but not in the high resolution form. A blind student would need to create two different analyses in order to gain knowledge as well as the format required for their assignments. The high resolution form gives the sighted user a confidence interval that is not available in the low-resolution form, so the blind user is left at a disadvantage.

For several years, the blind user was left using the last DOS version of Minitab while everyone else was using Windows versions. This meant we were not able to access newer Minitab functionality. Ultimately blind users were able to use the Windows operating system with the commercial screen readers then becoming available. This gave us the chance to catch up with our sighted peers.

Versions 12 and 13 were probably the height of accessibility for Minitab. Although many dialogue boxes were not fully functional, the experienced blind user could muddle their way around, or fall back on the command line codes. The worksheet was readable and the user could move around and read as well as edit the content of a worksheet. Having access to the printed documentation available at the time could make it possible for the blind user to function in Minitab.

Some of the help functionality offered via the menus in Windows versions of Minitab was created using the PDF format and therefore was of less and less use as new versions were released. Thankfully the use of PDF files for help (available from the web) is complemented by the use of the standard Windows help format (available on installation). Moves towards newer document types were not the only change however. The worksheet that was accessible under versions 12 and 13 was inaccessible in version 14. The addition of more visually appealing menus in version 15 where unavailable items are grayed out is a symptom if not the cause of what made the menus read poorly by screen reading software. To continue using Minitab was only practical if the user could memorize the order of the menu items or had access to the quality documentation with the command language laid out in full.

Documentation for completing the most basic tasks is now generally focused on the menus and dialogue boxes mode of operation. Much of the material commonly found in such documentation includes graphics of screen shots to assist the sighted person find their way. The better documentation shows the user the path through the menus using plain (and therefore readable) text, but seldom does this include the hot keys for the specific items. If Minitab is to be used by blind users in future, there will need to be some alterations to the appearance of menus or some investment in writing specific scripts for Minitab that will assist the screen reader read the content of the menus and dialogue boxes. The more realistic approach is for the blind user to rely on the use of commands. Help on the syntax of known commands is available by typing `help xxx` at the command prompt.

The first author was able to complete the assigned task in Minitab 16.2.3 using the knowledge built up under older versions, and a sighted research assistant looking over his shoulder. This is acceptable as a temporary solution such as needed by a non-statistics student taking a single course in statistics, but is certainly not viable practice in the long run.

To get started, the blind user needs to enable the command language of Minitab so that the commands implemented by dialogues are printed (for future reference) and the command prompt is available for typing new commands. Unfortunately, the command prompt is not present by default so the blind user needs to either know the required sequence of keystrokes (`Alt+D>E` for version 16) or have sighted assistance in the setup phase. Blind users would need their colleagues to also ensure that the text version of commands is stored in the session window if they are to collaborate effectively.

The practical experiment

The (edited) set of commands for the required task (as printed in Minitab's session window) were:

```
MTB > WOpen "c:\...\Data\DataFile.xlsx";
SUBC>   FType;
SUBC>     Excel.

MTB > wsave "test.csv"
MTB > wsave "test.mtw"

MTB > desc 'y' 'x'

MTB > Fitline 'y' 'x'

MTB > read c3
DATA> 10
DATA> 30
DATA> 50
DATA> 70
DATA> 90
DATA> end

MTB > Regress 'y' 1 'x';
SUBC>   Constant;
SUBC>   Predict c3.
```

Of particular note is our failure to find the command needed to save the graph window and the command history. The menus were used for these steps, but no code was given in the session window; we did know the graph was saved because feedback was given in the session window. The saving of the command history was not reported in the session window so we needed to check the intended destination folder for the file created. We raised this point with Minitab's technical support and were pointed to the use of the `GSAVE` command which is issued as a subcommand of the main plotting function used. The options for the graph type to be saved were limited to Minitab's proprietary format and some older types not commonly used in LaTeX documents. Creation of encapsulated postscript or PDF formats requires the use of the inaccessible menus and dialogue boxes. The blind user therefore needs to include the GSAVE subcommand at the time of graph creation if they are to avoid re-issuing the code to save the graph.

The output that was printed in the session window is readable by the **JAWS** screen reader, but the use of a free screen reader known as **NVDA** was necessitated in order to run this experiment on a lab computer. **NVDA** can be run from a memory stick so it is a very convenient tool for blind students needing to work in computer laboratories. Two versions of **NVDA** were used. Initially, version 2011.3 was tried but did not deliver any feedback from the session window as was expected; use of version 2013.2 gave much better results. This is highlighted in order to remind blind users that we need to have more than one screen reader option available if we are to maximize our capacity. A time-limited demo version of Minitab was obtained in order to test the above findings using **JAWS**. This installation was fairly easy when compared to SPSS or SAS, but not as easy as the installation of R.

Closing remarks

In conclusion, we find that Minitab is usable if absolutely necessary, but rather impractical – especially if no background knowledge is assumed or available from well-informed staff or colleagues. The authors call on the developers of Minitab to improve the accessibility of their product. Given it was once accessible, we are sure it could be again.

How can we improve the accessibility of statistical software?

The short answer to the question posed is that the responsibility of making software accessible lies with the developers. This does however, downplay the role that blind people themselves must play in making sure the development community is aware of our needs. In order to practice what we preach, the comments regarding Minitab were forwarded to their technical support team on 11 December 2013. We can only hope that provision of such feedback does lead to tangible improvements.

There are two strategies the developers of any software solution can follow if they want their software to be accessible. They could use the right development toolkits and rely on the accessible aspects of operating systems, or they could develop the necessary scripts that would help screen reading software render the material presented by their software. The reality is that both strategies need to be followed for most software. The more built-in accessibility that exists the easier it is to add functionality for blind users via add-on scripts for their screen reading software. For example, the accessibility features of SPSS include a dictionary file so that the **JAWS** screen reader renders some text more efficiently.

In the past, most scripting for screen reading software has fallen to the blindness community to initiate. The commercial screen reader companies tend to ensure their product works well with the most common applications used by all computer users. This means that the array of screen readers for Windows operating systems is greater than the options for other operating systems. Then, the screen readers for Windows are maintained at a level needed for most blind users, and that means ensuring access to such software as the Microsoft Office range of applications. Expecting screen reader developers to create scripts for every application that might be used by blind people is unrealistic at best and at worst absolves the statistical software developers from their moral responsibility to provide for all users.

Most commercial statistical software is produced and marketed in the United States of America where there is anti-discrimination legislation in place to force federal agencies to procure accessible software for their employees, and federally funded colleges and universities to provide for their students. In an open letter jointly written by the US Departments of Justice and Education to all college and university presidents (available from `http://www2.ed.gov/about/offices/list/ocr/letters/colleague-20100629.html`), we found the following text:

> US legislation prohibits "colleges and universities from affording individuals with disabilities with an opportunity to participate in or benefit from college and university aids, benefits, and services that is unequal to the opportunity afforded others. Similarly, individuals with disabilities must be provided with aids, benefits, or services that provide an equal opportunity to achieve the same result or the same level of achievement as others. A college or university may provide an individual with a disability, or a class of individuals with disabilities, with a different or separate aid, benefit, or service only if doing so is necessary to ensure that the aid, benefit, or service is as effective as that provided to others."

The consequence of this declaration is that software developers are (theoretically) being pressured into providing for the disabled population using market forces. From the preceding sections of this article, we can see that some developers are fulfilling their responsibility while others have some work to do.

In the authors' opinion, if these laws are used to create and maintain the accessibility of software used by blind students and federal employees in the United States of America, there will be a flow-on effect for the blind people not directly covered by USA anti-discrimination legislation. We found that the developers of SAS are deeply aware of the need to comply with this legislation by visiting their website at `http://www.sas.com/industry/government/accessibility.html` where we read the following extract:

> Section 508 of the U.S. Rehabilitation Act of 1973, as amended, mandates that when the federal government purchases most electronic information and technology (EIT), including software applications, it must ensure that the EIT provides access to and use of, information or data to federal government employees with disabilities that is comparable to the access provided to federal government employees without disabilities. Moreover, under the act, the federal government is also obligated to provide access to information and data to members of the public with disabilities that is comparable to the access provided to the public without disabilities.
>
> SAS understands the importance of Section 508 requirements to our customers. The regulations adopted by the federal government to implement Section 508 generally require U.S. government agencies to purchase software solutions that are most compliant with Section 508 requirements. In addition, other customers, including state governments and international bodies, increasingly require compliance with accessibility standards when making procurement decisions. Beyond the legislative compliance, SAS recognizes that universal design and accessibility are good business strategies.

Perhaps we need to rely on the goodwill of software development companies to employ experts in accessibility for blind users. Or, perhaps we need software developers to remember that good design for accessibility usually means good design for all users. Building accessibility features into software does require the developers to maintain higher standards of programming consistency, which should improve the quality of their software in the long run.

Conclusion

We have reviewed four commonly-used statistics packages for their accessibility and usefulness for blind users. Our reviews expose a number of strengths and weaknesses for each of the four applications considered. It is clear that no software is uniformly superior for blind students or practitioners.

Ultimately, we conclude that a blind user should use the same software as their classmates or work colleagues in order to maximize the sharing of knowledge and experience. The way we need to work will depend on the choice of application but there are some common threads.

1. We rely on the use of code instead of the GUI. The ability to process code without actually opening the GUI version of the software (often called batch mode) avoids the need to interact with the less than totally useful GUI. As a consequence, every task that is possible using the GUI, or using mouse clicks, must be included in the command language.

2. We need to save the output from our analyses in file formats that are easily read by our screen reading software. Ultimately, this means that any information that is created for graphics must be retrievable as plain text.

3. We need accessible documentation to support our methods of working.

At this time we feel that none of the statistical software applications reviewed can fully deliver on even these simple criteria. There are of course a large number of other software options that could be evaluated using this simple set or the more detailed set of criteria given earlier.

There is a moral obligation on those of us that know how statistical software could meet the needs of blind people to work with software developers to help them improve their products. The authors look forward to such opportunities.

Acknowledgments

The authors wish to acknowledge the assistance received from numerous individuals over the years that have helped them use the various statistical software options. In particular, the first author wishes to note the interactions with one of his colleagues (K. Govindaraju) whose knowledge of Minitab has proven invaluable for the preparation of this manuscript. We would also like to thank the journal's editorial team for their helpful comments and assistance in preparing this document.

References

Apple Inc (2011). ***VoiceOver*** *Version 4.0.* URL http://www.apple.com/accessibility/voiceover/.

Arnold T, Kuhfeld WF (2012). "Using SAS and LaTeX to Create Documents with Reproducible Results." URL http://support.sas.com/resources/papers/proceedings12/324-2012.pdf.

Calder M, Cohen R, Lanzoni J, Xu Y (2006). "**PLUMB**: An Interface for Users Who Are Blind to Display, Create and Modify Graphs." *ASSETS'06*, pp. 263–264. URL www.catea.gatech.edu/scitrain/kb/FullText_Articles/PLUMB.pdf.

Freedom Scientific (2012). ***JAWS*** *Version 14.* Freedom Scientific, St. Petersburg, FL. URL http://www.freedomscientific.com/.

Godfrey AJR (2009). "Are Statistics Courses Accessible?" In *Proceedings of the Workshop on E-Inclusion in Mathematics and Science 2009*, pp. 72–80. Fukuoka, Japan.

Godfrey AJR (2013). "Statistical Software from a Blind Person's Perspective: R is the Best, but We Can Make it Better." *The R Journal*, **5**(1), 73–80. URL http://journal.R-project.org/archive/2013-1/godfrey.pdf.

Insightful Corp (2003). *S-PLUS Version 6.2.* Seattle, WA. URL http://www.insightful.com/.

Leisch F (2002). "Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis." In W Härdle, B Rönz (eds.), *Compstat 2002 – Proceedings in Computational Statistics*, pp. 575–580. Physica Verlag, Heidelberg. ISBN 3-7908-1517-9.

Leisch F (2003). "Sweave, Part II: Package Vignettes." *R News*, **3**(2), 21–24. URL http://CRAN.R-project.org/doc/Rnews/.

Loots MT, van Staden PJ (2007). "Visualisation of Time Series by the Visually Impaired." In *The Sixth Southern Hemisphere Symposium on Undergraduate Mathematics and Statistics Teaching and Learning*. Calafate, Argentina.

Minitab Inc (2012). *Minitab Statistical Software Version 16.2.3.* Minitab Inc., State College, PA. URL http://www.minitab.com/.

NVDA Team (2014). ***NVDA*** *Version 2014.1.* URL http://www.nvaccess.org/.

ORCA Team (2006). ***ORCA*** *Version 3.10.2.* URL http://projects.gnome.org/orca/.

Orme R, Dimigen G, Roy AWN (1999). "A Screen Reader Solution for Accessing SPSS 9.0 without Sight." Software review published by Nature (online), URL http://www.nature.com/nature/software/screen/screen2d.html.

R Core Team (2013). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

SAS Institute Inc (2010). *SAS/STAT Software, Version 9.3.* SAS Institute Inc., Cary, NC. URL http://www.sas.com/.

SPSS Inc (2012). *IBM SPSS Statistics 21.* SPSS Inc., Chicago, IL. URL http://www.spss.com/.

Xie Y (2013a). *Dynamic Documents with R and **knitr**.* Chapman and Hall/CRC. URL http://yihui.name/knitr/.

Xie Y (2013b). ***knitr**: A General-Purpose Package for Dynamic Report Generation in R.* R package version 1.5, URL http://CRAN.R-project.org/package=knitr.

Reviewers:

A. Jonathan R. Godfrey
Institute of Fundamental Sciences
Massey University
Palmerston North, New Zealand
E-mail: a.j.godfrey@massey.ac.nz

M. Theodor Loots
Department of Statistics
University of Pretoria
Pretoria, South Africa
E-mail: theodor.loots@up.ac.za

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 58, Software Review 1 *Published:* 2014-06-19
June 2014