**Interaction between JAWS and Minitab 15**

Minitab 15 was not designed with screen-reading software in mind. Therefore, any successful interaction between JAWS and Minitab would be a bonus and a pleasant surprise.

I tested JAWS with Minitab 15 by performing a variety of common actions and noted the success of JAWS integration each step of the way.

I assessed many functionalities, but I always tried to find methods of using the keyboard only, because using a mouse inherently requires you to see its position on the screen.

**Navigating the menu bar**

JAWS does not read the names of the menus in the Top Menu Bar (File, Calc, Stat, Graph, etc), regardless of how you get there (either by typing Alt to select the bar, then navigating between choices with left/right arrow keys, or by clicking the mouse on a menu choice). The only way I found to make JAWS say anything about that top menu is to make it read them all at once from left to right using the JAWS command “Say Line” CAPS+I. Doing so makes JAWS say “File Edit Data Calc Stat Graph…”, which is not very useful. JAWS will say “File” only if you submit JAWS’ “Say Word” command CAPS+I, but it only says “File” no matter what menu choice is actually selected. You’d expect JAWS’ “Say Next Word” command CAPS+L to speak the subsequent menu name, but when you try JAWS again says “File”. Therefore, it’s difficult to ascertain which menu you’re choosing. You might think you could simply type Alt and then count the number of menus over from the File menu (i.e. type Alt and right-arrow x4 to select the Stat menu), but that’s not foolproof: if the cursor happens to be hovering over, say, the Graph menu, then typing Alt will activate Graphs instead of File. The only thing that saves the user from becoming lost in this case is that JAWS *is* very consistent about reading the contents of each menu, so the identity of the menu can be ascertained by listening to the choices within it (i.e. the user types Alt, then the down arrow, and JAWS says “Basic Statistics”; the user knows the Stat menu is activated.)

When JAWS reads menu choices, users can distinguish between a command and a submenu that has its own nested choices: commands end in “…” and a nested menu does not; therefore JAWS says “dot dot dot” after the name of a command that open a dialog box, and JAWS does not say “dot dot dot” if the choice is a menu within the current menu.

Comparing this menu-reading functionality to Excel 2007, its ribbon menu enables better JAWS behavior: typing Alt accesses the ribbon menu, and using the arrow keys makes JAWS read the Menu Tab you’re currently viewing as you arrow-key through them.

A suggestion for Minitab is to ensure all menu items have different names so listeners can distinguish them easily. The only offender I found was the “Editor>Go to…” command and the “Editor>Go to” submenu with more choices of its own. Only JAWS reading “dot dot dot” in the former case distinguishes them to the listener.

**Using dialog boxes**

A potentially fatal problem exists for dialog boxes: I found no way to move columns from the list of available columns into the appropriate dialog box field without using the mouse to click inside the list of columns – an action that requires visual cues to navigate the mouse. Once that list of available columns has been clicked, JAWS reads the choices – both their column number and name, if the user provided a name. But getting into that list seems impossible without using your eyes to move the cursor and click into that column list.

Aside from that dilemma, JAWS behavior in dialog boxes is decent, but definitely inconsistent between different types of dialog boxes: noticeably between dialog boxes in the Stat and Graph menus, but more surprisingly between “newer” and “older” dialog boxes within the Stat menu.

For example, radio buttons: in some Stat commands like the Censoring subdialog box of the Parametric Distribution Analysis (right censored) Reliability command, JAWS performs nicely: it says the name of the radio button, and also calls it “radio button 1 of 3”. However, the new-for-15 Poisson Test commands in Stat>Basic Stats do not behave this way: radio buttons there are not called “radio buttons” but only “buttons”; the text of the choice is spoken, but JAWS does not say “1 of 3” or the appropriate equivalent. This lesser functionality is also the case for all Graph dialog boxes. A pattern exists: when JAWS calls it a “radio button”, it also says “1 of 3”; but when JAWS calls it a “button”, it does not say the number. That consistent distinction suggests some pattern in the way these dialog boxes were programmed exists behind the scenes, which JAWS is recognizing as it decides what to speak.

Similarly, checkboxes are described inconsistently: in 1-Sample Z, JAWS says “perform hypothesis test checkbox not checked. To check press spacebar” . In 1-Sample Poisson, it says “perform hypothesis test button. To activate press spacebar”. Clearly there’s different programming underlying identically-appearing dialog box items. Perhaps older vs. newer dialog boxes is the issue (because 1Z is “old” and 1Poisson is “new”)?

Likewise, it’s clear JAWS is noticing behind-the-scenes programming architecture when it says thing like “graphic 589” in a variety of places – some more surprising than others. For example, the Graph Galleries that appear after choosing a command from the Graph menu are rendered useless because, instead of reading the choices for, say, histogram as “Simple, With Fit, With Outline and Groups, With Fit and Groups”, JAWS speaks those choices thus: “graphic 306 button, graphic 757 button, graphic 421 button, graphic 96 button”. That’s useless to a listener, although it makes sense: clearly graphics exist there, so even nonprogrammers can imagine those graphics are coded or entitled with a number, which JAWS detects and speaks.

However, this same behavior also occurs in an inexplicable place: when you put the cursor in a large multi-argument field of “older” dialog boxes in the Stat menu. For example, in 1-Sample Z, when I put the cursor in the field under “Samples in columns”, JAWS says “Edit. Graphic 542, Graphic 815, Graphic 542”. It’s a mystery to me how JAWS is detecting a graphic in that location. In the same dialog box, this does not occur when I put the cursor in the smaller field “Standard deviation”, so this behavior seems associated with large fields that accept multiple columns. But this does not occur with similar fields in commands from the Graphs menu.

In File>Open Worksheet, the button to “Look into Minitab Sample Data Folder” is spoken as “Graphic 876 button”, rendering it useless to the completely blind.

**Submitting commands using command language instead of dialog boxes**

This is obviously an attractive option, because it frees the user from the requirement to navigate the GUI, which inherently requires coordination between hands and eyes.

It’s easy enough to enable the command prompt (ensure Session Window is active by CTRL+TABing until JAWS speaks its name to indicate it’s the active window. Then click Alt to activate menu bar and use arrow-key to access the Editor menu. Then down-arrow to Editor>Enable Commands. JAWS reads the choices along the way, excepting the “Editor” top menu name, as previously noted.) Then, with Session Window active, they can distinguish between “MTB>” and “SUBC>” prompts by using JAWS’ “Say Line” command CAPS+I. Then, as you type commands, JAWS speaks each letter as you type. That seems all good.

But there’s one pitfall (although a clunky workaround exists): JAWS will not automatically read any error message that appears in response to a user typing an invalid command or subcommand. JAWS’ speech is no different in the two cases where the user correctly enters a subcommand, and where a user enters an invalid subcommand. See the visual difference below:

MTB > onet c1;

SUBC> conf 99;

SUBC> badcommand;

\* ERROR \* Subcommand BADC is out of sequence or is not valid with this

 \* command.

\* ERROR \* Subcommand ignored.

SUBC>

For both subcommands entered above (the valid “conf 99” and the invalid “badcommand”), JAWS reads letter by letter as I type them, then says the word “Enter” when I type enter to submit the command. But then, JAWS makes no audible distinction between the results of submitting those subcommands. Thus, a blind user can’t be sure a command or subcommand was accepted, any time one is entered.

The workaround for this is simple and effective, but it is a shame users must do this for every command and subcommand they ever type. After typing a command and pressing Enter, the user must use the JAWS command “Say Previous Line” CAPS+U. If, after doing so, JAWS speaks “MTB>…” or “SUBC>…”, then the user knows the command was accepted and valid. If it reads “error subcommand ignored”, then the user knows the previously submitted command was invalid. Using CAPS+U again will read the first error message as well, although it will read “star command” first because that text comes on its own line, and then it will read “star error star subcommand…not valid with this” as you resubmit JAWS’ “Say Previous Line” command.

**Navigating within the worksheet**

This aspect was a chief concern, and unfortunately Minitab fares poorly. In Excel, as you navigate the worksheet JAWS speaks the contents of the cell, followed by its column/row coordinates. That occurs whether you’re clicking from one cell to another, or moving around the spreadsheet with the arrow keys.

JAWS hardly speaks at all in Minitab’s worksheet. Activating the worksheet makes JAWS read the worksheet title (CTRL+Tab activates the various windows one by one, with JAWS speaking the name of the active window each time). Also, JAWS will read a column number and its title and contents (if provided), but only when you “select entire column” by clicking the column number (the cell above a column title). Unfortunately, I can’t find a keyboard command to do the same thing (select the entire column), so someone relying on keyboard commands is out of luck there. Also, JAWS will speak the row number and contents when you click on the row number to select entire row, although not as reliably as the equivalent column action (it seemed to randomly skip or repeat certain cells in a row according to no discernable reason or pattern). But again, there’s no key command to “select entire row” that I could find.

Regarding cell coordinates, JAWS speaks nothing in a Minitab worksheet. The only workaround is to make the worksheet active, and then choose Editor>Go to. This dialog box selects a cell when you specify a row and column coordinate. JAWS reads this dialog box perfectly. That’s the only way a user could determine where they are in a worksheet.

Regarding cell contents, JAWS reads contents as you type, but I found no reliable way to read contents of cells that are already entered. The JAWS commands to “say word” or “say line” behaved erratically, sometimes reading contents of a neighboring cell, or sometimes reading “C1 C2 C3…” and so on unto infinity.

**Interacting with Session output**

JAWS reads Session output exactly as it would read a paragraph in a book: left to right, line by line. This behavior causes big problems, since so much of the output (ANOVA tables, descriptive stat tables) is meant to be understood as columns or two-way tables. For example, consider the following output from 1-Sample Sign test:

 Confidence

 Achieved Interval

 N Median Confidence Lower Upper Position

PriceIndex 29 144.0 0.9386 110.0 210.0 10

 0.9500 108.5 211.7 NLI

 0.9759 101.0 220.0 9

JAWS reads this output as follows: “Confidence, Achieved, Interval, N, Median, Confidence, Lower, Upper, Position, PriceIndex, 29, 144.0, 0.9386…”

This is virtually useless to a blind person. They could use JAWS “Say Next Line” CAPS+O command, to read line by line, and try to piece together the structure in their imagination, but the limitation is still present.

It would be more useful to read most of our output by columns instead of rows, or even somehow make a 2-way table-method of reading output, since that’s the typical output format. For example, in the output below, the columns are the statistics and the rows are the variables. The values only make sense when you understand the row and column to which it corresponds. Reading it like a book is not helpful.

**Descriptive Statistics: C1, C2, C3**

Variable N N\* Mean SE Mean StDev Minimum Q1 Median Q3

C1 10 0 0.103 0.257 0.812 -1.545 -0.321 0.026 0.853

C2 10 0 -0.345 0.248 0.786 -1.688 -1.046 -0.198 0.304

C3 10 0 -0.040 0.285 0.902 -1.368 -0.695 -0.069 0.380

To effectively navigate the Session Window, despite the line-by-line speaking that’s problematic due to our output’s table layout, I used CTRL+ALT+M to open the Project Manager, used up and down arrow keys to browse my Session history and listen to JAWS read the title of each piece of output, typed Enter to select one, then CTRL+M to jump to the Session Window, then CAPS+I to read the heading and confirm I’m located at the analysis I’m seeking, and then CAPS+O to read subsequent lines one-by-one.

**Accessing Help documentation**

It’s easy enough to open Help by navigating the menu bar using Alt and arrow keys. JAWS then reads the choices as you browse Help’s TOC. However, once you type Enter to open the selected Help Page, I found no way via keyboard to jump from the left TOC pane to the right pane where the help content actually appears. If somehow the user was able to click the cursor to activate the pane containing the Help descriptions, then JAWS reads it like a book, but keyboard commands are better for the blind. Likewise, I couldn’t find keystrokes to switch between the Contents tab and the other tabs, Index and Search.

Opening Help from either a dialog box or Help>Help makes the left TOC pane the active pane, so the user must find a way to click into the right pane where the actual help content resides.

Note: I suspect the clunky interaction between JAWS and \*.CHM Help files is not limited to Minitab, and Minitab is abandoning CHMs starting with Rel 16 anyway.

Also, many Help links, especially all “see also” links contain pop-up menus. I found no way for JAWS to read these choices. The example link in Reliability>Parametric Distribution Analysis (Right censored) is a popup menu linking to two examples. JAWS is helpless there.

**Interacting with Graphical Output**

I ran Capability Analysis (Normal) because its output is entirely graphical. When you select different regions of the graph, JAWS’ speech to announce each one was a letter ID: “M” when clicking on the Process Data section; “W” when clicking the Histogram graph region. Sometimes JAWS would say the letter twice. Immediately after clicking OK to run the command, JAWS will speak the name of the graph by reading the Title of the graph window “Process Capability of C1”. But one time, it appended the phrase “Release 14 graph” to the end of this phrase. I could not replicate this behavior, and I was exclusively using Minitab 15.

The main issue with graphical-only output is how to hear the statistics that appear on the graph image; I suppose we must abandon hope of “describing” the graphs themselves, since graphs are inherently tools to represent data visually. The challenge of getting JAWS-like software to say something useful about graphs in general is an issue that transcends Minitab I’d guess. But what about the statistics that could’ve been output to the Session Window just as easily? First, I suggest we make an option in every graph command to display or store any statistics that might appear on the graph. The Capability Analysis commands display a wealth of statistical text embedded in the graph image. Let the user store those values in the Session window, where JAWS has a better chance of reading them.

But in the absence of that option, here’s the behavior: JAWS can be induced to read some of the statistical output on the Capability Analysis graph, but I only succeeded in making it read the first few lines, counting from the top down. Furthermore, JAWS suffers from the same limitation it has when reading text from the Session Window: it reads left-to-right, top-to-bottom without regard for the breaks between boxes. For example, consider this section of the output:



If I select the Process Data box and type CAPS+H (JAWS “Say Sentence” command), JAWS says “Process Data Within LSL 0.5 Overall”, jumping from the upper-leftmost box to the upper-rightmost box as if they were a continuous line of text separated by a histogram. For some reason, CAPS+H also served to move the boxes around the graph area too, surely unintended behavior.

**Test of Stat>DOE>Factorial>Create Factorial Design**

The subdialog box Display Available Designs is useless: JAWS is unable to read the text in the rows, columns, or cell contents of this table. A table like this requires visual coordination of row and column to determine a design’s resolution, so it’s useless to blind people, and not designed in a way JAWS can understand.

JAWS can read the text in the Designs subdialog box, but like the Session Window and Capability Analysis output, JAWS reads the table line-by-line, top-to-bottom, which is not very useful. It would be more informative if it read down columns from left to right, or read each row left-to-right, but referring back to the column title between each row element. Here’s the dialog box:



JAWS reads “Designs, Runs, Resolution, 2 star star (k-p), ½ fraction, 8, IV, 2 star star (4-1), Full factorial…”

It would be better if it read “Designs: ½ fraction, Runs: 8, Resolution: IV, 2\*\*(k-p): 2\*\*(4-1); Designs: Full factorial, Runs: 16, Resolution: Full, 2\*\*(k-p): 2\*\*(4)” to maintain the connection between the data in the rows and the columns of the table, which provide the context for that data.

In the Factors subdialog box, JAWS only reads the bold column headings, saying “Factor Name Type Low High”. It could not read any other contents, and did not treat the dropdown menus like others in different dialog boxes. JAWS was usually good at describing dropdown menus: even telling you how many choices were in the dropdown, and which one you had active. None of that behavior occurred with these dropdown menus.



**A note on SPSS**

The Help file in the JAWS software has a section called “Popular Applications in JAWS”. In contains JAWS commands that are specific to certain specific software applications. I noticed SPSS was in that list of 3rd party applications with specialized JAWS commands. Their special JAWS commands were very few and underwhelming, but the fact that their name appeared in that list reflected well on them.

**Conclusion**

Minitab clearly has a long way to go before it plays ideally with JAWS. I don’t think a completely blind person could use Minitab independently. However, Minitab is “almost there” in most aspects: most of its offences were inconveniences for which a clunky workaround exists. There were few true “deal breakers”. This document should clarify Minitab’s general behavior for users, but it could also be useful for Minitab programmers, because the nature of the JAWS/Minitab misbehaviors seemed to indicate patterns in the programming architecture behind the scenes, especially regarding how the GUI was programmed.

Although Minitab programmers never considering screen-reading software, Minitab 15 is surprisingly well on its way to working harmoniously with JAWS.