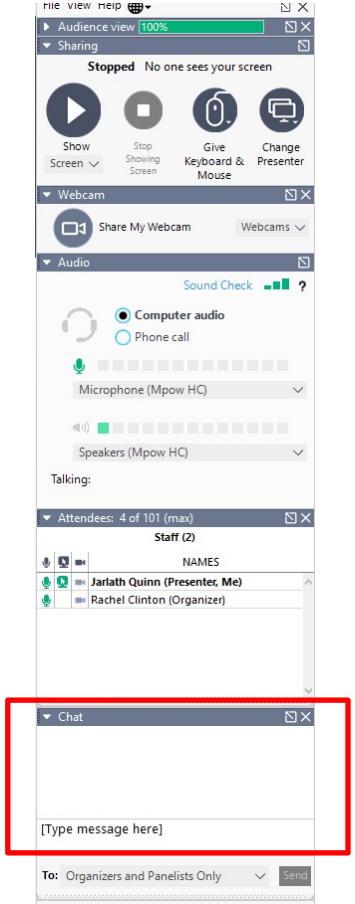

```
*Comment.
 Command
   /Subcommand Keyword = variable
   /Subcommand Keyword = Value.
```

Getting Started with SPSS Syntax

Jarlath Quinn – Analytics Consultant

FAQ's

- Is this session being recorded? Yes
- Can I get a copy of the slides? Yes, we'll email links to download materials after the session has ended.
- Can we arrange a re-run for colleagues? Yes, just ask us.
- How can I ask questions? All lines are muted so please use the chat facility – if we run out of time we will follow up with you.



SMARTVISION Europe

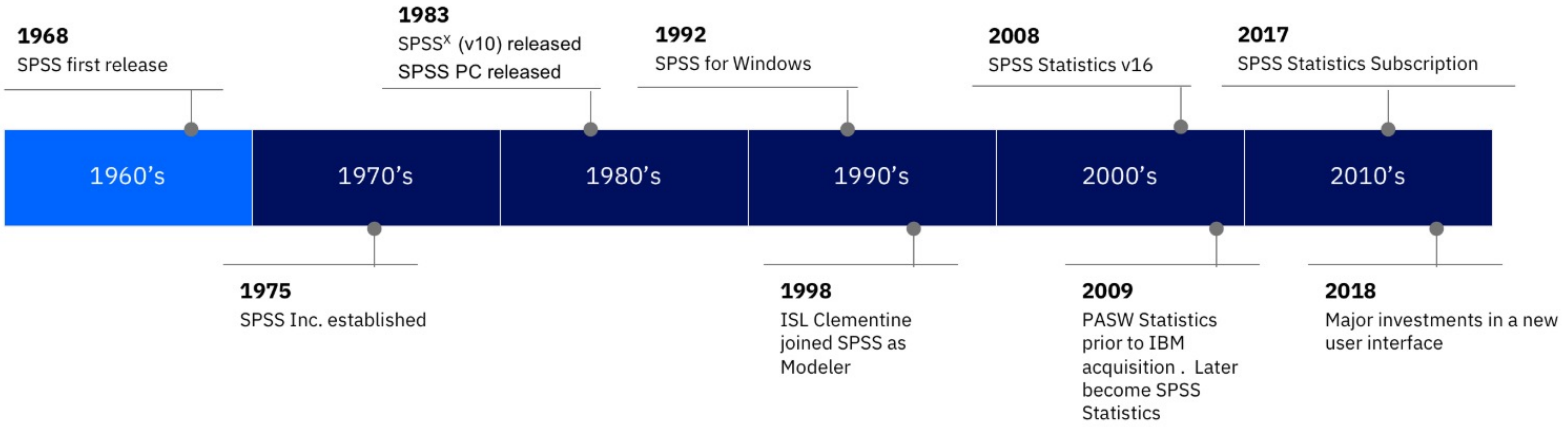
A SELECT INTERNATIONAL COMPANY

- Premier accredited partner to IBM and Predictive Solutions specialising in advanced analytics & big data technologies
- Work with open source technologies (R, Python, Spark etc.)
- Team each has 15 to 30 years of experience working in the advanced and predictive analytics industry

- Deep experience of applied advanced analytics applications across sectors
  - Retail
  - Gaming
  - Utilities
  - Insurance
  - Telecommunications
  - Media
  - FMCG

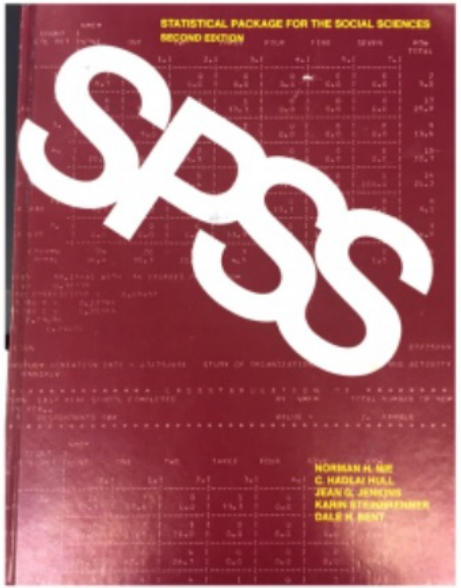
What *is* SPSS syntax

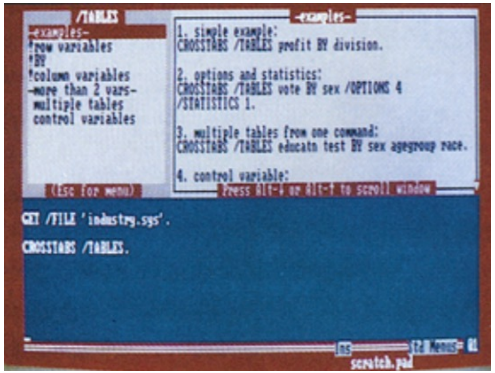
- SPSS Syntax is a command language that is unique to SPSS Statistics.
- Effectively, it's the SPSS operating system.
- The language was developed in the 1960's and so pre-dates Windows/Mac versions of SPSS by some margin



What *is* SPSS syntax

- Before the 1990's users of SPSS relied on Syntax manuals in order to learn how to use the software
- The SPSS syntax language was popular because it was well documented, relatively easy to learn and characterised by 'plain English' commands





SMARTVISION Europe

A SELECT INTERNATIONAL COMPANY

Why use syntax?

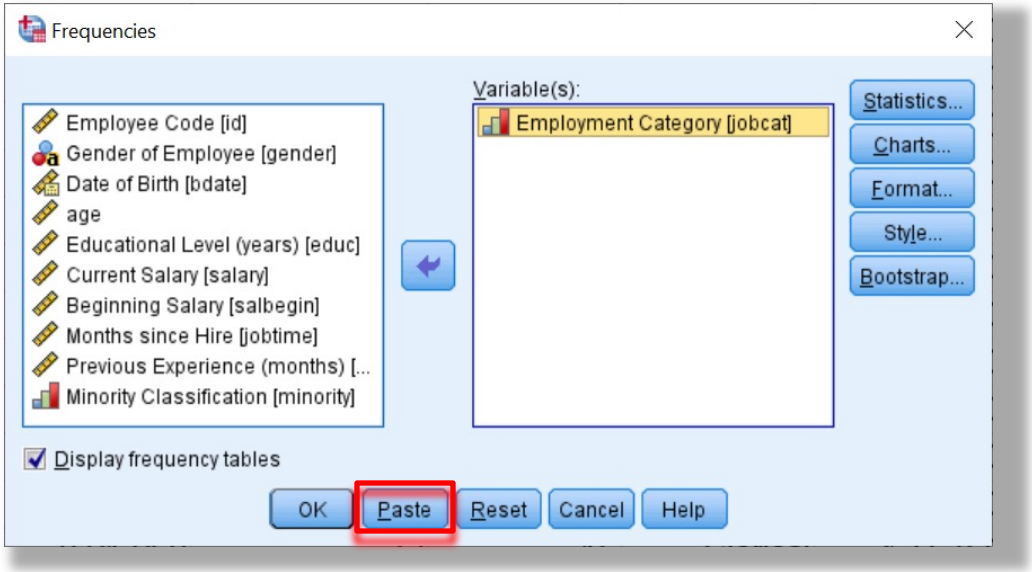
- Automate your entire analytical process
- Complete repetitive tasks more quickly
- Change/fix something to update all your work
- Share your data preparation and analysis with others
- Create a record or audit trail of your work

Why use syntax?

- A huge range of tasks in IBM SPSS Statistics can be completed using SPSS Syntax
- SPSS Syntax is a command language that is much easier to understand than many analytical programming languages such as Python or R

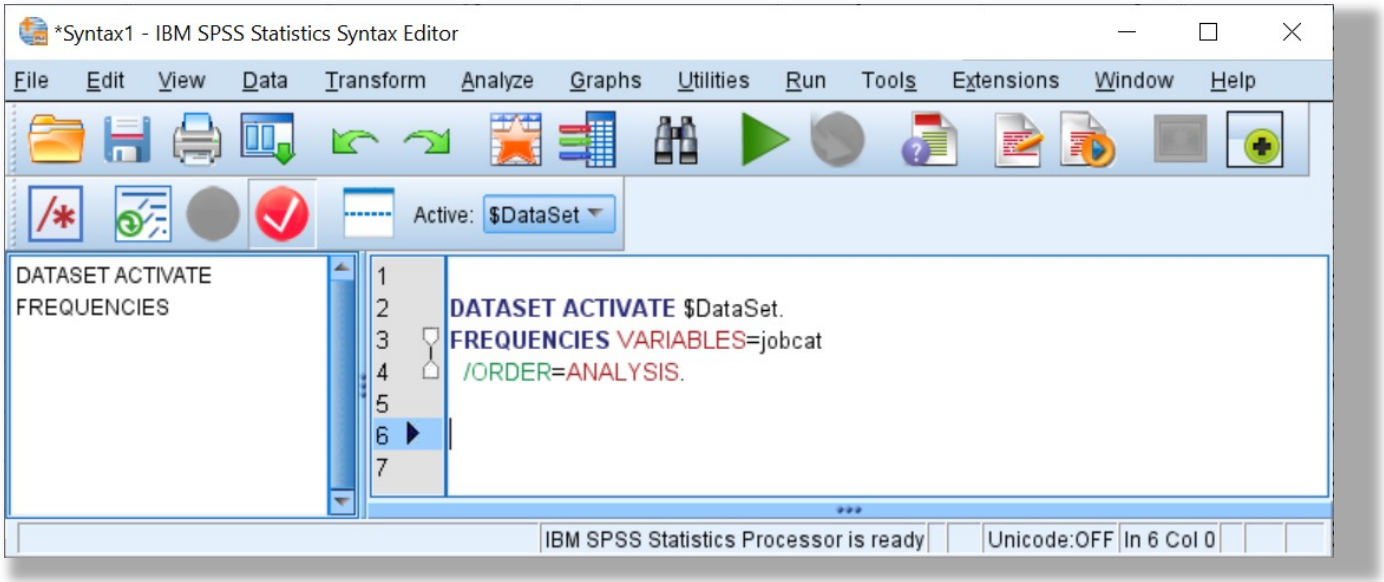
How can I generate SPSS syntax?

- That's easy – most of the SPSS procedures that you access through dialogs have a 'Paste' button.



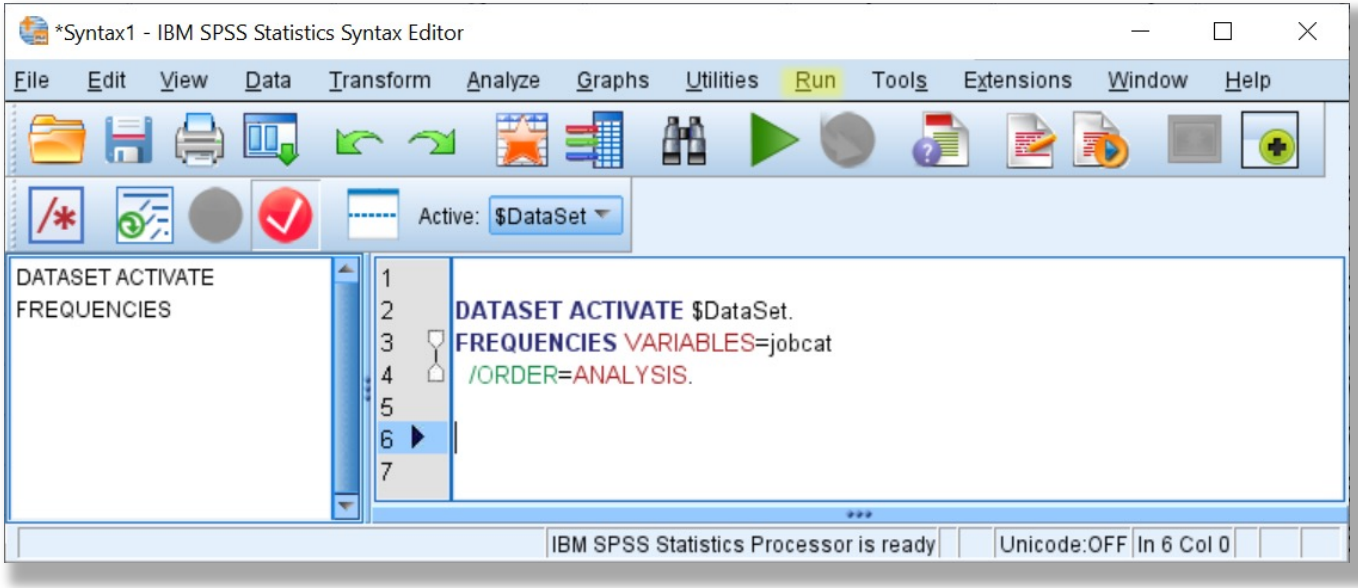
How can I generate SPSS syntax?

- Clicking the ‘Paste’ button means that the SPSS Syntax is pasted to a newly-opened syntax window



How can I execute SPSS Syntax?

- Clicking **Run > All** will execute the procedure and generate a frequency table of the variable ‘Jobcat’



How can I execute SPSS syntax?

- This is the output from the Frequencies procedure

Frequencies

```
[$DataSet] C:\Temp\Employee data with age.sav
```

Statistics

Employment Category

N	Valid	475
	Missing	0

Employment Category

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Clerical	364	76.6	76.6	76.6
	Custodial	27	5.7	5.7	82.3
	Manager	84	17.7	17.7	100.0
	Total	475	100.0	100.0	

Making sense of SPSS syntax

- The pasted syntax in this example consisted of two procedures
  1. A **Dataset Activate** command
  2. A **Frequencies** command

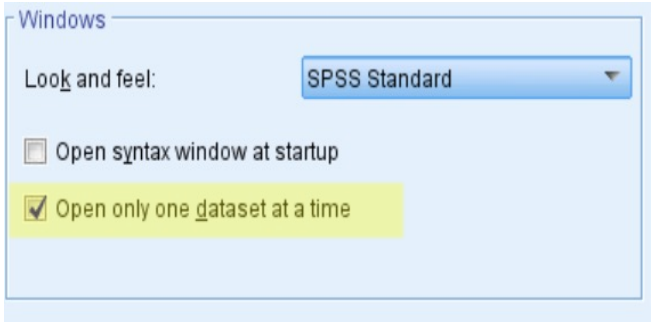
```
1 DATASET ACTIVATE $DataSet.
2 FREQUENCIES VARIABLES=jobcat
    /ORDER=ANALYSIS.
```

The Dataset Activate command

- By default, when SPSS Statistics opens a new data file it labels it as **'$DataSet1'**.
- When it opens a second data file, it labels it as **'$DataSet2'** and continues add a sequential number to each new data file that is opened
- When a command such as **Frequencies** is executed therefore, SPSS needs to know which file it applies to so it adds the **Dataset Activate** command before the next relevant procedure (i.e. Frequencies)
- E.g. **DATASET ACTIVATE $DataSet1**.

The Dataset Activate command

- However, within SPSS Statistics we can request that the system only opens ***one dataset at a time.***
- To do this click: **Edit > Options** navigate to the **General** tab and check the box marked '**Open only one dataset at a time**'. This will mean that each time a dataset is opened the previous one is closed.
- As a result every dataset will simply be labelled "**$DataSet**"
- This is what has happened in our example. Moreover, because only one file can be opened at a time it means the "**DATASET ACTIVATE $DataSet**" command can be ignored.



The Frequencies command

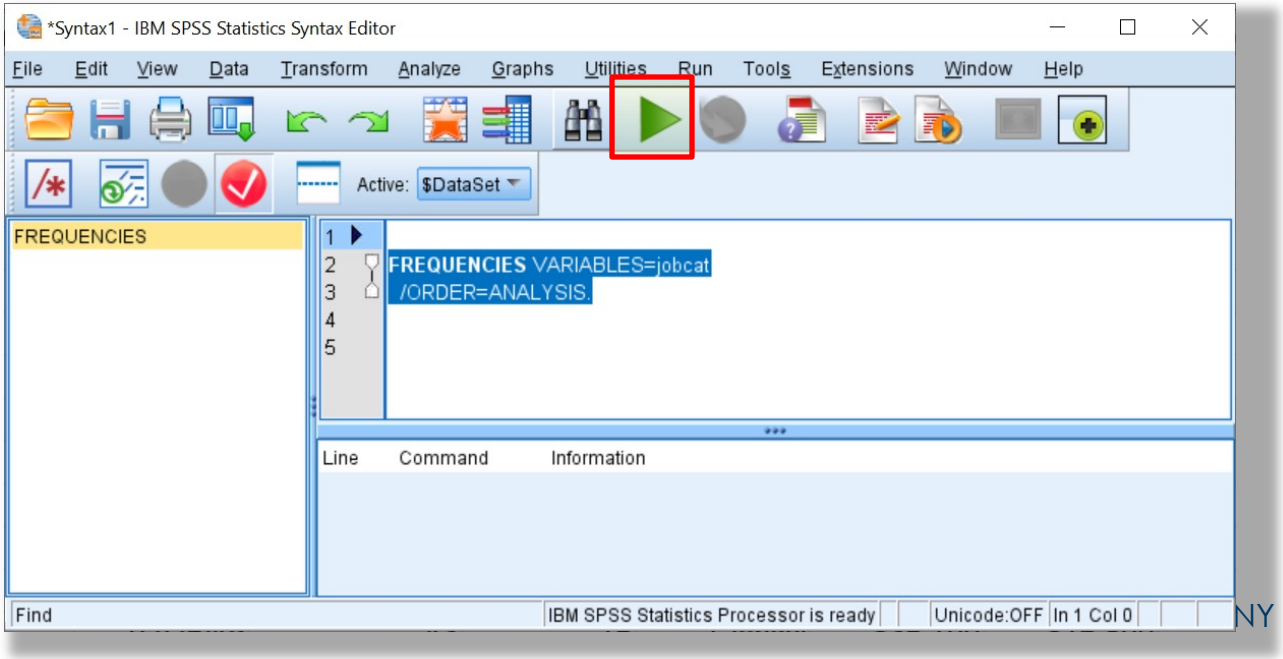
[1]FREQUENCIES [2]VARIABLES=[3]jobcat
[4] /ORDER=ANALYSIS.[5]

The syntax for the Frequencies procedure is actually very simple.

1. As with all SPSS syntax procedures it starts with a single Command: FREQUENCIES
2. This is followed by an optional Keyword: VARIABLES
3. It then specifies a variable: **jobcat**
4. This is followed by a slash with a Subcommand and another Keyword: /ORDER=ANALYSIS
5. Finally the procedure is completed with a single period character: **.**

Executing a specific procedure

- To run a specific procedure in the Syntax window you can highlight the procedure syntax and click: 



SMARTVISION Europe

A few things to note

- The pasted syntax tends to be more verbose than is often required. SPSS will run syntax assuming default settings so sometimes you can actually make the syntax much simpler.
- This is because historically the syntax pre-dates environments like Microsoft Windows, i.e. it was designed to be typed.
- For instance, the following simpler syntax works just as well as the pasted Frequencies syntax:

```
freq jobcat.
```

A few things to note

- SPSS Syntax is remarkably unfussy about how it's typed:
  - It's generally not case sensitive
  - You can use as many lines as you want to specify a single command.
  - It normally doesn't require correct indentation to run.
  - Three or four-letter abbreviations can be used for many command specifications.
- However there are some basic rules:
  - Each new command (e.g. Frequencies) must start on a new line.
  - Each procedure must end with a period as a 'command terminator'.
  - Most subcommands are separated by slashes (/). The slash before the first subcommand on a command is usually optional.

Adding comments in syntax

- Sometimes you may want to add comment in your syntax. Anything between an asterisk and a period (.) is treated as comment and is therefore ignored.
- You can also add comment after a command terminator on the same line by adding a slash (/) in front of the asterisk.

```
*This is example syntax for a Frequencies procedure.
FREQUENCIES VARIABLES=jobcat /ORDER=ANALYSIS. /* Notice the syntax is all on a single line.
```

- Another useful feature of syntax comments is that you can tell the system to ignore a procedure, perhaps as a temporary test, by simply adding an asterisk before the command. Notice that the entire procedure is instantly greyed-out.

```
* FREQUENCIES VARIABLES=jobcat
 /ORDER=ANALYSIS.
```

SPSS Syntax colour codes

- These are the default colour codes that SPSS uses to make syntax more readable*.

Dark Blue	Commands/ Procedure names
Green	Subcommands
Dark Red	Optional Keywords
Black	Variable names, optional text
Orange	Procedure specific values
Grey	Comments (ignored by execution)

SMARTVISION Europe

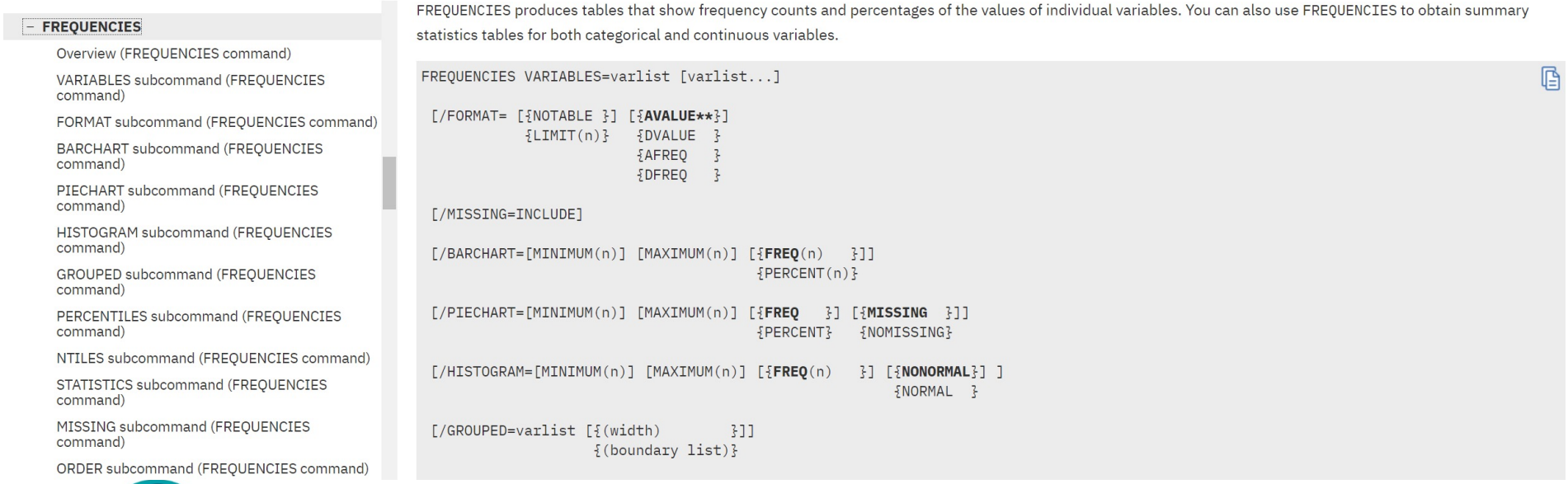
* The default colours can be changed under the Edit > Options menu


```
FREQUENCIES VARIABLES=varlist [varlist...]

 [/FORMAT= [{NOTABLE }] [{AVALUE**}]
           {LIMIT(n)}   {DVALUE  }
                        {AFREQ   }
                        {DFREQ   }

 [/MISSING=INCLUDE]

 [/BARCHART=[MINIMUM(n)] [MAXIMUM(n)] [{FREQ(n)   }]]
                                      {PERCENT(n)}

 [/PIECHART=[MINIMUM(n)] [MAXIMUM(n)] [{FREQ    }] [{MISSING  }]]
                                      {PERCENT}   {NOMISSING}
```

Getting Help with SPSS Syntax

Getting help with syntax

- IBM provides an online [Command Syntax Reference guide](#) that details every SPSS Syntax command.



SMARTVISION Europe

A SELECT INTERNATIONAL COMPANY

Getting help with syntax

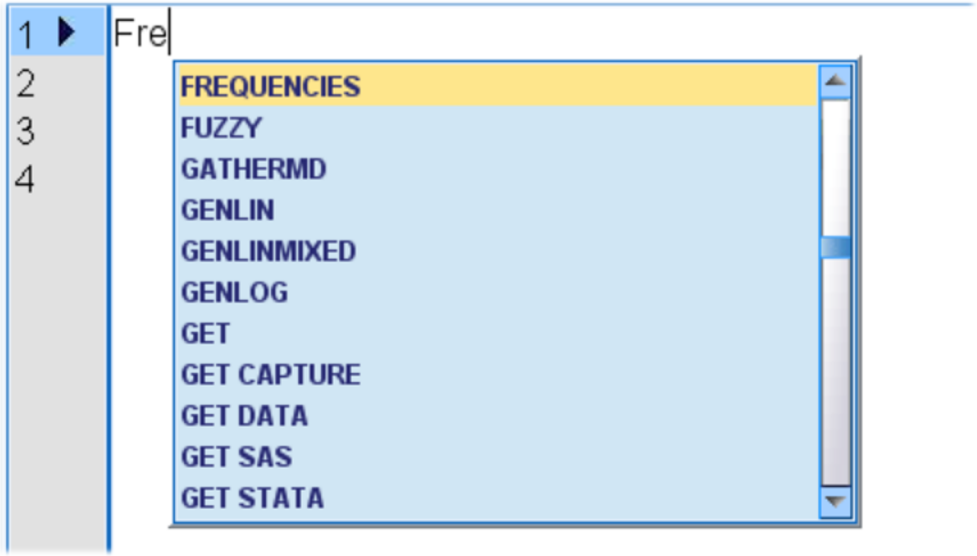
Each command in the Command Syntax Reference guide shows a syntax diagram that maps out all the potential subcommands, keywords, and specifications allowed for that command.

For reference purposes here's an overview of how to interpret the syntax diagrams:

- UPPER CASE indicates keywords to identify commands, subcommands, functions, operators, and other specifications. For example FREQUENCIES is a command and FORMAT is a subcommand.
- Lower case text indicates something the user needs to provide. E.g. the term varlist indicates that you need to enter a list of variables.
- Text in **bold** indicates default settings. There are two types of default: when the default is followed by **, as AVALUE** is in the FREQUENCIES syntax, the default AVALUE (ascending values) is used even if the subcommand (/FORMAT) is not specified. If a default is not followed by **, it is in effect when the subcommand (or keyword) is specified by itself.
- When parentheses, apostrophes, and quotation marks are indicated it means they are required elements.
- Text enclosed in square brackets ([ ]) indicates they are optional elements – unless otherwise specified
- Braces ({ }) indicate a choice between elements. You can specify any one of the elements enclosed within the aligned braces.
- Ellipses indicate that you can have multiple elements in the specification. So FREQUENCIES VARIABLES=varlist [varlist...] just means you can specify many individual variable name

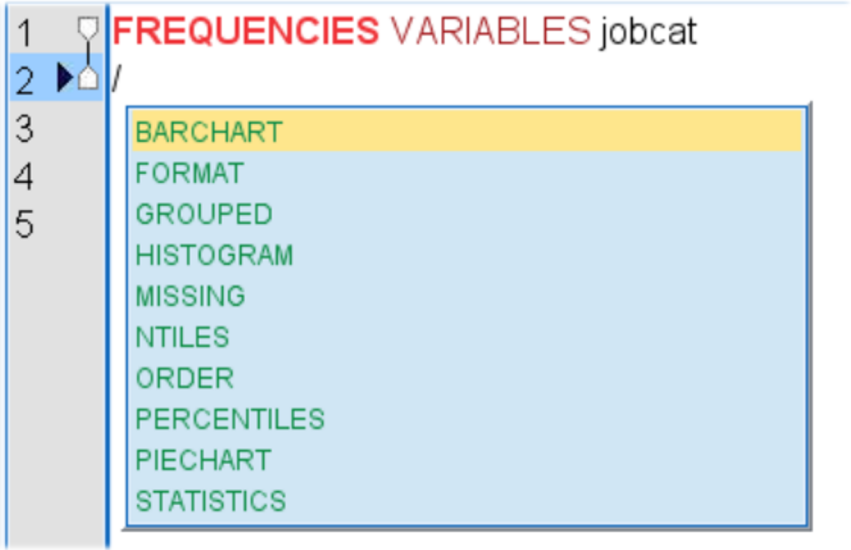
Getting help with syntax

- Notice the useful autocomplete function works when manually typing syntax.



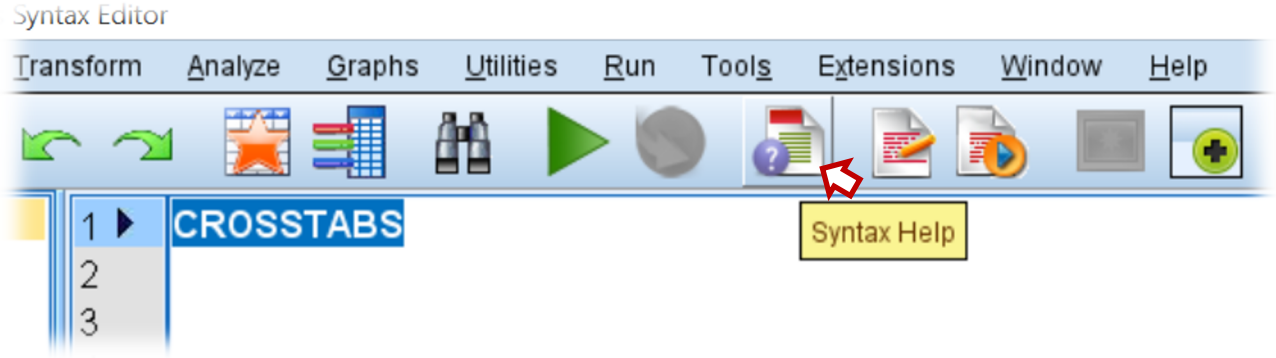
Getting help with syntax

- For instance, if you add a forward slash after a command, a list of subcommands appears.



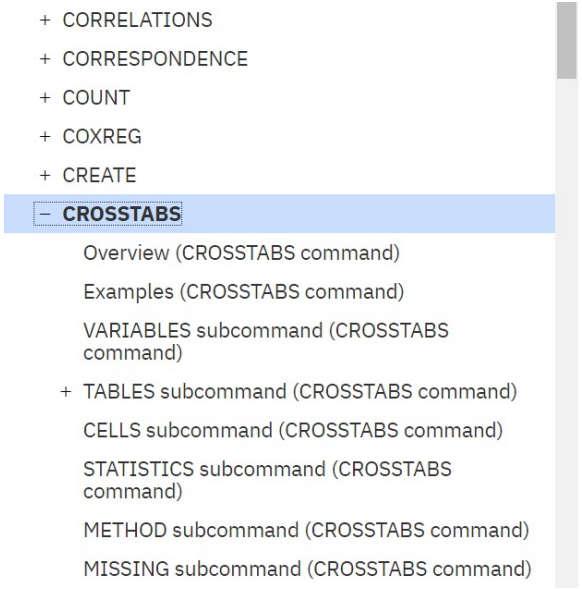
Getting help with syntax

- A really useful feature is the ability to quickly look-up the syntax diagram for a procedure just by highlighting the command word and clicking the Syntax Help button on the toolbar of an open Syntax window.



Getting help with syntax

- SPSS will immediately open the relevant section of the Command Syntax Reference guide so you can see all the additional elements for that procedure




```
CROSSTABS VARIABLES=varlist(min,max) [varlist...]

 /TABLES=varlist BY varlist [BY...] [/varlist...]

 [/MISSING={TABLE**}]
           {INCLUDE}
           {REPORT }

 [/WRITE[={NONE**}]]
          {CELLS }
          {ALL   }
```

Both modes:

```
 [/FORMAT= {AVALUE**}  {TABLES**}]
           {DVALUE  }  {NOTABLES}

 [/COUNT = [{CELL}] [{ROUND }]
            {CASE  }   {TRUNCATE}
            {AGIS  }
```

SMARTVISION Europe

A SELECT INTERNATIONAL COMPANY

SMARTVISION Europe

Examples of SPSS Syntax

Examples of SPSS syntax

- Remember that you can always create syntax simply by clicking the Paste button in an SPSS dialog
- However, in the next few examples we will delve into aspects of syntax that illustrate how different procedures work or reveal functions that aren't obvious to the casual user
- To begin with let's look at opening data in IBM SPSS Statistics

Opening data files

- The GET FILE command is used to import an SPSS data file (*.sav).

```
GET  FILE='C:\Temp\Employee.sav'.
```

- The GET DATA command however is used to import non-native file formats such as Excel workbooks or comma separated (CSV) files.
- Note the use of the /TYPE subcommand that specifies the file type to be imported

```
GET DATA
 /TYPE=XLSX
 /FILE='C:\Temp\Employee.xlsx'.
```

Opening data files

- When reading data, we can easily change the default directory. This is useful when we have to read files on a regular basis from specific sub-folders.
- For example, instead of using:

  GET FILE= “C:\Users\cwjones\Documents\Monthly Reports\january.sav”.

- We can use the CD command to define the default folder

  CD ‘C:\Users\cwjones\Documents\Monthly Reports’.

- And then simply type:

  GET FILE=‘january.sav'.

Labelling data

- If you read non-SPSS data (such as Excel or Text files) you may find you need to apply labelling to the newly imported variables.
- Labels and other ‘dictionary’ information such as missing values have no Paste button in the Variable View window. So knowing the syntax is useful.
- Assigning a Variable Label is pretty easy. Here’s the syntax for attaching/replacing the variable label for the variable **Jobcat.**

```
VARIABLE LABELS jobcat 'Employee Job Category'.
```

Labelling data

- Value Labels work in a similar way. Bear in mind however that using the standard VALUE LABELS command will overwrite any existing value labels.

```
VALUE LABELS jobcat  1'Clerical' 2'Custodial'  3'Manager'.
```

- An alternative is the ADD VALUE LABELS command which is useful when you create a new category.

```
ADD VALUE LABELS jobcat  4 ‘VP’.
```

Recoding data

- Many users actually find that Recoding categories using syntax is *easier and more intuitive* than using the dialogs for “Recode into Same Variables” or “Recode into Different Variables”
- Here we use RECODE to create a new variable called jobcat2

RECODE jobcat[1] (1 thru 2[2] = 1) (3 thru HI[3] = 2) into jobcat2[4].

1. We specify the name of the variable we want to recode
2. We state that categories 1 and 2 will be recoded into a value of 1.
3. We state that the value 3 or higher will be recoded into a value of 2.
4. We specify the name of the new variable that the recode will create

Recoding data

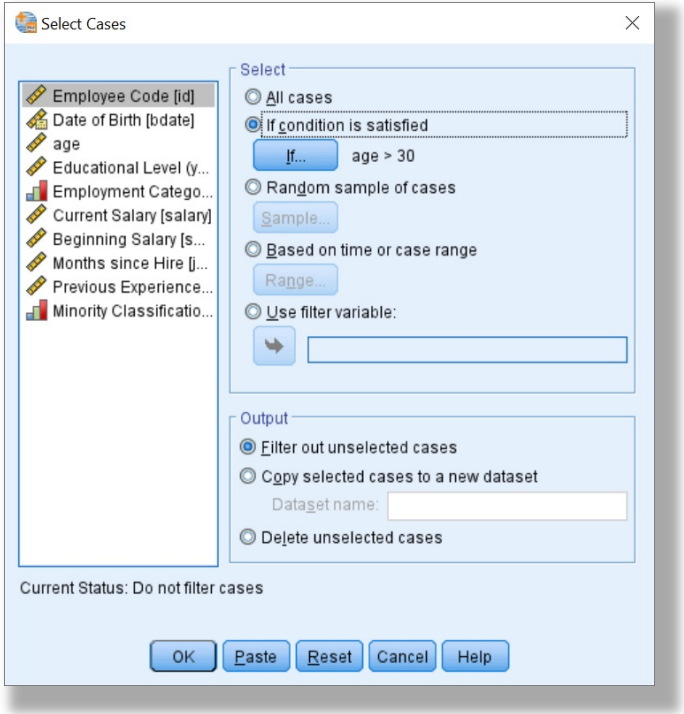
- We need to add some labels to the new variable so the full syntax looks like this:

```
RECODE jobcat  (1 thru 2 = 1) ( 3 thru HI = 2)  into Jobcat2.

VARIABLE LABELS jobcat2 'Management Status'.

VALUE LABELS jobcat2  1'Non-Manager' 2 'Manager'.
```

Filtering data with the Select Cases dialog

- Using the paste button in the Select Cases dialog tends to generate surprisingly complex syntax
- Here is the pasted syntax for a filter where employees older than 30 are selected

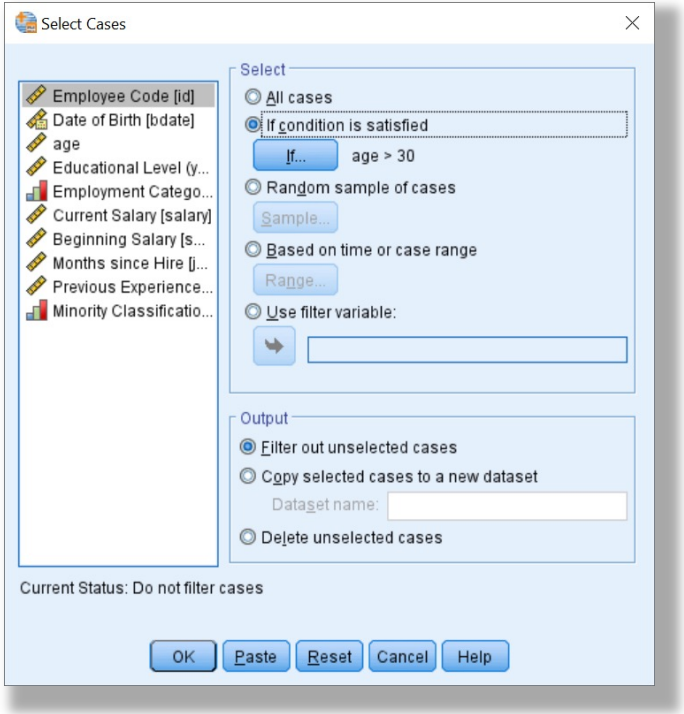
```
DATASET ACTIVATE $DataSet.
USE ALL.
COMPUTE filter_$=(age > 30).
VARIABLE LABELS filter_$ 'age > 30 (FILTER)'.
VALUE LABELS filter_$ 0 'Not Selected' 1 'Selected'.
FORMATS filter_$ (f1.0).
FILTER BY filter_$.
EXECUTE.
```



Filtering data with the Select Cases dialog

- One of the reasons the pasted syntax is so verbose is that the dialog creates a filter variable (filter_$) that marks which cases were included or dropped from the selection.

```
DATASET ACTIVATE $DataSet.
USE ALL.
COMPUTE filter_$=(age > 30).
VARIABLE LABELS filter_$ 'age > 30 (FILTER)'.
VALUE LABELS filter_$ 0 'Not Selected' 1 'Selected'.
FORMATS filter_$ (f1.0).
FILTER BY filter_$.
EXECUTE.
```



Filtering data with FILTER

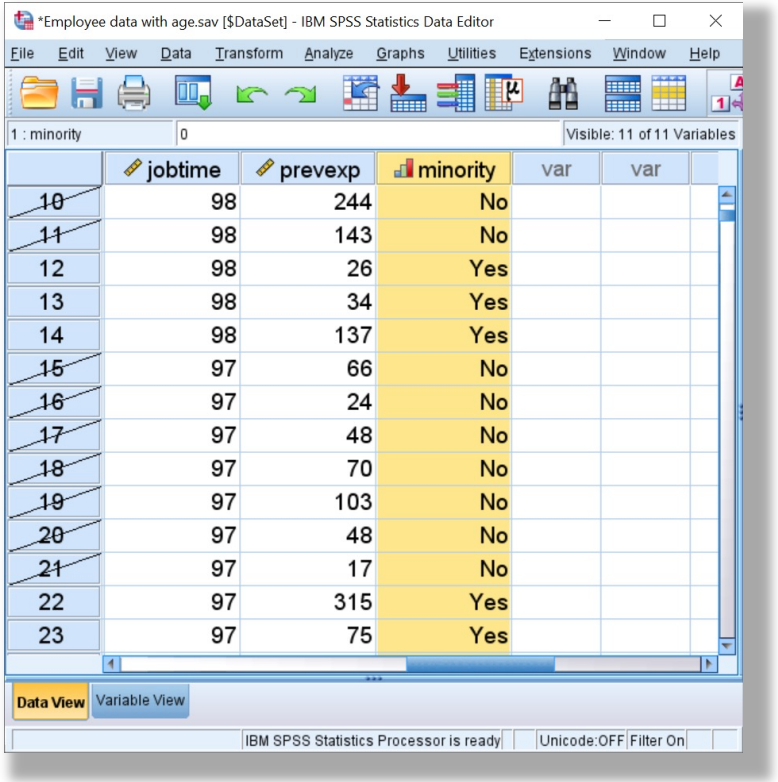
- Filtering data with syntax is usually less complex.
- Using the FILTER command we can filter out any cases that are equal to zero or have missing values.
- The variable minority has two values: 0 ‘No’ 1’Yes’
- So the following syntax will select all cases that are in the ‘Yes’ group.

```
FILTER by minority.
```

Filtering data with FILTER

- Note that all the cases labelled ‘No’ in the final column (minority) are filtered out (as indicated by the diagonal stroke through the case header).
- To switch off the filter, simply type:

FILTER OFF.



Filtering data with FILTER

- We can use commands like COMPUTE or RECODE to recreate what the Select Cases dialog does.
- Simply create a new field that marks any cases you don't want to include in your analysis with a zero or a missing value and use the FILTER command to select everyone else. For Example.

```
*This compute command simply creates a variable with a 1 or a 0.
*Managers over 40 are given the value 1.
*Everyone else is given the value 0.
COMPUTE  Older_Managers = jobcat = 3 and age > 40.
*The FILTER command then selects every case with a value of 1.
FILTER by Older_Managers.
```

Filtering data with SELECT IF

- The SELECT IF command permanently selects cases for analysis based on a logical expression.
- Once the SELECT IF command is executed, users may notice that the data file seems to be unaffected and that the message 'Transformations Pending' appears on the task bar of the Data Editor window.

- The SELECT IF command only really takes effect when another procedure (like Frequencies) is run after the command is executed.

```
SELECT IF (age GT 30).
```

Filtering data with SELECT IF

- A simple way to get around SELECT IF permanently removing the data is to use the TEMPORARY command beforehand.
- This is useful if you just want to filter out unwanted cases for the next immediate procedure. So if we type:

```
TEMPORARY.
SELECT IF (age GT 30).
Freq Jobcat. /*Temporary selection takes effect.
Freq Jobcat. /*Temporary selection is switched off.
```

- We will see two frequency tables for the variable Jobcat. The first only shows results for employees older than 30 and the second shows the results for everyone.

Other Useful Syntax Procedures

Applying syntax to multiple variables

- We can use keywords TO and ALL to apply syntax to multiple variables.
- In this VALUE LABELS example, the labels are applied to seven variables that share the same response options.
- These variables appear sequentially in the data file so the TO keyword applies the labels to variable each in turn.

```
VALUE LABELS rate1 TO rate7
0 "Don't Know"
1 'Excellent'
2 'Good'
3 'Poor'
4 'Very Poor'.
```

SMARTVISION Europe

* You may notice that the value label “Don’t Know” is defined by double quotes as it already contains an apostrophe which would be interpreted as a quotation mark.

Applying syntax to multiple variables

- We can also use the keyword ALL in analytical procedures

```
FREQUENCIES rate1 TO rate7.
```

- Alternatively, the ALL keyword applies to all the fields in a file.

```
FREQUENCIES ALL.
```

Working with Scratch variables

- Scratch variables are temporary variables that can be used as part of a calculation process.
- The idea is that the scratch variables are used as an intermediary step and the user doesn't want to keep them as part of the data file. All scratch variables are prefixed with a **#** symbol
- The following could be completed in a single step but is used only as a simple example.

```
COMPUTE #yearsemployed =  jobtime/12.
COMPUTE Age_Start = TRUNC (age -  #yearsemployed).
EXECUTE.
```

Working with Scratch variables

- The variable **jobtime** records *months employed* by the organisation. So the scratch variable **#yearsemployed** is simply **jobtime** divided by 12.
- The second Compute statement simply subtracts **#yearsemployed** from the employee's current **age** to estimate the age they started working for the organisation (**Age_Start**).
- The keyword **TRUNC** truncates the result to remove decimal values.
- The scratch variable **#yearsemployed** is not retained in the data file

```
COMPUTE #yearsemployed =  jobtime/12.
COMPUTE Age_Start = TRUNC (age -  #yearsemployed).
EXECUTE.
```

Delete variables

- The DELETE VARIABLES command can used on an individual basis or apply to a list of variables incorporating the TO keyword.

```
DELETE VARIABLES  bdate prevexp.
```

Renaming variables

- RENAME VARIABLES simply changes the variable names.
- It can be applied to individual variables or on a group basis.
- Can be used as a subcommand with the GET FILE and SAVE OUTFILE commands

```
RENAME VARIABLES (jobcat = jobtype).

RENAME VARIABLES ( salary salbegin minority = Current_Pay  Starting_Pay  BAME_Group).

SAVE OUTFILE='C:\Temp\Employee data.sav' /RENAME (gender = sex).
```

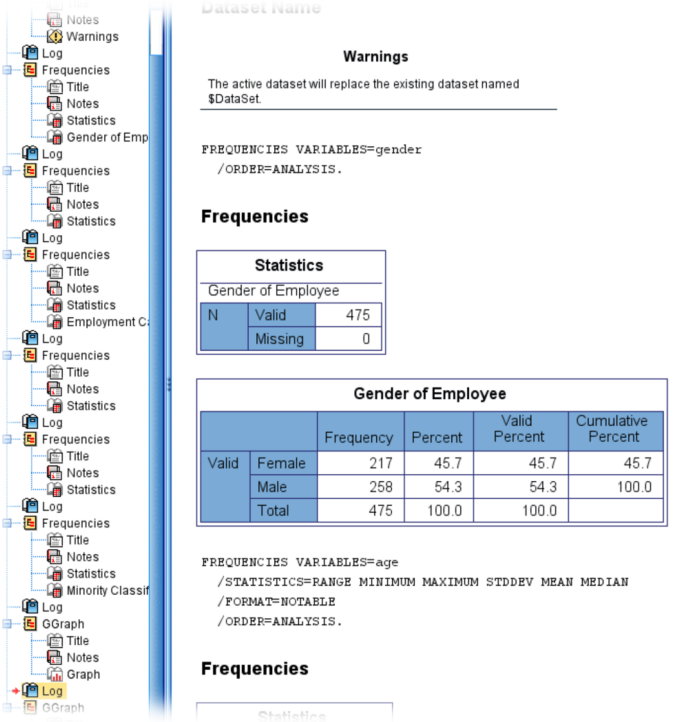
/KEEP /DROP subcommands

- The /KEEP and /DROP subcommands can be used with reading and saving datasets.
- /KEEP and /DROP respectively allow users to either specify the subset of variables that they wish to retain in a file, or the variables they *don't* want to include

```
*Only the variables id salary and salbegin will be loaded.
GET  FILE='C:\Temp\Employee data with age.sav' /KEEP id salary salbegin.

*All variables except salary and salbegin will be saved.
SAVE OUTFILE='C:\Temp\Employee no salary.sav'  /DROP salary salbegin.
```

The OUTPUT MODIFY command

- OUTPUT MODIFY is a powerful and functionally rich procedure that can apply formatting and other changes to the contents of the active Viewer window.
- It is useful for automatically deleting unwanted objects such as warnings or syntax logs
- It can also be used to highlight specific types of values such as totals in tables



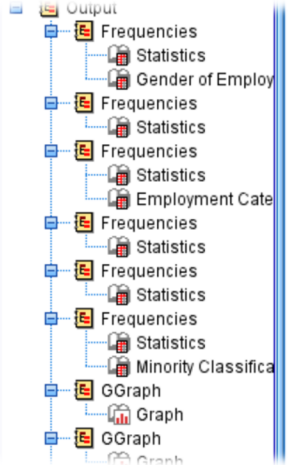
SMARTVISION Europe

A SELECT INTERNATIONAL COMPANY

The OUTPUT MODIFY command

- The following syntax deletes all preceding output except for tables and charts

```
OUTPUT MODIFY
  /SELECT ALL EXCEPT (TABLES CHARTS)
  /DELETEOBJECT DELETE = YES.
```



Median	$28,875.00
Std. Deviation	$17075.661
Range	$119,250
Minimum	$15,750
Maximum	$135,000

Statistics

Beginning Salary

N	Valid	475
	Missing	0
Mean		$17,002.47
Median		$15,000.00

The OUTPUT MODIFY command

- The following syntax finds Frequency tables in the output viewer and emboldens the Total values in each table

```
OUTPUT MODIFY
  /SELECT TABLES
  /IF SUBTYPES=["Frequencies"]
  /TABLECELLS SELECT=["Total"]
    SELECTDIMENSION=ROWS
    STYLE=BOLD APPLYTO=ROW.
```

Employment Category

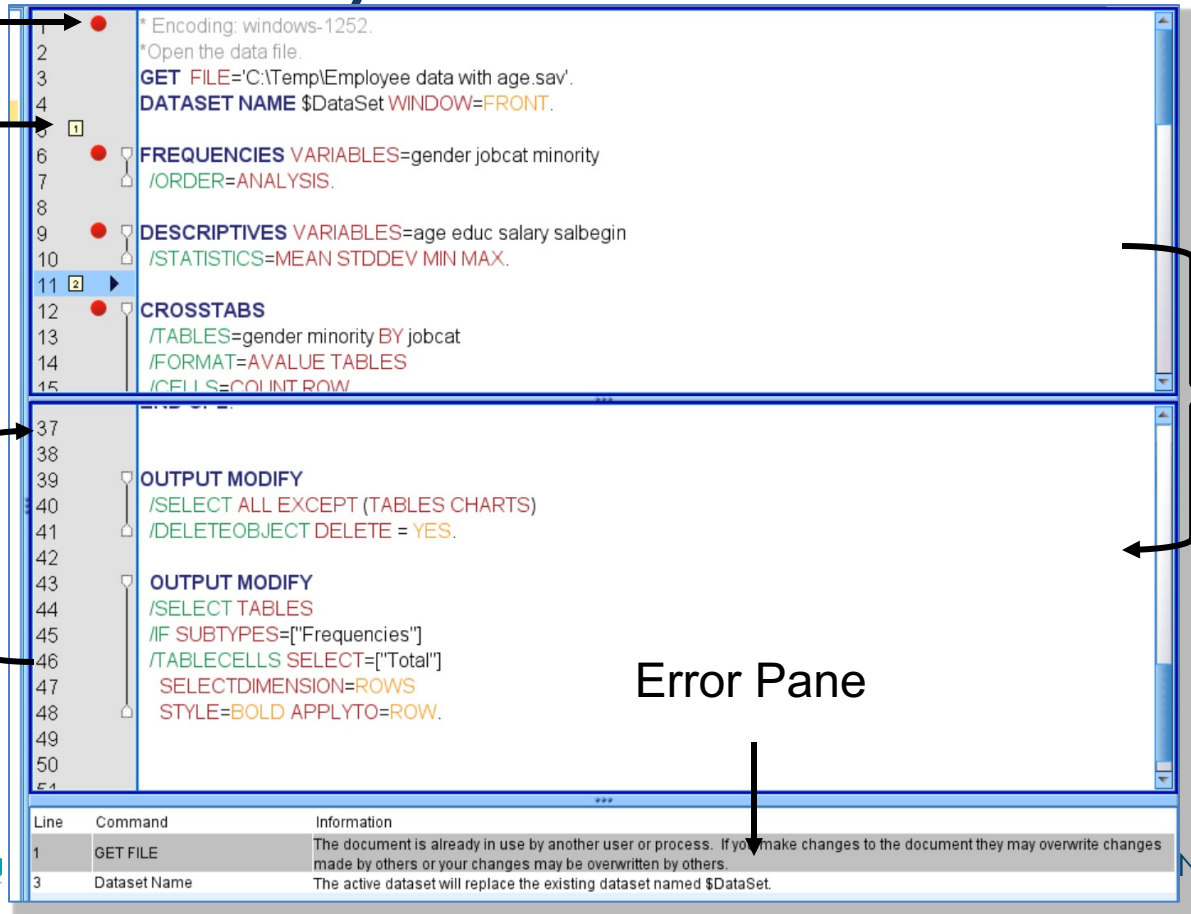
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Clerical	364	76.6	76.6	76.6
	Custodial	27	5.7	5.7	82.3
	Manager	84	17.7	17.7	100.0
	Total	**475**	**100.0**	**100.0**	

Useful Syntax Window Functions

Useful Syntax Window functions

- **Breakpoints:** Breakpoints are used to stop command syntax execution at pre-specified positions so you can check the results at each stage before moving on.
- **Bookmarks:** Creating bookmarks allow you to more easily navigate large command syntax files.
- **Auto-Indentation:** You can automatically format your syntax for improved readability using an indentation style like the syntax pasted from a dialog.
- **Step Through:** You can step through command syntax one command at a time, advancing to the next command with a single click.
- **Split view:** From the main menu click Window > Split View. This splits the syntax editor window into two panes so that you can check one part of the syntax file without losing focus on another.

Useful Syntax Window functions



SMARTVISION Europe

NATIONAL COMPANY

Syntax Keyboard Shortcuts

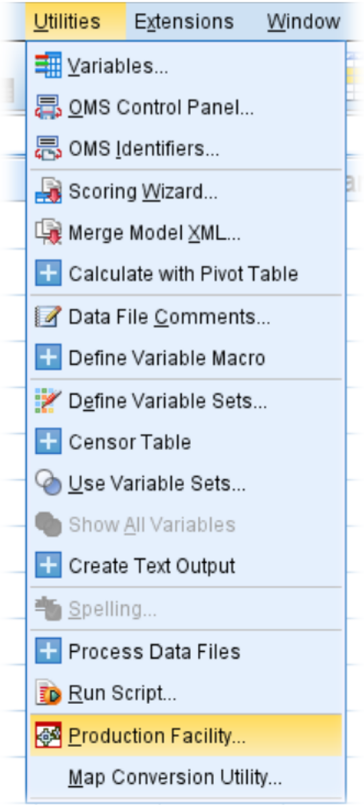
CTRL+E	Go to next error
CTRL+J	Join lines(remove line breaks, creating single line)
CTRL+U	Duplicate current line, or selection
CTRL+D	Delete current line, or selected lines
CTRL+M	Remove any empty lines within selection
CTRL+G	Activate column editing mode on selected lines
ESCAPE	Deactivate column editing mode
ALT+UP	Move selected text up
ALT+DOWN	Move selected text down
CTRL+L	Trim leading spaces
CTRL+T	Trim trailing spaces
CTRL+B	Trim leading and trailing spaces

SMARTVISION
Europe

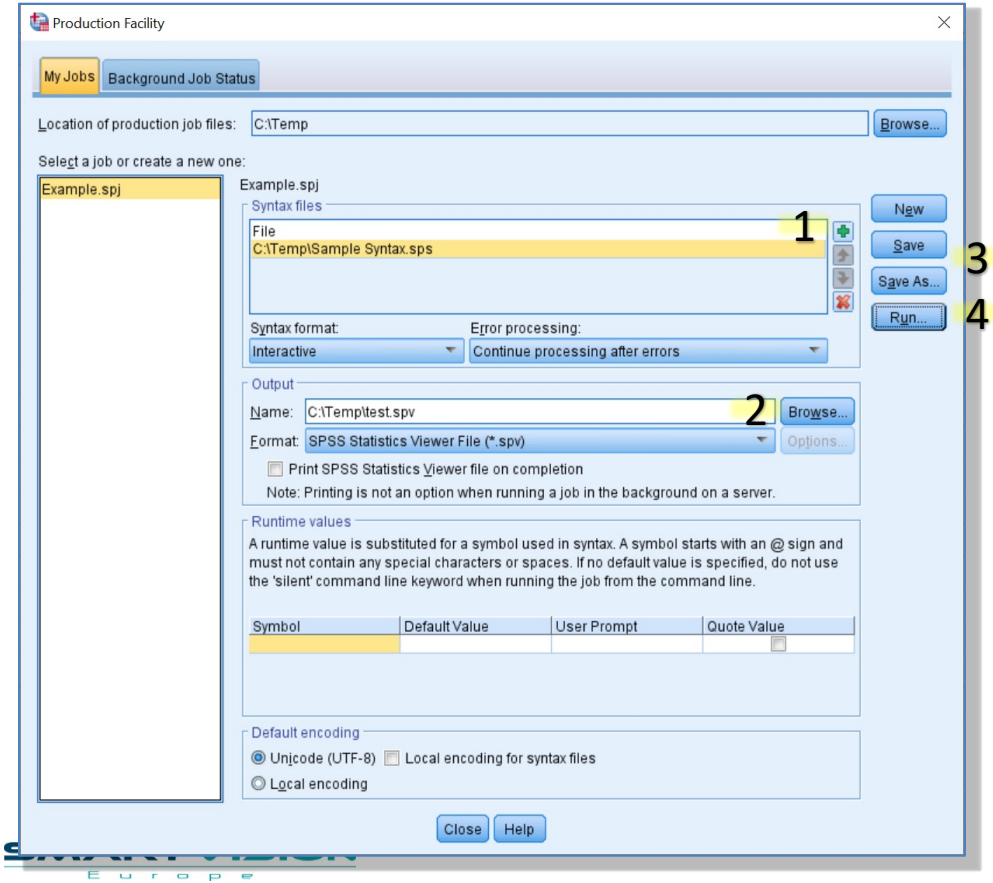
Using the Production Facility

Using the Production Facility

- The Production Facility within SPSS Statistics is useful for automating syntax. It allows users to run one or more syntax jobs and export the output in a particular format (e.g. *.spv, *.pdf, *.doc etc) to a specified location
- The facility has two modes:
  1. Interactively. The program runs unattended in a separate session on either your local computer or a remote server. Your local computer must remain on (and connected to the remote server, if applicable) until the job is complete.
  2. In the background on a server. The program runs in a separate session on a remote server. Your local computer does not have to remain on and does not have to remain connected to the remote server. You can disconnect and retrieve the results later.
- To access the facility click: **Utilities > Production Facility**



Using the Production Facility dialog



1. Click the plus button  to add one or more syntax files (*.sps)
2. Specify an output format, filename and location
3. Specify a name and location for the production job file (*.spj)
4. Run the job

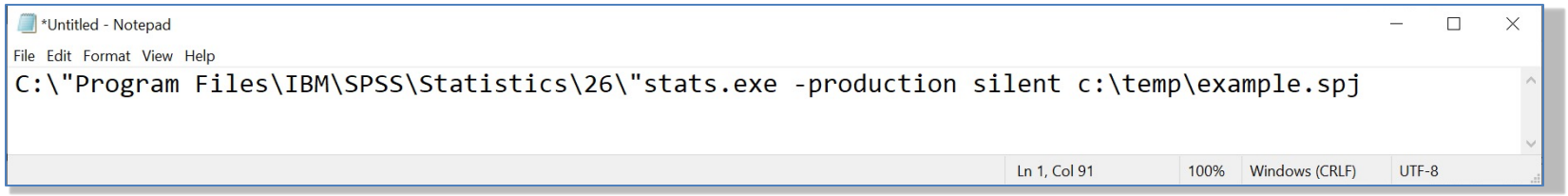
Automating SPSS execution

- Both SPSS syntax files (*.sps) and Production files (*.spj) can be run from command lines with various options
- A popular method is use *.bat files. These are DOS batch files written in plain text that execute commands with the Windows prompt.
- To create a .BAT file to execute an SPSS production job, all we need to do is open Notepad.exe and type:
  1. The address of the SPSS executable
  2. The execution mode i.e. Production Silent *
  3. The name and location of the production job file

*The keyword ‘silent’ means you don’t see the production facility dialog

Automating SPSS execution

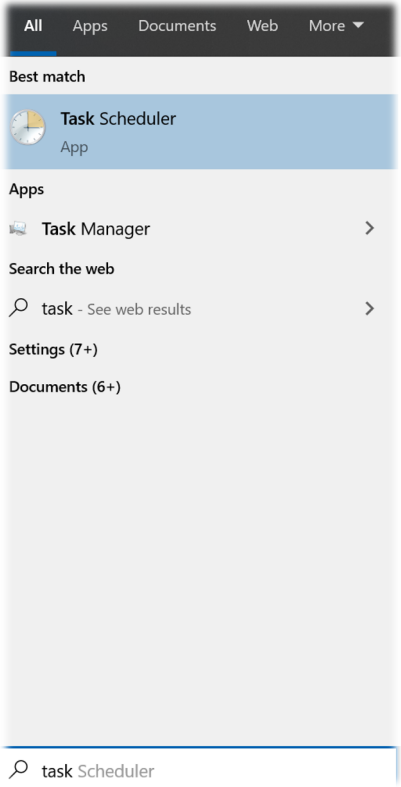
- As an example, we can type the following command into Notepad…



- …and save it as “SPSS batch example.bat”

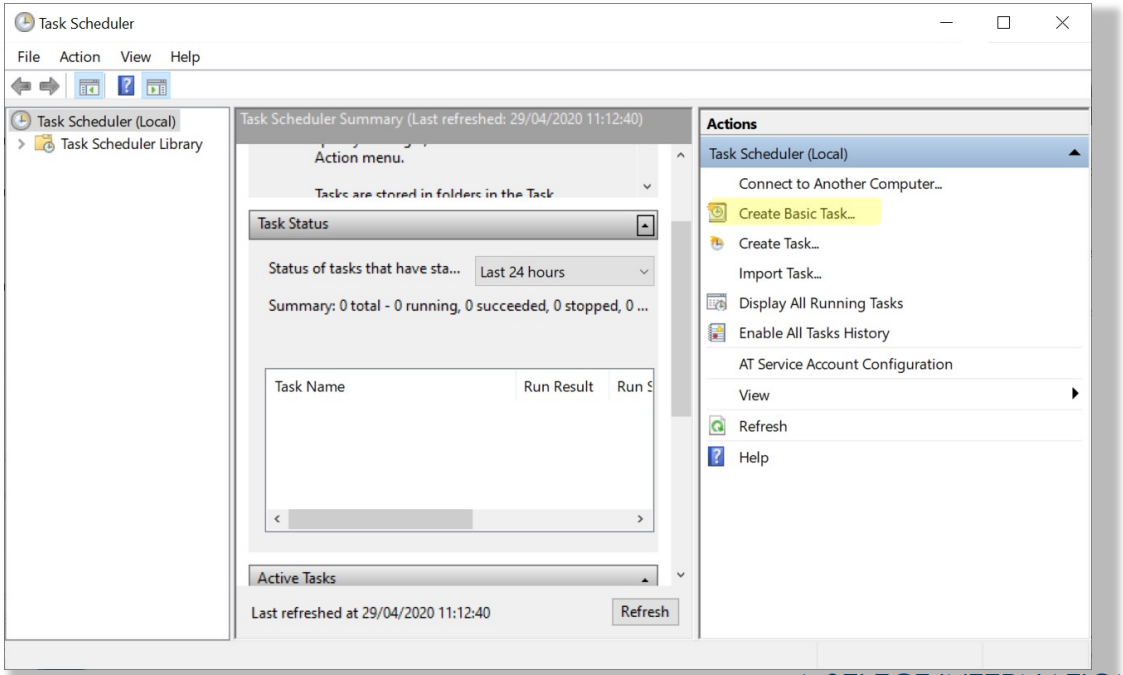
Automating SPSS execution

- To execute the .bat file we can simply double-click it
- Alternatively we can use the Windows in-built Task Scheduler to execute it on a timed or scheduled basis
- To find this program type ‘Task Scheduler’ in the Search box within Windows



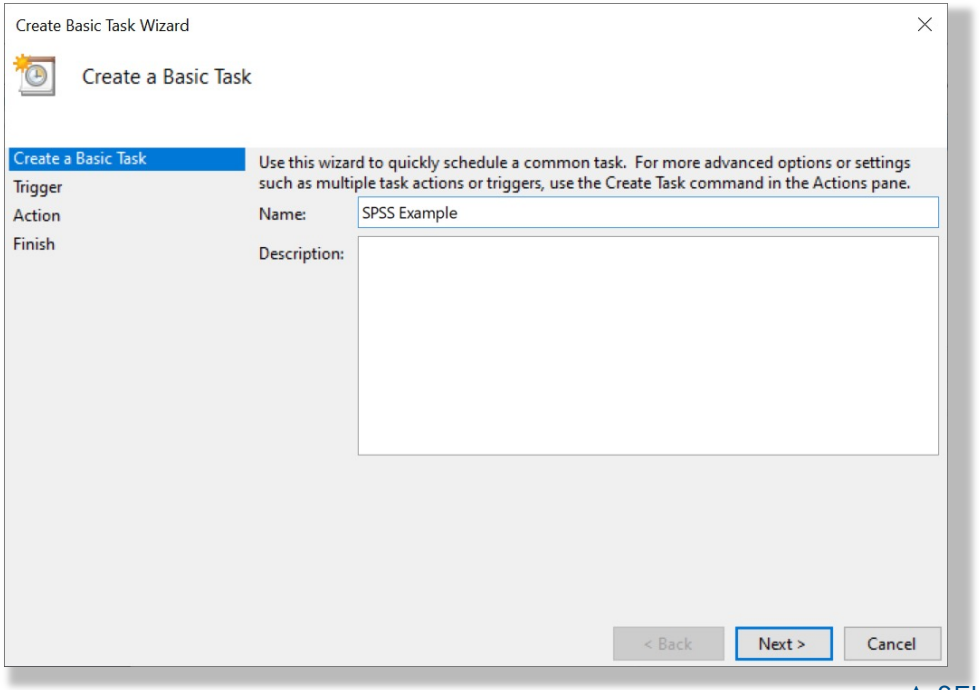
Automating SPSS execution

- When Task Scheduler is launched, click Create Basic Task to get started



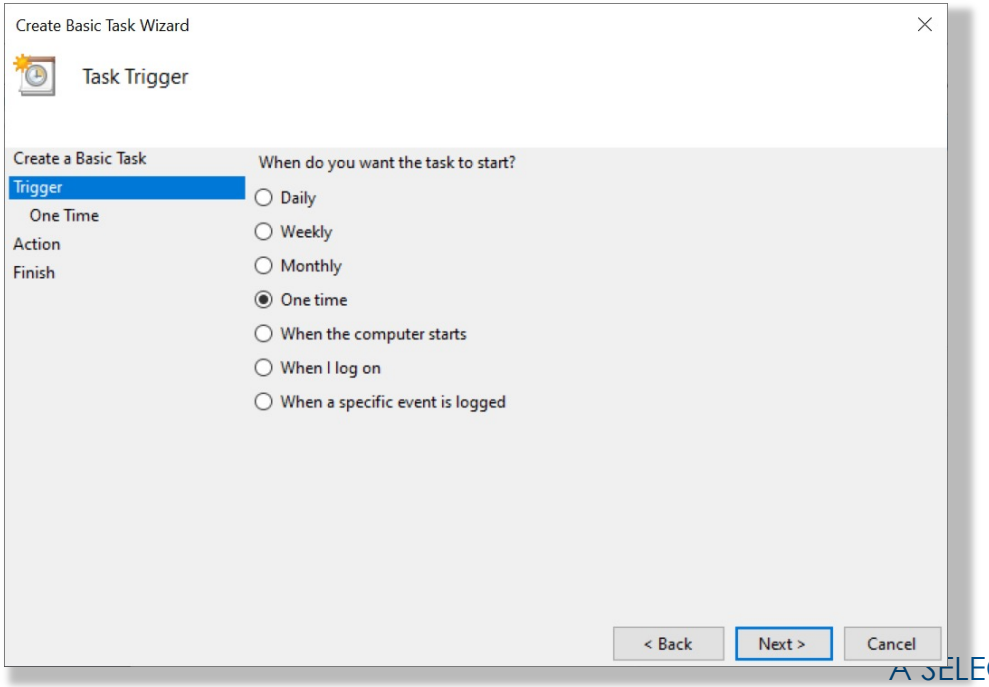
Automating SPSS execution

- Define a name for the task…



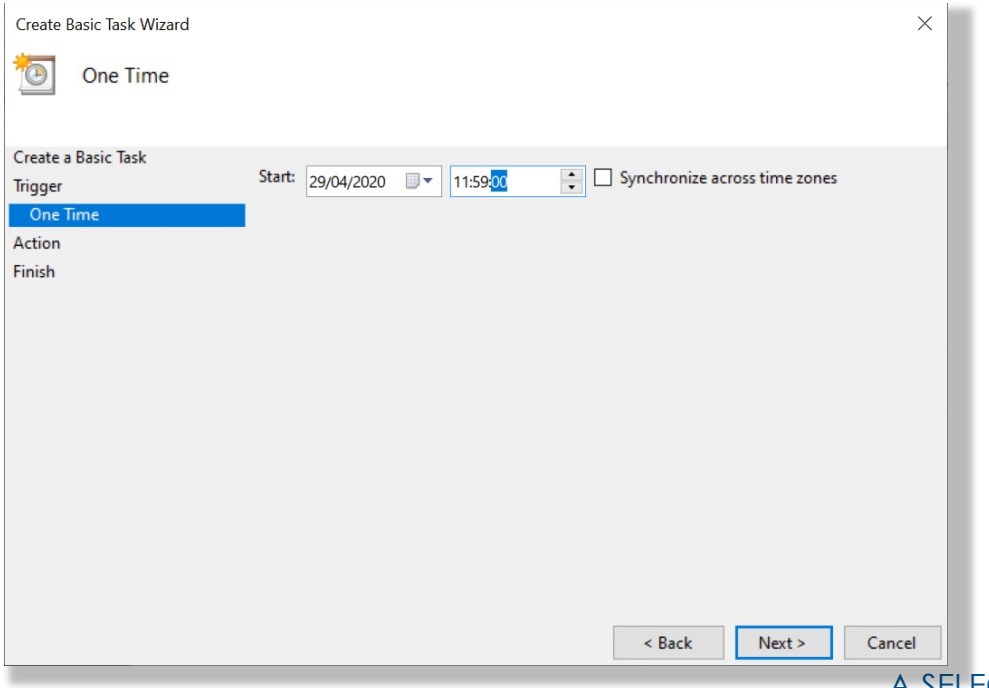
Automating SPSS execution

- Specify it as a one time task…



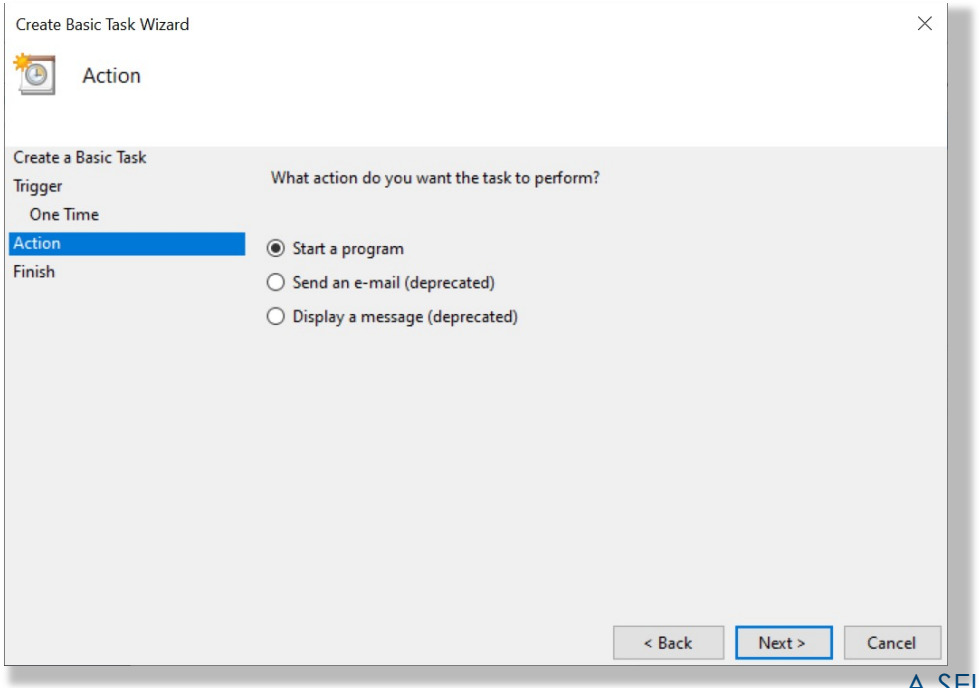
Automating SPSS execution

- Specify a time for it to execute…



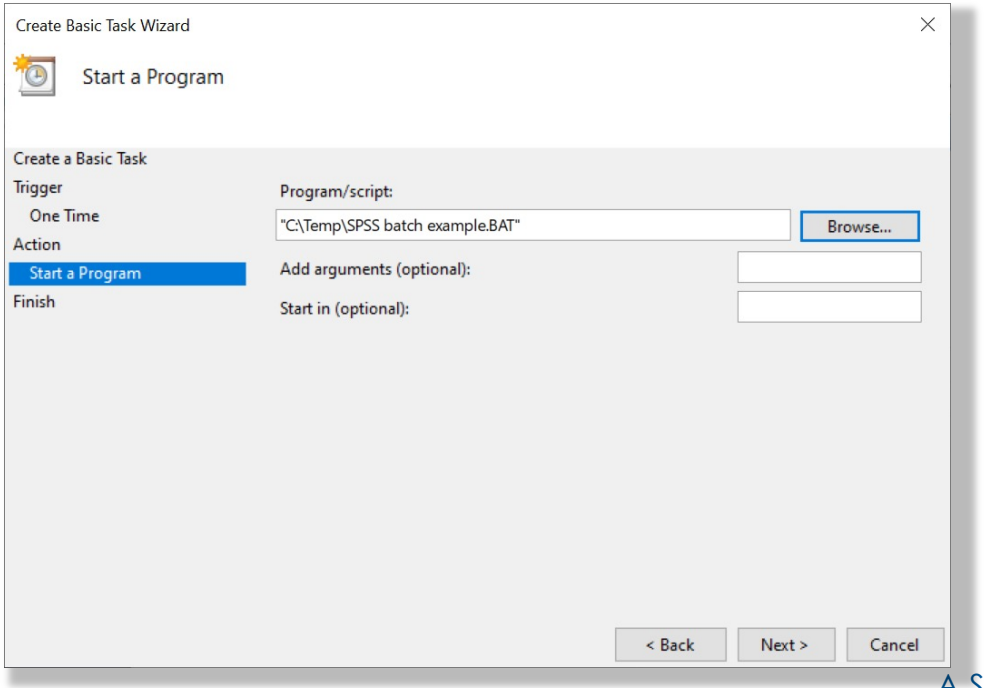
Automating SPSS execution

- Specify the action (i.e. Start a program)



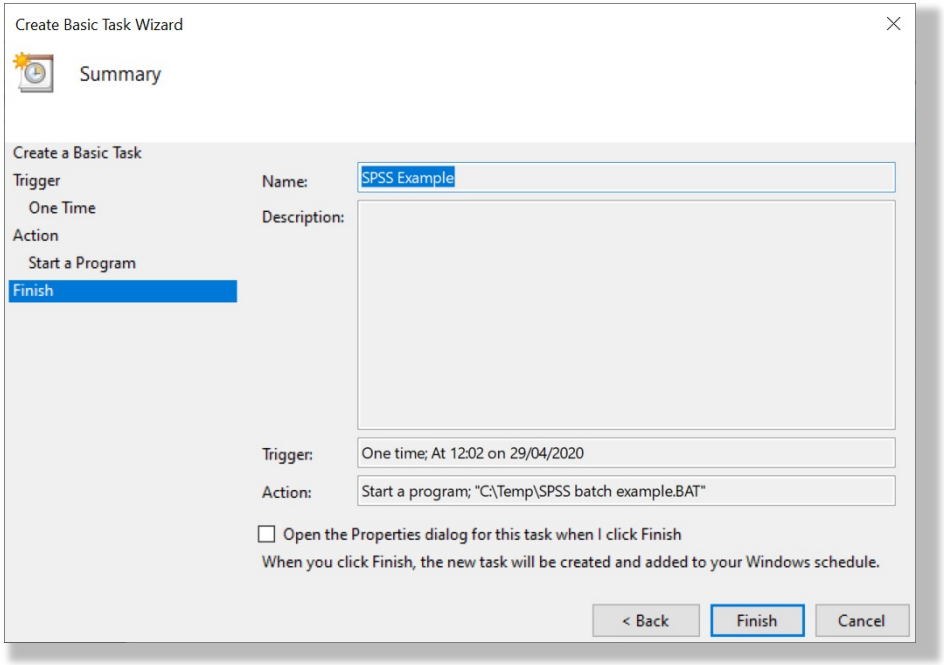
Automating SPSS execution

- Specify the newly created *.BAT file…



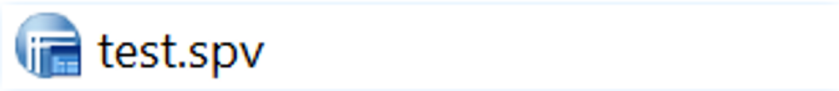
Automating SPSS execution

- And finish.



Automating SPSS execution

- At the specified execution time the Windows DOS command window is briefly opened along with the SPSS Statistics logo
- You should find that output file has been generated and saved to your chosen folder



SMARTVISION Europe

A SELECT INTERNATIONAL COMPANY

Special offer for webinar attendees

- 50% off all our online self-paced training courses before 31st March
- Range of courses includes:
  - Full online version of the 2 day "Introduction to SPSS Statistics" course
  - Factor & Cluster Analysis
  - Linear and Logistic Regression
  - Time series forecasting
  - Significance testing
  - Decision Trees
- Your access to the courses will not expire – no deadline on when you complete them
- Quote code **50%sig** when you checkout
- Check out the full list of courses here

SMARTVISION Europe

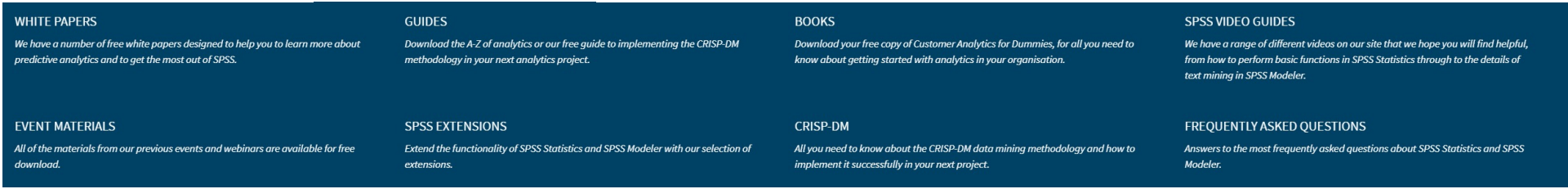
A SELECT INTERNATIONAL COMPANY

Additional Resources

- Video Guides 5 part walk through on using syntax to refer back to
- SPSS FAQs everything from finding out what you have installed to how to merge files or change the language
- White Papers guides for working with SPSS and R together and writing SPSS extensions
- SPSS Software information on products, modules and pricing
- Eat your greens blog series on statistical testing and procedures

Useful Resources

- Raynald's SPSS Tools
- SPSS Tutorials
- ASSESS SPSS User Group
- Smart Vision Europe's resource page



Contact us:

+44 (0)207 786 3568
info@sv-europe.com
Twitter: @sveurope
Follow us on Linked In
Sign up for our Newsletter

Thank you