Data Management using SPSS

Course instructors: Diarmuid McDonnell and Pauline Ward
(diarmuid.mcdonnell@ed.ac.uk and pauline.ward@ed.ac.uk)
University of Edinburgh. Data Library
March 2017

Course Outline

The objective of this workshop is to introduce you to some techniques for using SPSS to support your research data management (RDM) activities during the course of your project(s). It focuses on the principles and practices of good data management, and how these can be implemented in SPSS. The workshop outlines how to process and transform your data, including the crucial step of documenting your RDM activities, in preparation for statistical analysis. We argue that time spent on data management activities enables and enhances the production of robust, credible analyses.

Research data management in a nutshell

1. When embarking on the exploration of a new research question, after the literature review, and the formulation of preliminary hypotheses, the next task is generally to begin to identify (a) what variables you need in order to test your hypotheses, and run your models (b) what datafiles (if any) are already available that contain those variables, or whether and how to collect new data, and (c) what software package(s) has the statistical analysis routines and related capabilities (data cleaning, data transformations, etc) that you require.

2. Every researcher wants to get straight to analysing data and publishing results. Good! However, doing good analysis and producing credible, defensible results is dependent on adhering to the four pillars of wisdom:
    a) **Effectiveness**: performing tasks correctly; minimising information loss and errors in analysis and outputs.
    b) **Efficiency:** maximising features in software e.g. using loops to perform repetitive tasks.
    c) **Transparency:** the audit trail showing what you did, why, when and how.
    d) **Replication:** same results every time, whoever and whenever.

    These ideas, and more, are covered elsewhere but for now, it is important to recognise the role of data management in enabling and enhancing the production of robust, credible analyses. Good research data management (RDM) practices help you implement the four pillars of wisdom in your work; being able to rerun analyses at the click of a button in response to suggestions by your supervisor is one example of efficiency. Your work is transparent to reviewers etc through clear documentation of each step in the analysis e.g. which variables and cases were included. Table 1 below outlines some of the main principles and practices associated with good data management. The next stage is to implement these ideas via an appropriate statistical software package.

Table 1. Principles and practices of RDM

Principles	Practices
Use syntax/code to capture every data management and analysis task Maintain a high degree of documentation (the audit trail) Standardisation/consistency of approach Organising files and tracking versions	Getting data and loading it into software Inclusion of variables and cases Dealing with missing values or duplicate cases Checking for invalid/extreme values Variable operationalisation Data linkage Exporting and/or sharing data

3. The questions you need to be able to answer, vis-à-vis any software you decide to use, are:
    a) Does the software support the statistical analyses that are most appropriate for my research question and data?

b) How good/defensible are the measures that the software will produce?
c) Will it support the data exploration and data cleaning/transformations I need to perform?
d) How will I get my data into the software (i.e. what file formats can it read)?
e) Equally importantly, how can I get my data out of that software (along with any transformations, computations etc) so that I can read it into other software for other analyses, or store it in a software-neutral format for the longer term?

This workbook assumes you have decided to use SPSS for your data cleaning and analyses, at least in part. The workbook and the exercises are based on SPSS versions 21/22/23 for Windows.

4. Advantages to SPSS include:
   - flexible input capabilities (eg hierarchical data formats)
   - flexible output capabilities
   - metadata management capabilities, such as variable and value labels, missing values etc
   - data recoding and computing capabilities
   - intuitive command names (for the most part)
   - statistical measures comparable to those from SAS, Stata, etc.
   - good documentation and user support groups (see Appendix A)

   Disadvantages to SPSS include:
   - doesn't do all possible statistical procedures, but then, no statistical package does
   - does not handle long question text well
   - allows very long variable names (>32 characters) which cannot be read by other statistical packages
   - default storage formats for data and output log files are software-dependent (but this is also true for most statistical packages)

Getting started with SPSS

5. The data being used in this exercise are a subset of variables and cases from:
   Sandercock, Peter; Niewada, Maciej; Czlonkowska, Anna. (2014). International Stroke Trial database (version 2), [Dataset]. University of Edinburgh, Department of Clinical Neurosciences. http://dx.doi.org/10.7488/ds/104. The citation specifies that this is 'version 2'. An important part of data management is keeping track of dataset versions and documenting the changes that have happened between versions, especially of your own data. The web page describing the data set has that information.

   This handbook, and the zip file containing the example data files within an example folder structure can be downloaded from: http://datalib.edina.ac.uk/mantra/softwarepracticals.html

Download the zip file and then extract it to a suitable drive which you know you will be able to access via SPSS. You should find the following files in the extracted folders:

- \raw_data\ist_corrected_uk.csv – a comma-delimited file, which we will convert to an SPSS system file
- \clean_data\ist_corrected_uk_addvar.sav – an SPSS system file from which we will add variables
- \clean_data\ist_corrected_eu15_addcas.sav – an SPSS system file from which we will add cases
- \syntax\data_management\ist_dm_20170328_v1.sps – an SPSS syntax file to add variable-level metadata to the SPSS file
- \documents\ist_logfile_20170208.xlsx – a sample log file in Excel format

6. As part of managing your data it is important to create your own documentation as you work through your analyses. For example it is good practice to set up a *Data log* right at the start of a project, to keep track of e.g. the locations of versions of datafiles and documentation, notes about variables and values, and file and variable transformations, output log files, etc. This is information that is easily forgotten, and should you make a mistake, could help you backtrack to a good version of your dataset.

   If you choose to keep a data log file it will also help you, at the end of your project, to identify what versions of the data file(s), syntax files, and output files to keep, and which can be discarded. At a minimum, you should keep the original and the final versions of the datafile(s), as well as the syntax files or output files which include the syntax, for all transformations, as well, of course, as keeping the data log file. Keeping these files will assist you to defend your transformations and analyses, should they be queried in the future. You do not need to create a Data log file for this session.

7. The software you choose in which to manage your data log is a matter of personal choice. Some researchers prefer to use a word processor (e.g. MS Word), others to use a format-neutral text editor, such as Notepad++, and yet others prefer the table handling and sorting capability of Microsoft Excel (see the file "ist_logfile_20170208.xlsx"). The following are suggested fields for the Data log file:

   - Date (YYYY-MM-DD)
   - The input file location and format ('format' is especially important if you are working in a MacOS environment, which does not require format based filename extensions). The first entry should be where you obtained the data (if doing secondary analysis).
   - The output file location, name and format
   - A comment as to what was done between input and output.
   - If using Excel, rename the sheet, e.g. 'data log'.

Date	Input path	Input filename	Output path	Output datafile	Output SPSSlog	process
2014-11-05	M:\teach_spss\strokev2	IST_corrected.csv	M:\teach_spss\strokev2	IST_corrected_sorted.csv		sorted on CNTRYNUM (n=19,435, vars=112)
2014-11-05	M:\teach_spss\strokev2	IST_corrected_sorted.csv	M:\teach_spss\strokev2	ist_corrected_uk.csv		subset (CNTRYNUM=27), n=6,257 vars=112
2014-11-05	M:\teach_spss\strokev2	IST_corrected_sorted.csv	M:\teach_spss\strokev2	ist_corrected_eu15.sav	20141105_Output1.spv	subset (any(CNTRYNUM,2,3,8,9,10,11,14,15,19,22,24,31)), n=6,561 vars=112
2014-11-06	M:\teach_spss\strokev2	IST_corrected.sav	M:\teach_spss\strokev2	ist_corrected_uk.sav		added respno variable, subset (CNTRYNUM=27), n=6,257 vars=113
2014-11-06	M:\teach_spss\strokev2	IST_corrected.sav	M:\teach_spss\strokev2	ist_corrected_eu15.sav		added respno variable, subset (any(CNTRYNUM,2,3,8,9,10,11,14,15,19,22,24,31)), n=6,561 vars=113
2014-11-06	M:\teach_spss\strokev2	ist_corrected_uk.sav	M:\teach_spss\strokev2	ist_corrected_uk1.sav		all vars except followup (FDEAD to FOAC, FU2_DONE) n=6,257 vars=102
2014-11-06	M:\teach_spss\strokev2	ist_corrected_uk.sav	M:\teach_spss\strokev2	ist_corrected_uk2.sav		followup VARS (FDEAD to FOAC, FU2_DONE, respno) n=6,257 vars=12

Hint: in order to get the correct path and filename of a file in a Windows environment, locate the file in Windows Explorer, and either:

Alternative 1: Click on the folder icon in the far left corner of the address bar. Copy and paste the file path, and add the following to the end: \filename.type e.g. C:\Documents\Thesis\ChapterOne.docx.

Alternative 2: Click on the file to select it. Then right-click, and select 'Properties'. The exact path will be displayed in the 'Location' field of the properties window, and the filename in the first field. Both path and filename can be copied and pasted into the data log.

8. It is good practice to assume that you may not always be using SPSS, or the same version of SPSS, for your analyses. You may need to migrate data from/to different computing environments (Windows, Mac, Linux/Unix) and/or different statistical software, because no statistical package supports all types of analysis (SAS, Stata, R, etc). Therefore you also need to be aware of constraints on lengths of filenames, variable names, and other metadata such as variable labels, value labels, and missing values codes in different operating systems and software packages, some of which are listed in Appendix B.

9. **Note:** Especially if you are in the habit of working in different computer environments, it is not recommended that you use spaces ('blanks') in file or directory/folder names. Different operating systems treat embedded spaces/blanks differently. Instead, use e.g. underscores or CamelCase to separate words to make names more readable. Ie, not "variable list.xlsx" but "variable_list.xlsx" or "VariableList.xlsx".

Running SPSS and configuring options

10. **Open SPSS** through your programs menu: **All Programs > IBM SPSS Statistics [nn]**. If a dialog box appears asking you whether you wish to open an existing data source, click '**Cancel**'. When you run SPSS in Windows, one window is opened automatically:
    - a **Data editor window** - empty until you open a data file or begin to enter variable values, after which it will have two views, a **Variable View** and a **Data View**.

    Once you have loaded a data file, or issued a command from the drop-down menus, a second window will open:
    - an **Output window**, to which output from your commands and error messages will be written.

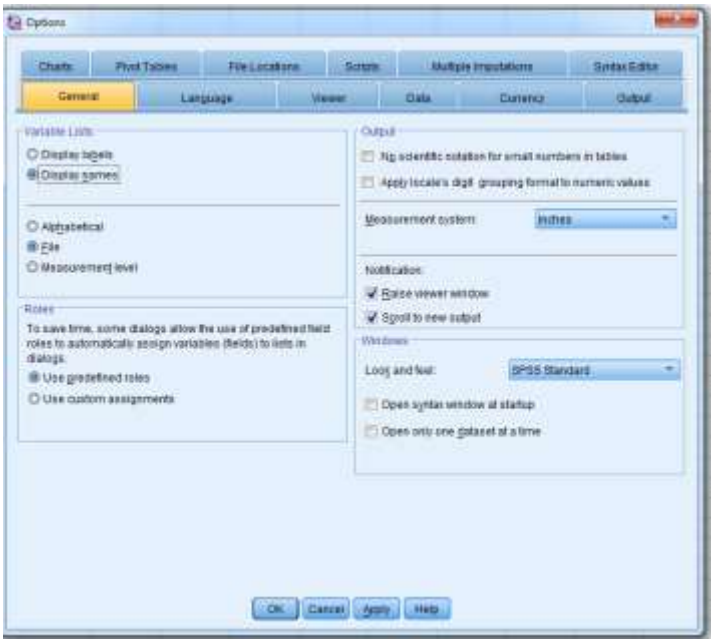
    Additional windows which can be opened from **File > New**[1] or **File > Open** (if they already exist) are:
    - a **Syntax window**, in which you can enter syntax directly, or 'paste' syntax from the drop-down menu choices, edit and run syntax.
    - a **Script** window, in which you can enter, and edit, Python scripts.

    Three additional windows, in addition to dialogue windows etc., may or may not open depending on the procedures you are running:
    - a Pivot table editor window;
    - a Chart editor window;
    - a Text output editor window.

11. Before starting to import data, you should make some changes to the SPSS environment defaults. Select **Edit > Options**. The **Options** box has several tabs.

12. Select the **General** tab, look under '**Variable Lists**', and make sure that '**Display names**' and '**File**' are selected. This will ensure that the variables in the dataset are displayed by variable name rather than by variable label and that variables are listed in the same order as they occur in the dataset – knowing this order is essential when referring to individual variables or ranges of variables, e.g. in recode and/or compute commands.

[1] Unless otherwise specified, all following **Menu > [Submenu] > Item** references are to drop-down menus in the SPSS interface.



13. It is also useful, especially for checking and recoding purposes, to see variable names and values in the output. By default SPSS shows only labels, not variable names or value codes. Click on the '**Output'** tab, and under both '**Outline Labeling**' and '**Pivot Table Labeling**', select the options to show:
    - Variables in item labels shown as: '**Names and Labels'**,
    - Variable values in item labels shown as: '**Values and Labels**'.

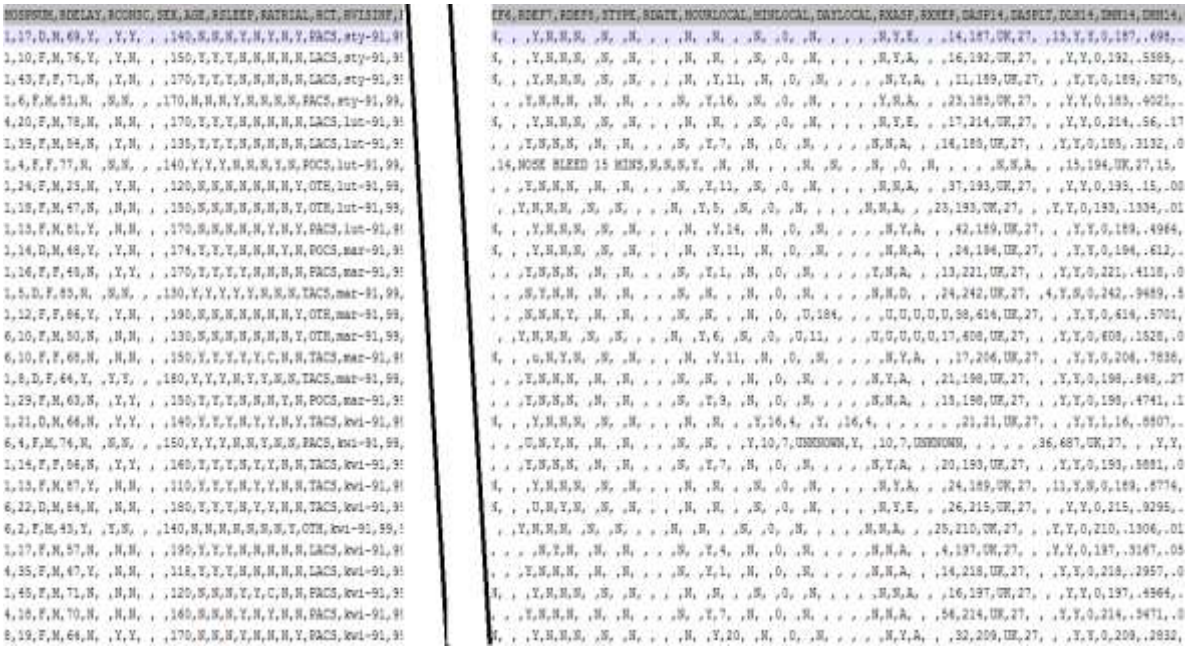
14. Finally, select the '**Viewer'** tab and ensure that the **'Display commands in the log'** checkbox (bottom left of the screen) is checked. This causes the SPSS syntax for any procedures you run to be written to your output file along with results of the procedure. This is useful for checking for errors, as well as a reminder of the details of recodes and other variable transformations, etc. **Click 'OK'** to save the changes.

Examine the data file

15. First we will look at one common type of external, raw data file, in this case a comma-delimited file, with extension '.csv'. Run **Notepad++** and open the file '**ist_corrected_uk.csv**'.

> *Jakobsen's Law*. "If Excel can misinterpret your data, it will. And usually in the worst possible way". *<http://asiapacificparllibs.org/wp-content/uploads/sites/2/2015/12/Australia-Sam-Spencer-Parliamentary-Data-Management.pdf>*

Note: Do not use Excel to open the file. Notepad++ (or Notepad) will display the file in a format-neutral way, in a non-proportional font, so that we can see what the file really contains, rather than what Excel interprets the content to be.



In this data file, each row or unit of observation represents a stroke patient in the IST sample: patients with suspected acute ischaemic stroke entering participating hospitals in the early 1990s, randomised within 48 hours of symptom onset. The variables in the rows describe characteristics of the patients, their symptoms, treatment, and outcomes. This particular subset contains patients from the UK only, and only those variables describing the patient at the time of enrollment in the trial, and at the 14 day follow-up.

This a simple flat .csv file, with one unit of observation (case) in each row, and the variables relating to that case, in the same order, making up the row, separated by commas. Using the cursor to move around the file, determine:

How many cases (rows) are there in this dataset? *(Hint: scroll down and click on the last row. The number of the row is given by Ln in the bottom ribbon of the screen)*[2]

Is there a row of variable names as the first row? Y|N

Are there blanks in the data, between commas (the delimiters)? Y|N

Are there blanks embedded among other characters in individual fields? Y|N

Are comment fields and/or other alphabetic variables enclosed in quotation marks? Y|N

Are full stops or commas used to indicate the position of the decimal in real numbers?
NB: SPSS requires that all decimal places be indicated by full stops.

In SPSS, variables must be assigned a **variable name**, which must follow certain rules (see below). Each variable may optionally also have a **variable label** which provides a fuller description of the content of the variable. The variable label is free text, up to 256 characters long, and is often used to give e.g. the text of a survey question.

Variable name: likeCameron
Variable label: How much do you like or dislike David Cameron

Rules for variable names in SPSS: (a) variable names must be unique in the data set, (b) must start with a letter, (c) be short, about 8 characters is best (d) **must not contain spaces** but may contain a few special characters such as full stop, underscore, and the characters $, #, and @, (e) should not end with a full stop, and (f) should reflect the content of the variable. Variable names beginning with a '$' (e.g. $CASEID, $CASENUM, $DATE, $SYSMIS, etc) are system variables – do not use these as regular variable names. Also, do not use names of SPSS commands as variable names.

Not variable names:

[2] **T**o enable the display of the number of lines and line length in Notepad (as opposed to Notepad++), turn off **Format > Word Wrap** and click on the last line of the file. The line number of the last line will be displayed at the bottom of the screen.

- Patient #
- # chemo cycle
- 7. On a scale of 1 to 5, where 1 is 'disagree' and 5 is 'agree', please tell us … [etc]

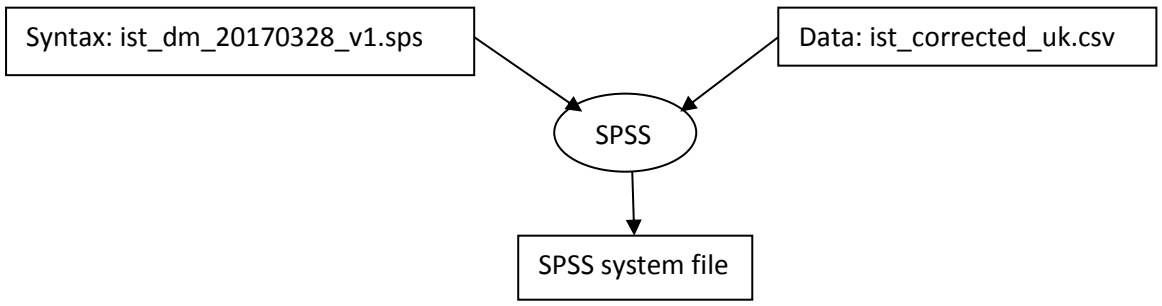
Good variable names:

- Patient# or Patient_no
- chemo_cycle# or ChemoCycleNo
- q7, or agree7, or disagree7

Working in SPSS I: Getting and Exploring Data

[EXERCISE 1 and 2]

16. In common with most statistical packages, SPSS needs a variety of information in order to read a raw numeric data file: (a) the data, and (b) instructions as to how to read the data. In its simplest form, SPSS reads a **raw data file** (eg 'ist_corrected_uk.csv'), a **syntax file** (eg 'ist_dm_20170328_v1.sps'), and using the input instructions in the data and syntax files, converts data and metadata into its preferred format, a **system file** (extension '.sav'), which exists only during your current SPSS session **unless you save it**.



17. You can carry out most of your data management and analysis activities (e.g. creating new variables, producing frequency tables) in SPSS using drop-down menus. Alternatively, you can also analyse and manipulate your data using SPSS command language (syntax), which you can save and edit in a 'syntax file', rather than using drop-down menus. A syntax file contains the set of instructions for SPSS to follow in order to perform data management and analysis tasks. At first glance, utilising the drop-down menu seems the more intuitive choice: on the surface it is easier to understand and more visually engaging. However, the use of the drop-down menu is not consistent with the principles of good data management and can lead to some considerable issues during the research workflow. Using syntax is more efficient, minimises errors, records every step in the data management and analysis workflow, and facilitates rapid replication of work. In summary, **all serious analyses use syntax**. See Appendix D for a detailed description of using SPSS syntax files, including how to create them and the rules governing their use.

18. There are four main ways to collect the syntax you need:
    a. From the drop-down menus, written to the Output window and copied to the syntax file

b. Clicking on the 'Paste' button of appropriate procedure windows, writes directly to the syntax file

c. **Writing it from 'scratch' or using templates based on the explanations and examples in the appropriate SPSS manual (see Appendix A)**

d. **From other external sources on the world wide web (Google is your friend)**

19. From the SPSS drop-down menus, select **File > Read Text Data.** In the Open Data window, browse to where you placed the '**ist_corrected_uk.csv**' file, and finally click on '**Open**'. (This will not work if the file is already open in Excel.)

    This will launch the **SPSS Text Import Wizard**, a 6-step sequence that will request instructions from you as to how to read the .csv file. Remember the answers you gave to the questions in paragraph 15, above, as you work through the steps, particularly in step 2 (yes, you have a row of headers) and step 4 (no, a Space is NOT a field delimiter in this file, only the comma is, and no, there is no 'text qualifier').

    SPSS will use your input as well as the data in the first 200 cases to automatically compile a format specification for the file. NB if any variable field is longer in later cases than in any instance in the first 200 cases, the content of the longer fields will be truncated.

20. You should, at the end of the Import Wizard process, have three SPSS windows:

    A **Data Editor: Data View** window which contains a spreadsheet-like display of the data:

	HOSPNUM	RDELAY	RC	SEX	AGE	RSL	RATRIAL	RCT	RVI	RHEP24	RASP3	RSBP	RD	RD	RD	RD	RD	RD	RD	STY...	RDATE
1	1	17	D	M	69	Y	.	Y	Y	.	.	140	N	N	N	Y	N	Y	N	Y	PACS	sty-91
2	1	10	F	M	76	Y	.	Y	N	.	.	150	Y	Y	Y	N	N	N	N	N	LACS	sty-91
3	1	43	F	F	71	N	.	Y	N	.	.	170	Y	Y	Y	N	N	N	N	N	LACS	sty-91
4	1	6	F	M	81	N	.	N	N	.	.	170	N	N	N	Y	N	N	N	N	PACS	sty-91
5	4	20	F	M	78	N	.	N	N	.	.	170	Y	Y	Y	N	N	N	N	N	LACS	lut-91
6	1	39	F	M	54	N	.	Y	N	.	.	135	Y	Y	Y	N	N	N	N	N	LACS	lut-91
7	1	4	F	F	77	N	.	N	N	.	.	140	Y	Y	Y	N	N	Y	N	N	POCS	lut-91

    A **Data Editor : Variable View** window contains a list of variable names and their associated characteristics :

	Name	Type	Width	Decimals	Label	Values	Missing	Columns	Align	Measure	Role
1	HOSPNUM	Numeric	2	0		None	None	8	Right	Scale	Input
2	RDELAY	Numeric	2	0		None	None	8	Right	Scale	Input
3	RCONSC	String	1	0		None	None	1	Left	Nominal	Input
4	SEX	String	1	0		None	None	1	Left	Nominal	Input
5	AGE	Numeric	2	0		None	None	8	Right	Scale	Input
6	RSLEEP	String	1	0		None	None	1	Left	Nominal	Input
7	RATRIAL	Numeric	1	0		None	None	8	Right	Nominal	Input
8	RCT	String	1	0		None	None	1	Left	Nominal	Input
9	RVISINF	String	1	0		None	None	1	Left	Nominal	Input
10	RHEP24	Numeric	1	0		None	None	8	Right	Nominal	Input
11	RASP3	Numeric	1	0		None	None	8	Right	Nominal	Input

Note that we have no information on what values the variables contain or what the their names mean.

An **Ouput window** listing the syntax SPSS used to read the input data file:

```
GET DATA  /TYPE=TXT
  /FILE="\\sg.datastore.ed.ac.uk\sg\edina\users\lruus\data\ist2\subsets\ist_corrected_uk1.csv"
  /ENCODING='Locale'
  /DELCASE=LINE
  /DELIMITERS=","
  /ARRANGEMENT=DELIMITED
  /FIRSTCASE=2
  /IMPORTCASE=ALL
  /VARIABLES=
  HOSPNUM F2.0
  RDELAY F2.0
  RCONSC A1
  SEX A1
  AGE F2.0
  RSLEEP A1
  RATRIAL F1.0
  RCT A1
  RVISINF A1
    ….LOTS OF LINES DELETED
  HTI14 F1.0
  PE14 F1.0
  DVT14 F1.0
  TRAN14 F1.0
  NCB14 F1.0
  respno F3.0.
CACHE.
EXECUTE.
DATASET NAME DataSet1 WINDOW=FRONT.
```

21. SPSS can read (and write) a variety of formats. See Appendix E for a list of software-dependent formats and the SPSS commands to read and write them. SPSS can also read more complex file formats, such as multiple records per case, mixed files, hierarchical files, etc.

    A full SPSS syntax file to read a raw data file contains instructions to SPSS re (a) what file to read (FILE="…") and (b) how to read it (TYPE=TXT), as well as (c) a data list statement listing the variables, their locations and formats (VARIABLES=…), (d) variable labels statements, (e) value labels statements, and (f) missing data statements. Thus far, we have only provided information for (a) through (c), and later will add (d) through (f).

    Data list statement for a fixed field format file:

```
DATA LIST fixed records=1 FILE="[path and filename of input data file]"
 /1       caseid 311-315            hospnum 1-3             rdelay 4-5
          rconsc 6 (A)                sex 7 (A)               age 8-9
         rsleep 10 (A)            ratrial 11 (A)             rct 12 (A)
         rvisinf 13 (A)            rhep24 14 (A)            rasp3 15 (A)
           rsbp 16-18              rdef1 19 (A)             rdef2 20 (A)
         rdef3 21 (A)              rdef4 22 (A)             rdef5 23 (A)
         rdef6 24 (A)              rdef7 25 (A)             rdef8 26 (A)
        stype 27-30 (A)          rdate 31-37 (A)         hourlocal 38-39.
```

Data list statement for a delimited file with one case per row, and no column headers:

```
DATA LIST list FILE="[path and filename of input data file]"
 /        hospnum rdelay rconsc (a) sex (a) age (f2.0) rsleep (a) ratrial (a)
          rct (a) rvisinf (a) rhep24 (a) rasp3 (a) rsbp (f3.0) rdef1 (a) rdef2 (a)
          rdef3 (a) drdef4 (a) rdef5 (a) rdef6 (a) rdef7 (a) rdef8 (a) stype (a)
          rdate (a7) hourlocal (f2.0).
```

Checking and saving the output

22. **Checking**: (1) check the Output window for Error messages, (2) click on the Data Editor window, and check both the **Variable View**, and the **Data View**, for anything that looks not quite right. If there are errors, try to figure out what they are. Normally, fix the first error first, and then rerun the job – errors often have a cascading effect, and fixing the first can eliminate later errors.

    Scroll through the Data View window, up, down and sideways, to CHECK that each variable contains the same type of coding, eg there are no words in a column with numeric codes, etc.

How many cases have been read? Is this the same as the number of rows in the raw data?
Are there the same number of variable names and columns of data? *(SPSS assigns default names 'VAR[nnn]' to unnamed variables.)*
Does each column appear to contain the same type and coding range of data?
Have variables containing embedded blanks, eg comment fields, been read correctly?
Do any variables (eg comment fields) appear to have been truncated?
Have numbers containing decimals been read correctly?

23. **Saving the work so far**

    a. **Save the SPSS system file**. We have two options for saving our data: use the syntax command 'sav' or select **File > Save as** and save the file with format 'SPSS Statistics (*.sav)'. One method to distinguish among versions of a file is to begin or end each filename with the YYYYMMDD of the date on which it was created, eg:

        i. ist_corrected_uk_20170317.sav

b. **Save the Output file.** In current versions of SPSS, the **Output Viewer** is labelled 'Output[n] [Document[n]] IBM SPSS Statistics Viewer'). It is the window in which output from your procedures is displayed, as well as the syntax that generated it. The Output window should now contain the syntax SPSS used to read the .csv file. For data management purposes, this Output file is important documentation and your record of what you have done to the file/variables and what the results were. Therefore, it is very important to keep these output files.

   This output file can be saved. By default the output file is saved as an SPSS-dependent format with default filename 'Output[n]', and the extension .spv (.spo in versions prior to SPSS18) which can only be read by SPSS; therefore, instead of saving it, use **File > Export** to convert it to a .txt, .html or .pdf format (which you will be able to read with other software), with a meaningful filename. And, of course, add this information to the Data log file.

c. **Save the syntax file**: **File > Save**. Note that an SPSS syntax file takes the default extension '.sps', and is a flat text file, so readable by any text software.

d. **Update the Data log file** (Excel) with the names and locations of these files.

Adding variable and value labels, and user-defined missing data codes

24. **Common metadata management tasks in SPSS:**
    a. Rename variables
    b. Add variable labels
    c. Optimize variable labels for output
    d. Add value labels to coded values, eg '1' and '2' for 'male' and 'female'
    e. Optimize length and clarity of value labels for output
    f. Add missing data specifications, to avoid the inclusion of missing cases in your analyses
    g. Change size (width), write and/or print formats, and # of decimals (if applicable)
    h. Change variable measure type: nominal, ordinal, or scale

25. Metadata help us understand the values and content of a data file e.g. the meaning of variable names and values. There are a number of additional types of metadata that can be added to an SPSS system file, to make output from your analyses easier to read and interpret. A syntax file to read in a raw dataset normally consists of the following basic parts:
    - Data list: the data list statement begins with an indication of the type of raw file (.csv, .tab. fixed-field; flat, mixed or hierarchical; see Appendix E), as well as the input data file path and filename. When reading in the data using drop-down menus (as we have done), SPSS takes this information from the **Text import wizard**, launched via **File > Read text data**.
    - Data list subcommand: Variable names, locations, and formats. SPSS takes the information about variable names, [relative] locations, and formats, from input to the **Text import wizard**, the first row of the data file (if it contains variable names), and the first 200 lines of a .csv, .tab, or blank delimited input file. If the file is a fixed-field format file, or a delimited file with no column headers (but delimited by eg tabs, commas, or blanks), a Data list subcommand listing variable names, column locations (for fixed-field files), and formats is essential.

- Variable labels: explanatory labels for each variable, eg is 'weight' a sample weight, or the weight of the respondent in kilograms/pounds/stone? Variable labels should be succinct enough to allow one to quickly decide, from the first 20-40 characters, which variable(s) to select. This is not the place for full question text – ie don't have 5 variables in a row that start "On a scale of 1 to 5….", eg

  Instead, a shorter variable label, omitting all extraneous words, such as 'Attended: creating a DMP' is more efficient and informative.
- Value labels represent an explanatory label that is applied to the values of a variable. They help you understand what the numeric values refer to. E.g. 1: 'disagree', 5: 'agree'; or 1: 'female', 2: 'male'; 'D': 'drowsy', 'F': 'fully alert'; 'U': 'unconscious'; or 1: 'female' and 2: 'male'.
- Missing data: assigns certain variable values as user-defined missing (as opposed to system-missing, ie blank fields, unreadable codes, etc), which affects how variables are used in statistical analyses, data transformations, and case selection. More about missing values in paragraph 25.

These additional characteristics can be entered, for each variable, directly into the **Data Editor: Variable View** window, although this can become quite tedious, depending on how many variables are in the data set. Alternatively, you can use a syntax file to batch-add this information. A SPSS syntax file(s) containing the commands to read a data file into SPSS may accompany the data obtained from a secondary source (such as a data archive/data library) or you may need to create it using the information included in codebooks, questionnaires, and other documentation describing the data file.

26. There are two types of **missing values**:
    a. **System missing** – blanks instead of a value, ie no value at all for one or more cases (name=SYSMIS)
        i. Note: $SYSMIS is a system variable, as in:
```
IF (v1 < 2) v1 = $SYSMIS.
```
        Whereas SYSMIS is a keyword, as in:
```
RECODE v1 (SYSMIS = 99).
```
    b. **User-defined missing** – values that should not be included in analyses, eg "Don't know", "No response", "Not asked". These are often coded as '7, 8, 9' or '97, 98, 99' or '-1, -2, -3', or even 'DK' and 'NA'.

User-defined missing can be recoded into system missing, and vice versa; we'll practise this task later in the session.

Descriptive statistics: checking the variables

[EXERCISE 3]

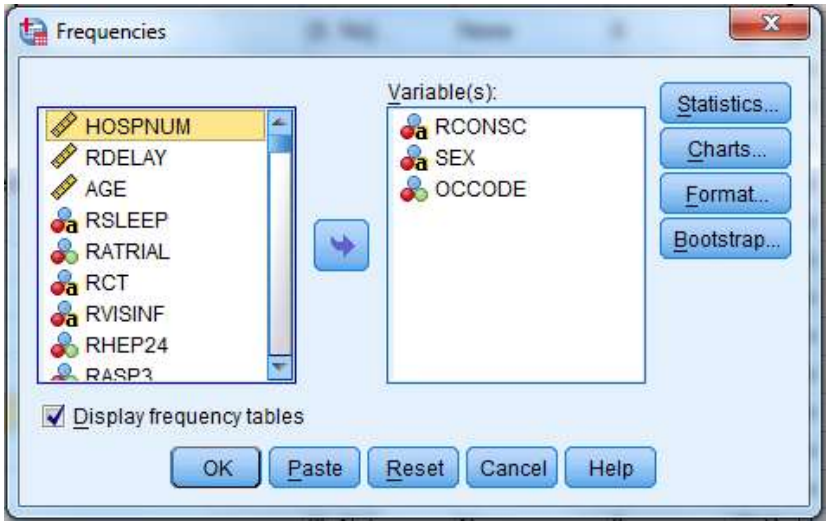
27. Why run descriptive statistics?
    a. Determine how values are coded and distributed in each variable
    b. Identify data entry errors, undocumented codes, string variables that should be converted to numeric variables, etc
    c. Determine what other data transformations are needed for analysis, eg recoding variables (such as the order of the values), missing data codes, dummy variables, new variables that need to be computed
    d. After any recode/compute procedure, ALWAYS check resulting recoded/computed variable against original using FREQUENCIES and CROSSTABS

28. SPSS can display the number of cases coded to each value of each variable, undocumented codes, missing values, etc. Run these basic procedures to familiarise yourself with a new dataset, new or recoded/computed variables, and to check for problems.

29. **Continuous ('scale') variables:** To generate descriptive statistics for continuous variables (eg AGE), the type of information provided by frequencies is often not informative. We need a different set of commands. See the accompanying syntax file for some examples.

30. **Nominal and ordinal ('categorical') variables**: As an alternative to syntax, frequencies for variables (string or numeric, including continuous variables with > 50 values) can be run through the drop-down menus by clicking on **Analyse > Descriptive statistics > Frequencies**, selecting the variables[3], moving them into the right part of the screen, and then clicking OK. In the example below we have chosen **rconsc, sex,** and **occode –** of which the first two are string variables and the last is defined as a numeric, nominal variable (according to the Measure column in the Data Editor variable view).

[3] Due to the SPSS configuration changes in paragraphs 11-14, the left window should list variable names rather than variable labels. To list variable names in alphabetic order, hover the mouse over the variable list, click R-mouse button, and select 'List alphabetically'.



Notice the difference in variable type icons to the left of each variable name in variable selection menus. SPSS does its best to guess the type of each variable when the data are read in, but the types can also be set/changed in the **Data editor > Variable view** window ('Measure' column):

 indicates a string or alphabetic variable,

 indicates a nominal variable,

 an ordinal variable, and

 a scale or continuous variable.

You can see from the first table in the output below that one of our variables has missing data for some of the cases.

```
FREQUENCIES VARIABLES=rconsc sex occode.
```

Frequencies

Statistics

		RCONSC Randomisation: Conscious state at randomisation	SEX Randomisation: Sex	OCCODE Six month outcome
N	Valid	6257	6257	6252
	Missing	0	0	5

The second table contains the frequencies, percents, valid percents (not including missing values), and cumulative percents:

Frequency Table

RCONSC Randomisation: Conscious state at randomisation

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	D drowsy	1576	25.2	25.2	25.2
	F fully alert	4615	73.8	73.8	98.9
	U unconscious	66	1.1	1.1	100.0
	Total	6257	100.0	100.0	

SEX Randomisation: Sex

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	F female	3032	48.5	48.5	48.5
	M male	3225	51.5	51.5	100.0
	Total	6257	100.0	100.0	

OCCODE Six month outcome

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	1 dead	1712	27.4	27.4	27.4
	2 dependant	3240	51.8	51.8	79.2
	3 not recovered	664	10.6	10.6	89.8
	4 recovered	636	10.2	10.2	100.0
	Total	6252	99.9	100.0	
Missing	0 missing status	2	.0		
	9 missing status	3	.0		
	Total	5	.1		
Total		6257	100.0		

31. It is good practice to use comments to give each group of commands a header explaining what the syntax is doing, and, if you are working as part of a team with shared files and file space, name of the person who wrote the syntax and the date it was written. If you highlight and run Comments, together with the syntax to which they refer, the comments will also be echoed in your Output Window.

32. Output and syntax files should be saved for future reference and use the Data log file to record the name of each, what each contains, and where it is located. SPSS syntax files are flat text files, with the default extension .sps (see paragraph 23 for information about the Output file), and so only need to be saved. As the Data log file grows, you may find it easier to add new information at the top of the table (ie reverse chronological order) rather than the bottom.

Working in SPSS II: Transforming and Exporting Data

[EXERCISE 4 and 5]

Recoding string variables (recode), or creating derived variables

33. **Common recoding tasks, ie recoding existing variable(s)**
    - Convert string variables to numeric
    - Change order of values of variables (nominal → ordinal)
    - Change system missing to user-defined missing
    - Collapse categories
    - Replace missing values with eg variable mean
    - Creating dummy variables

    a) This data file contains a large number of string (alphabetic) variables, eg variables coded as Y=yes and N=no, etc. The statistical uses of string variables are very limited. In order to make maximum analytical use of string variables, they must be recoded into numeric variables. **ALWAYS recode into a new variable,** otherwise you will overwrite the existing variable and lose the original values. I like to use the original variable name, with '_r' appended (eg 'age' recoded into 'age_r') to indicate that the new variable is a recode, and what the source variable was, but this is not the only way to keep track of these parent-child relationships. See the accompanying syntax file for how you can recode variables. SPSS will change all string characters '1' to number '1', string character '2' to number '2', string character '3' to number '3', etc., in addition to the changes specified, so that the output 'varname_r' is a numeric variable, rather than a string variable.

        i. [OPTIONAL] If a variable is defined in SPSS as a string variable because 'NA', 'DK' or similar have been used as missing data codes, but otherwise consists of numbers that you want to preserve, you can use the 'CONVERT' option to the RECODE command.

b. **Collapsing categories**: we are often interested in grouping conceptually similar categories and values e.g. 'agree' and 'strongly agree'. We can do so using the recode command – run the syntax below to see how we can group values of our age variable.

```
*Recoding individual years of age into age groups.
RECODE age (lo THRU 60=1)(61 THRU 80=2)(81  THRU hi=3)(else=9) INTO agegrp.
```

34. **Checking recodes:** Regardless of the recode method you choose, you should alway run the frequencies and crosstabulations commands on the source variable and the new recoded variable, including missing values, to ensure that that all values have been captured correctly, and that you are satisfied with the recodes, etc - run the syntax below (don't forget to create the recoded version of 'sex').

```
FREQUENCIES VARIABLES=sex sex_r.
CROSSTABS   /TABLES=sex BY sex_r.
```

If the output variable is a continuous variable with many values, producing frequencies and crosstabulations for all possible values may not be feasible. But you should examine very carefully the correspondence of missing data codes, and other values that have been specifically mentioned in the RECODE statement:

```
* Frequencies for selected values of a continuous variable, for all values higher than 85.
TEMPORARY.
SELECT IF (age GE 85).
FREQUENCIES VARIABLES=age agegrp.
CROSSTABS / TABLES=age by agegrp.
```

Missing data

35. SPSS recognises two classes of missing data: system-missing and user-defined missing.

System-missing are blank variable fields, ie cases with no valid code for a variable (eg a blank); these are automatically coded as system-missing by SPSS.

User-defined missing values are valid codes, often with labels such as 'unknown', 'not applicable', 'not asked', etc., but are values that a researcher may want to include in some statistical analyses, but exclude in others. These must be explicitly defined via a missing values statement, or the missing values defined in the 'Variable View' of the Data Editor window.

```
* Recoding a string variable and defining a user-defined missing value.
RECODE dsch  ('Y'=1)('N'=0)('U'=8) INTO dsch_r.
MISSING VALUES dsch_r (8).
FREQUENCIES VARIABLES=dsch dsch_r.
```

Missing values (both user-defined and system-missing) are by default included in frequencies (SPSS 21 and later versions), but not in valid or cumulative percentages, nor in descriptive statistics, cross-tabulations, charts or histograms, etc.

System missing values are identified as 'Missing System' in frequencies output. User-defined missing are identified simply as 'Missing':

System missing (original variable):

FDEADC

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	0	13	.2	.7	.7
	1	637	10.2	36.0	36.7
	2	228	3.6	12.9	49.6
	3	23	.4	1.3	50.9
	4	462	7.4	26.1	77.0
	5	135	2.2	7.6	84.6
	6	46	.7	2.6	87.2
	7	131	2.1	7.4	94.6
	8	95	1.5	5.4	100.0
	Total	1770	28.3	100.0	
Missing	System	4487	71.7		
Total		6257	100.0		

User defined missing (recoded variable):

fdeadc_r

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	.00	13	.2	.7	.7
	1.00	637	10.2	36.0	36.7
	2.00	228	3.6	12.9	49.6
	3.00	23	.4	1.3	50.9
	4.00	462	7.4	26.1	77.0
	5.00	135	2.2	7.6	84.6
	6.00	46	.7	2.6	87.2
	7.00	131	2.1	7.4	94.6
	8.00	95	1.5	5.4	100.0
	Total	1770	28.3	100.0	
Missing	9.00	4487	71.7		
Total		6257	100.0		

What codes to use for missing data is a matter of personal preference. Some prefer to consistently reserve the highest numeric codes, eg codes '7', '8', and '9' for various missing data values, such as 'don't know', 'not applicable', and 'not asked' respectively (or '97', '98', and '99' for 2-digit codes, '997', '998', and '999' etc). Others prefer to use negative values, eg '-7', '-8', and '-9', and some use alphabetic codes 'DK' and 'NA'.

36. We can explicitly request user-defined missing values (but not system-missing values) be included in crosstabs using syntax – run the commands below:

```
RECODE dplace ('A'=1) ('B'=2) ('C'=3) ('D'=4) ('E'=5) ('U'=8) (ELSE=9) INTO dplace_r.
VARIABLE LABELS dplace_r 'Other 14 day event: discharge destination - recoded'.
VALUE LABELS dplace_r 1 'home' 2 'relatives home' 3 'residential care' 4 'nursing home'
  5 'other hospital departments' 8 'unknown' 9 'not coded'.
MISSING VALUES dplace_r (8,9).
FREQUENCIES VARIABLES=dplace dplace_r.
CROSSTABS / tables= dplace_r by dplace / MISSING=INCLUDE.
```

Which produces the following output:

dplace_r Other 14 day event: discharge destination - recoded * DPLACE Other 14 day event: Discharge destination Crosstabulation

Count

		DPLACE Other 14 day event: Discharge destination							
			A home	B relatives home	C residential care	D nursing home	E other hospital departments	U unknown	Total
dplace_r Other 14 day event: discharge destination - recoded	1.00 'home'	0	1777	0	0	0	0	0	1777
	2.00 'relatives home'	0	0	42	0	0	0	0	42
	3.00 'residential care'	0	0	0	28	0	0	0	28
	4.00 'nursing home'	0	0	0	0	84	0	0	84
	5.00 'other hospital departments'	0	0	0	0	0	519	0	519
	8.00 unknown	0	0	0	0	0	0	13	13
	9.00 not coded	3794	0	0	0	0	0	0	3794
Total		3794	1777	42	28	84	519	13	6257

Creating new variables (compute)

37. **Common compute operations:**
    a. Create a unique respondent or record identifier
    b. Compute eg an index variable from a group of related variables
    c. Create a new variable from part of an existing variable
    d. Log transformation of a variable
    e. Creating a weight variable from existing variables

38. **Compute** is an important related command that can be used to create a new variable (numeric or string), such as a constant, a sequential case number, a random number, or to initialize a new variable, or used to modify the values of string or numeric variables.

    (a) To create a **unique identifier variable** for each case in a dataset, the syntax is very simple:

```
COMPUTE respid=$CASENUM.
FORMAT respid (F8.0).
```

    (b) To compute an **index variable** as the sum of 8 binary (aka dummy) variables:

```
RECODE rdef1 to rdef8  ('Y'=1)('N'=0)('C'=SYSMIS) INTO rdef1_r  rdef2_r rdef3_r rdef4_r rdef5_r
rdef6_r rdef7_r  rdef8_r.
COMPUTE deficits=SUM(rdef1_r,rdef2_r,rdef3_r,rdef4_r,rdef5_r,rdef6_r,rdef7_r,rdef8_r).
FREQUENCIES deficits.
```

(c) Compute can also be used to **create more than one output variable from parts of an existing variable**. For example, the variable 'rdate' (para. 16) was coded as 'lut-91', 'mar-91', 'sty-91', etc (ie month (in Polish, abbreviated to 3 characters) plus a 2-character year code. This string variable can be unscrambled into two separate numeric variables 'rmonth' and 'ryear'.

First the month component:

```
* Declare a new string variable 'montha'.
STRING montha (a3).
*Compute the new variable=the 1st 3 characters of the original variable 'rdate'.
COMPUTE montha=(substr(rdate,1,3)).
* Recode the new variable 'montha' into a numeric variable 'rmonth' assign labels.
RECODE montha ('sty'=1)('lut'=2)('mar'=3)('kwi'=4)('maj'=5)('cze'=6)
  ('lip'=7)('sie'=8)('wrz'=9)('lis'=11)('gru'=12)(else=10) INTO rmonth.
VARIABLE LABELS rmonth 'Month of randomization - recoded from rdate'.
VALUE LABELS rmonth  1 'January'  2 'February' 3 'March' 4 'April' 5 'May' 6 'June'
  7 'July' 8 'August' 9 'September' 10 'October' 11 'November' 12 'December'.
* Check the new numeric variable.
FREQUENCIES VARIABLES=rdate montha rmonth.
CROSSTABS TABLES=rmonth by montha rdate.
```

And then the year component:

```
* Declare a new string variable 'yra' 2 characters in width.
STRING yra (a2).
* Compute a new string variable 'yra' starting after the dash and 2 characters long.
COMPUTE dash = index(rdate,'-').
COMPUTE yra = substr(rdate,dash+2).
* Recode the new variable 'yra' into a numeric variable 'yr'.
RECODE yra (CONVERT) into yr.
* Compute a new 4-digit numeric variable 'ryear'.
COMPUTE ryear=(1900+yr).
FORMAT ryear (f4.0).
* Label the new variable, and check the frequencies.
VARIABLE LABELS ryear 'Year of randomization - recoded from rdate'.
FREQUENCIES VARIABLES=yra yr ryear.
CROSSTABS TABLES=ryear by yra yr.
```

(d) To **compute the log** of eg a skewed variable, the syntax is:

```
COMPUTE ln_y=LN(y).
VARIABLE LABELS ln_y 'Log of y'.
```

39. As with recodes, where possible, you should always check the accuracy of the results of compute functions using frequencies and crosstabs, (paragraph 47, above) to compare the results of the compute statement(s) against existing variables, where applicable.

    Remember: 'Frequencies' lists missing data, both user defined and system missing. 'Crosstabs' lists only user-defined missing data categories, if explicitly requested via the '/ MISSING=INCLUDE' subcommand.

40. It is important to update your Data log file with information as to how new variables have been computed, and the new values of recoded variables:

    New variable-level information in Data log file - computed or recoded:

1	Variable Information								
2	Variable	Position	Label	Measurement Level	Role	Column Width	Alignment	Print Format	Write Format
105	rdef1_r		Recoded: Face deficit	Dummy					recode
106	rdef2_r		Recoded: Arm/hand deficit	Dummy					recode
107	rdef3_r		Recoded: Leg/foot deficit	Dummy					recode
108	rdef4_r		Recoded: Dysphasia	Dummy					recode
109	rdef5_r		Recoded: Hemianopia	Dummy					recode
110	rdef6_r		Recoded: Visuospatial disorder	Dummy					recode
111	rdef7_r		Recoded: Brainstem/cerebellar signs	Dummy					recode
112	rdef8_r		Recoded: Other deficit	Dummy					recode
113	deficits	117	Total deficits	scale					compute deficits=sum(rdef1_r ... rdef8_r)

    New value-level information in data log file – computed or recoded:

1	Original Varname	Original value	Original label	Recoded varname	Recoded value	Label	Comment
2	RCONSC	D	drowsy	rconsc_r	2	drowsy	
3		F	fully alert	rconsc_r	1	fully alert	
4		U	unconscious	rconsc_r	3	unconscious	
5	SEX	F	female	sex_r	1	female	
6		M	male	sex_r	2	male	
129	DPLACE	A	home	dplace_r	1	home	
130		B	relatives home	dplace_r	2	relatives home	
131		C	residential care	dplace_r	3	residential care	
132		D	nursing home	dplace_r	4	nursing home	
133		E	other hospital departments	dplace_r	5	other hospital departments	
134		U	unknown	dplace_r	8	unknown	missing
135		blank	NA	dplace_r	9	not coded	missing
263	RDATE			ryear		STRING monthp year (A3).	
264				rmonth		COMPUTE dash = INDEX(rdate,'-').	
265						COMPUTE rmonthp = SUBSTR(rdate,1,dash-3).	
266						COMPUTE ryear = SUBSTR(rdate,dash+2).	

Common file transformations:

- Add variables from other dataset(s)
- Add cases from other dataset(s)
- CASESTOVARS and/or VARSTOCASES transformations

Adding variables to a data file

41. If the variables comprising a data file have been split across two (or more) physical files, and those files have a unique case-identifier variable in common, you can use MATCH FILES to add the variables from one data file to the other, so that you can do analyses using variables from both files together. MATCH FILES can match up to 50 files in one operation, as long as:
    a. All files are SPSS system files (.sav extensions)
    b. All files have the same unique case identifier variable (key variable):
        i. Case identifier names must be the same (including same case, ie upper/lower)
        ii. Case identifier variables must be the same type (string or numeric)
        iii. Case identifier variables must have the same width and number of decimals
    c. All files are sorted in the same order by the unique case identifier (Data > Sort Cases)
    d. Any duplicate variable names must be renamed or be excluded

42. To check whether the respondent identifier variable is unique in each file, select **Data > Identify Duplicate Cases** from the drop-down menus and run, with the respondent identifier variable as input, for each file (doing this with syntax is much more involved). The following output informs us that all 6,257 cases in the data set are primary cases (ie unique), and not duplicate records.

PrimaryLast Indicator of each last matching case as Primary

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	1 Primary Case	6257	100.0	100.0	100.0

43. Once you have sorted the files, and checked for unique key variables, the syntax for MATCH FILES is actually quite simple (see syntax file).

How many cases are there now in the data set? How many variables are there now in the data set?

Adding cases to a data file

44. Another common file transformation activity is to add cases to an existing data file. In this instance, the 'ist_corrected_uk_addvar.sav' dataset (which now contains additional variables) currently contains only UK stroke victims. In order to compare them with cases from other countries in the EU, we need to add additional cases from the trials in the EU countries as well.

45. Variables must match on several aspects in order to be 'paired' between the two files: (a) the variable names match is case sensitive, (b) both variables in both files must be the same type (string or numeric); string variables are flagged by '>', (c) variables must have the same width and number of decimals.

Once you are satisfied with the list of variables to be included in the new dataset, click on the 'OK' button. The new cases will be added to the active datafile ('ist_corrected_uk.sav'). Check the Output file, and the 'ist_corrected_uk.sav' file for errors or problems you did not anticipate.

How many cases are there now in in the data set? How many variables are there now in in the data set?

46. When SPSS does something unexpected, it is a good idea to check the the syntax against the SPSS manual to see what is happening, in case you missed or misinterpreted some defaults or options. As you can see from the syntax below, SPSS's default is to first rename all unmatched variables, and then DROP them from the resulting merged file.

Writing a raw data file
[Exercise 6]

47. As important as reading a data set into SPSS is writing it out in another format, either for use in a different statistical package, for long-term preservation, or both.

48. **Options**:
    a. Software packages such as **StatTransfer** convert data files among a wide variety of different software-dependent and generic formats (with/without syntax files). The Data Library has StatTransfer and can help University of Edinburgh students and staff with this.
    b. Software packages such as SledgeHammer, and Colectica will write a generic format data file (usually .csv) and a DDI-standard .xml metadata file. The Data Library has SledgeHammer, and can help with this also.
    c. Many statistical software packages can read in an SPSS system file (*.sav)
    d. Use SPSS menus: **File > Save as** to write tab- or comma-delimited (*.csv), or fixed field ASCII formats (in the latter case, ALWAYS use the TABLE subcommand and save the output)
    e. Use SPSS syntax to write out one of a number of other software-specific output formats (see Appendix E of the workshop handout)
    f. **Note**: SPSS no longer writes SPSS syntax files, so if you have used option d, you will need to write a syntax file 'manually'.

49. Use the **appropriate SPSS command** to write the output data file in an appropriate format (see Appendix E for a list of SPSS outfile commands and the formats SPSS can write).

    Generate the variable information using the DISPLAY DICTIONARY command. Save the variable and value lists (i.e. by saving the output window in an appropriate format).

    The most generic, non-controversial, and least error-prone data format (for reading into other software) is fixed-field format ASCII. This format can handle many file structures, including varieties of hierarchical files, and is not sensitive to commas and/or blanks embedded in variables. A popular format is .csv, which is more error-prone.

50. We can **use syntax** to write a fixed field format or a comma-delimited file. The advantage of using syntax is that you can easily switch recoded variables for original variables, and change the order of variables, rather than have all your recoded variables appear at the end of the data file. You can also use the TABLE option to have SPSS produce a list of the column locations to which variables are being written:

```
DISPLAY DICTIONARY.
WRITE OUTFILE="[path]\[filename].txt" TABLE / hospnum rdelay rconsc_r sex_r age rsleep ratrial rct
    to respno.
EXECUTE.
```

51. If you are writing a fixed-field ASCII data file (ie 'Fixed ASCII'), **make sure to EXPORT the output table** which indicates what columns the variables are written to, as well as the file information generated above – this is your only record of what variable is in what column(s), and what the values mean.

52. **Check your output.** Open the data file in a format neutral editor, such as Notepad++. Check the column number against the output table generated by SPSS to make sure that the final column numbers and case counts match.

53. Save the raw data file (file extensions such as .txt or .dat), the output file (in a software neutral format), and the file in which you have stored the variable and value lists. And finally, of course, update and save the Data log!

- SPSS data file (Data view/Variable View window)
  File > Save as > SPSS system file (.sav extension)
- SPSS Output viewer window
  File > Export as > (prefer text, html or PDF formats)
- SPSS syntax file
  File > Save as > SPSS syntax file (.sps extension)
- Data log file
  File > Save as > (appropriate format and extension)

Wrap-up

If you remember nothing else from this training session:

- Keep secure and backed up copies of the original data files (e.g. on your university network drive).
- You must always be able to backtrack through the versions of the data, therefore:
  - **NEVER, NEVER, NEVER** overwrite an existing variable when recoding or computing. **ALWAYS recode or compute to a new variable name.** Why? – because you WILL make mistakes.
  - **ALWAYS, ALWAYS, ALWAYS** save each version of the data file under a new name (ie **NEVER overwrite the old dataset**) after variable and file transformations.
    Why? – because you will very soon forget what you did to each file, especially if you didn't use and save the syntax/output file(s). Always keep your Data log file up-to-date – you will be grateful you did, in the long run.
- And, for support, Google is your friend (see also Appendix A).

And listen to the legendary Brian Clough:

Source: Gayle, V. (2016). *Producing Publication-Ready Graphs in Stata*.

Appendix A: Recommended online resources

IBM SPSS Statistics 22 manuals
http://www-01.ibm.com/support/docview.wss?uid=swg27038407

IBM SPSS Statistics 22 Command Syntax Reference
ftp://public.dhe.ibm.com/software/analytics/spss/documentation/statistics/22.0/en/client/Manuals/IBM_SPSS_Statistics_Command_Syntax_Reference.pdf

Raynald's SPSS tools: http://www.spsstools.net/

University of California, Institute for Digital Research and Education (IDRE) - Resources to help you learn and use SPSS: http://www.ats.ucla.edu/stat/spss/

Longitudinal Data Analysis for Social Science Researchers, SPSS training and workshop materials:
http://www.restore.ac.uk/Longitudinal/SPSS_support.html

The Analysis Factor's collection of articles explaining various features of SPSS:
http://www.theanalysisfactor.com/spss-syntax-101/

Selecting the right statistical analysis: http://statpages.info/#WhichAnalysis

Appendix B: Selected inter-system limitations on filenames, variable and value names etc [at time of writing]

Path (subdirectory) and file names: Different operating systems treat embedded spaces (blanks) in subdirectory and file names differently. Windows and MacOS allow filenames with embedded spaces, whereas these need to be surrounded by quotes in Unix/Linux operating systems.

- **Do not use** spaces or most other special characters in subdirectory and/or file names (eg 'variable list.xlsx').
- **Do use:** underscores ('variable_list.xlsx'), or CamelCase ('VariableList.xlsx').

Variable names

- In SPSS variable names must begin with a letter or the characters '@', '#' or '$', and names beginning with '#' or '$' have special functions (scratch and system variables). Variable names should not end in a full stop since this is a command terminator in SPSS.
- SAS variable names must begin with a letter, or an underscore '_'. Tab characters embedded in string variables are preserved in tab-delimited export formats.
- Special characters such as '@', '#' and '$' are not allowed in SAS variable names, the last three are replaced with underscores. In Stata, the only allowable characters are letters, numbers, and underscores.
- Case sensitivity: SAS will convert variable names 'mpage' and 'MPage' to 'MPAGE' for purposes of analysis (ie treat all 3 versions as one and the same variable), 'though not for purposes of display. In Stata, however, these are treated as 3 different variables. In SPSS, existing variable names are not case sensitive, while new variable names are.
- Variable names longer than 8 characters are truncated when exported to SPSS versions pre 12.0, SPSS .por files, SAS pre-V7, and Stata versions pre-7.
- **Note: it is generally recommended that variable names be no more than 8 characters**

Table: Maximum variable name length in four popular statistical packages

statistical package	variable name	variable label	value label
SAS 9.2	32 characters	256 characters	16 characters
SPSS 12+	64 characters Some procedures will display only first 8.	256 characters	120 characters Some procedures will display only first 20.
Stata 13+	32 characters	80 characters	32,000 characters
SDA 4.0+	32 characters	"no long"	any Truncates to first 16-20 for SAS/SPSS/Stata syntax files.

Other restrictions

- When writing files in Stata 5-6 or Intercooled 7-8 formats, only the first 2,047 variables are saved.
- All SPSS user-defined missing values are mapped to a system-missing value in SAS.
- Variable labels longer than 40 bytes are truncated when exported to SAS v6 or earlier.

Appendix C: Common file and variable transformations and their corresponding SPSS commands

	File transformations	SPSS syntax
-	Sort cases	SORT CASES
-	Sort variables	SORT VARIABLES
-	Transpose (cases and variables)	FLIP
-	Convert multiple records/case to one record per case with multiple variables	CASESTOVARS
-	'Flip the file'	VARSTOCASES
-	Merge – add cases	ADD FILES
-	Merge – add variables	MATCH FILES
-	Weight cases	WEIGHT
-	Split files	SPLIT FILE
-	Aggregate data	AGGREGATE
	Variable transformations	
-	Compute new variables	COMPUTE
-	Recode	RECODE
-	Rank cases	RANK
-	Random number generation	Transform > Random Number Generators
-	Count occurrences	COUNT
-	Shift values	SHIFT VALUES
-	Time series operations	CREATE RMV SEASON DATE SPECTRA

Appendix D: SPSS Syntax – Further information

You need syntax files when:

- You want to have the option of correcting some details in your analysis path while keeping the rest unchanged,
- You want to repeat the analyses on different variables or data files
- Some operations are best automated in programming constructs, such as IFs or DO LOOPs
- You want a detailed log of all your analysis steps, including comments (and didn't configure SPSS as above)
- You need procedures or options which are available only with syntax
- You want to save custom data transformations to use it them later in other analyses
- You want to integrate your analysis in some external application which uses the power of SPSS for data processing

Source: Raynald's SPSS tools <http://spsstools.net/en/syntax/>

For example, if you discover that SPSS has truncated some data fields when reading in the .csv file, you can save the syntax to read the data file to a syntax file, edit it to increase the size of individual fields, and rerun it:

- **Double R-click** in the Output window in the area of the syntax written by SPSS
- **Ctrl-C** to save the content of the yellow-bounded box around the output
- Menus: **File > New > Syntax** to open a new syntax file
- **Ctrl-V** to copy the content of the yellow-bounded box from the Output to the Syntax window
- **Edit the syntax file** to keep only the syntax text and run it.

The variable 'DSIDEX' in this data file is defined, based on the first 200 cases, as a 26 character string variable (A26). To increase the size of that variable to 50 characters, edit the syntax file to read 'DSIDEX A50'. Then rerun the syntax to read in the raw data file again: click and drag to select the syntax file contents, from the DATA statement down to and including the full stop '.' at the end of the file, or select **Edit > Select All**, and click on the large green arrowhead (the **'Run'** icon) on the SPSS tool bar to run it. Then of course, you will need to check the new file as discussed above.

Advantages to using syntax:

- a handful of SPSS commands/subcommands are available via syntax but not via the drop-down menus, such as **temporary, missing=include** and **manova**
- you can perform with one click all the variable recoding/checking and labelling assignments necessary for a variable
- you can run the same set of syntax (cut and paste or edit) with different variables merely by changing the variable names, and run or re-run it by highlighting just those commands you want to run, and then clicking on the **Run** icon.
- annotate the syntax file with COMMENTS as a reminder of what each set of commands does for future reference. COMMENTS will be included in your output files.

R**ules to remember about SPSS syntax**:

- Commands must start on a new line, but may start in any column (older versions: column '1').
- Commands must end with a full stop ('.').
- Commands are not case sensitive ie 'FREQS' is the same as 'freqs'.
- Each line of command syntax should be less than 256 characters in length.
- Subcommands usually start with a forward slash ('/').
- Add comments to syntax (preceded by asterisk '*' or 'COMMENT', and ending with a full stop) before or after commands, but not in the middle of commands and their subcommands.

- Many commands may be truncated (to 3-4 letters), but variable names must be spelled out in full.

Where do syntax files for reading in the data come from?

If you have collected your own data:

- You should write your own syntax file as you plan, collect and code the data.
- Some sites, such as the Bristol Online Surveys (BOS) site, will provide documentation as to what the questions and responses in your survey were, but you will have to reformat that information to SPSS specifications.

If you are doing secondary analysis, ie using data from another source:

- data from a data archive, should also be accompanied by a syntax file or be a system file which includes the relevant metadata
- If the data are from somewhere else, eg on the WWW, look to see if a syntax file is provided. If there is a SAS or Stata syntax file, it can be edited to SPSS specs.
- Failing a syntax file, look for some other type of document that explains what is in each variable and how it is coded. You will then need to write your own syntax file.
- And failing that, you should think twice about using the data, if you have no documentation as to how it was collected, coded, and what variables it contains, and how they are coded.

To generate syntax from SPSS:

a. If unsure about how to write a particular set of syntax, try to find the procedure in the drop-down menus
b. Many procedures have a **'Paste' button** beside the 'OK' button
c. Clicking on the 'Paste' button will cause the syntax for the current procedure to be written to the current syntax file, if you have one already open; If you do not have a syntax file open, SPSS will create one
d. Note: if you use the 'Paste' button, the procedure will not actually be run until you select the syntax (click-and-drag) and click the 'Run' button on the SPSS tool bar

Appendix E: SPSS read and write commands			
SPSS 19 commands to read file	**format**	**SPSS 19 commands to write outfile**	Note: "[fn]" = path and filename
data list file="[fn]" list or data list file="[fn]" free	comma-delimited file	save translate / outfile="[fn]" / type=csv	By default, both commas and blanks are interpreted as delimiters on input.
get data / type=oledb / file="[fn]"	database with Microsoft OLEDB technology		
get data / type=odbc / file="[fn]"	database with ODBC driver	save translate / connect=ODBC	
get translate / type=dbf / file="[fn]"	dBASE II files	save translate / outfile="[fn]" / type=db2 / version=2	
get translate / type=dbf / file="[fn]"	dBASE III files, dBASE III Plus	save translate / outfile="[fn]" / type=db3	
get translate / type=dbf / file="[fn]"	dBASE IV files	save translate / outfile="[fn]" / type=db4	
get data / type=xls / file="[fn]"	Excel 2.1 (pre-Excel 95)	save translate / outfile="[fn]" / type=xls / version=2	
get data / type=xlsm / file="[fn]"	Excel 2007 or later macro-enabled workbook		
get data / type=xlsx / file = "[fn]"	Excel 2007 or later workbook	save translate / outfile="[fn]" / type=xls / version=12	
get data / type=xls / file="[fn]"	Excel 95	save translate / outfile="[fn]" / type=xls / version=5	
get data / type=xls / file="[fn]"	Excel 97 thru Excel 2003 files	save translate / outfile="[fn]" / type=xls / version=8	
get translate / type=slk / file="[fn]"	Excel and Multiplan in SYLK format	save translate / outfile="[fn]" / type=slk	
get translate / type=xls / file="[fn]"	Excel pre-5	save translate / outfile="[fn]" / type=xls / version=2	
get translate / type=wk / file="[fn]"	Lotus 1-2-3 file, any		
get translate / type=wks / file="[fn]"	Lotus 1-2-3 release 1A	save translate / outfile="[fn.sys]" / type=wks	
get translate / type=wk1 / file="[fn]"	Lotus 1-2-3 release 2.0	save translate / outfile="[fn.sys]" / type=wk1	
file handle [nickname] / mode=multipunch. data list file=”[nickname]”	multipunched raw data (aka column binary)		
data list file="[fn]" fixed	raw data, inline or external file, fixed format	write outfile="[fn]" table. Execute.	
data list file="[fn]" list or data list file="[fn]" free	raw data, inline or external file, freefield format (blank or comma delimited)	write outfile="[fn]"	By default, both commas and blanks are interpreted as delimiters on input.

input program - data list - repeating data - end input program or file type - repeating data - end file type	raw data, input cases with records containing repeating groups of data		
file type - record type - data list - end file type	raw data, mixed, hierarchical, nested files, grouped files		
matrix data	raw matrix data, including vectors		
file handle [nickname] name="[fn]" followed by get file "nickname"	record length >8,192, EBCDIC data files, binary data files, character data files not delimited by ASCII line feeds	file handle [nickname] name="[fn]" followed by write outfile="[nickname]	
get sas data="[fn]"	SAS dataset version 9	save translate / outfile="[fn]" / type=sas / version=9	
get sas data="[fn].sd2" (or Unix: [fn].ssd[nn])	SAS dataset version 6	save translate / outfile="[fn]" / type=sas / version=6	
get sas data="[fn].sd2" (or Unix: [fn].ssd[nn])	SAS dataset version 7 with file of value labels	save translate / outfile="[fn]" / type=sas / version=7 / valfile="[fn]"	
get sas data="[fn].sas7bdat"	SAS dataset version 7 with file of value labels	save translate / outfile="[fn]" / type=sas / version=7 / valfile="[fn]"	
get sas data="[fn].dat"	SAS transport file	save translate / outfile="[fn]" / type=sas / version=X	
get data / type=txt / file="[fn]"	similar to DATA LIST, does not create temporary file	save translate / outfile="[fn]" / type=csv	
get file="[fn].sys"	SPSS /PC+	save translate / outfile="[fn].sys" / type=pc	
import file="[fn].por"	SPSS portable	export outfile="[fn].por"	
get file="[fn].sav"	SPSS system file	save outfile="[fn]" or xsave outfile="[fn]"	
get stata file="[fn].dta"	Stata-format data files version 6	save translate / outfile="[fn]" / type=stata / version=6	
get stata file="[fn].dta"	Stata-format data files version 7	save translate / outfile="[fn]" / type=stata / version=7	
get stata file="[fn].dta"	Stata-format data files version 8	save translate / outfile="[fn]" / type=stata / version=8	
get stata file="[fn].dta"	Stata-format data files versions 4-5	save translate / outfile="[fn]" / type=stata / version=5	
get translate / type=wk / file="[fn]"	Symphony file, any		
get translate / type=wrk / file="[fn]"	Symphony release 1.0	save translate / outfile="[fn]" / type=sym / version=1	
get translate / type=wr1 / file="[fn]"	Symphony release 2.0	save translate / outfile="[fn]" / type=sym / version=2	
get translate / type=sys / file="[fn]"	Systat data file		
get translate / type=tab / file="[fn]"	tab-delimited ASCII file	save translate / outfile="[fn]" / type=tab	Embedded tabs are interpreted as delimiters
get capture			Obsolete, use get data