Codebook cookbook

A guide to writing a good codebook for data analysis projects in medicine

1. Introduction

Writing a codebook is an important step in the management of any data analysis project. The codebook will serve as a reference for the clinical team; it will help newcomers to the project to rapidly have a flavor of what is at stake and will serve as a communication tool with the statistical unit.

Indeed, when comes time to perform statistical analyses on your data, the statistician will be grateful to have a codebook that is readily usable, that is, a codebook that is easy to turn into code for whichever statistical analysis package he/she will use (SAS, R, Stata, or other).

2. Data preparation

Whether you enter data in a spreadsheet such as Excel (as is currently popular in biomedical research) or a database program such as Access, there is much freedom in the way data can be entered. A few rules, however, should be followed, to make both the data entry and subsequent data analysis as smooth as possible. A specific example will be presented in Section 3, but first let’s look at a few general suggestions.

2.1 Variables names

A unique, unambiguous name should be given to each variable. Variables names MUST consist of one string only, consisting of letters and — when useful — numbers and underscores (_). Spaces are not allowed in variables names in most statistical programs, even if data entry programs like Excel or Access will allow this. It is good practice to enter variables names at the top of each column. Variables names should be long enough to be meaningful, but short enough to be easy to handle; variables labels (below) are the place to clarify what is coded in each variable more substantially.

2.2 Variables labels

A label is a description of the variable, such as a textual description or a reference to the question number, if the item arises from a questionnaire.

It is important to include a descriptive variable label for each variable in the file. In practice, these can be added to the existing data base or as a listing in a text file separate from your data file.

Variables labels are important not only for the statistician to understand the contents of each data item, but also to the researchers, as the labels will facilitate understanding of the outputs of the statistical analysis, which they must interpret.

Examples of variable labels will be presented in Section 3.1.1.

2.3 Variables codes

Each categorical variable should have a set of exhaustive, mutually exclusive codes. These codes should be thoroughly documented in the codebook. Where possible, standard data codes should be used (e.g. 0=no, 1=yes for yes/no variables): the use of such standards facilitates the comparison of results across variables, or even across studies.

Examples of variable codes (sometimes called value labels) will be presented in Section 3.1.2.

2.4 Variables formats

Data should be reduced to numeric codes whenever possible. This avoids the occurrence of typographical errors in entering literal answers, leading to misinterpretation of two equivalent answers as being different, and above all greatly facilitates the use of the variables in statistical models, as most statistical packages cannot use character variables (that is, variables taking alphanumeric values, e.g. gender=F or M) directly in statistical models. (In fact, the alphanumeric variables can be used, but only after some preprocessing that can be time consuming and hence expensive.)

Character entries should be used for descriptive purposes only, e.g. in comment variables — reason for withdrawal from experiment, reason for non-adherence to medication, etc — and even in these cases, only when the number of possible answers is large, so that creating numeric codes for all possibilities becomes too cumbersome.

If for some reason you still prefer to enter alphanumeric values (we can imagine that sometimes it saves some time and/or is less error-prone at data entry), we will still suggest a few hints in Section 5 to reduce problems further down the line, at the statistician’s end.

2.5 Missing data

In practice, missing data almost always occur in clinical projects; they can arise from many sources, such as refusal to answer, omission, missing by design, etc. The coding of missing data often does not receive special attention, which is often benign; indeed, cells left empty in Excel spreadsheets, for example, will be interpreted as missing data by most statistical programs.

It is sometimes important however — either in data analysis or when writing reports — to be able to distinguish between different types of missing data, and that will require some coding. Furthermore, specifically coding missing values in a data bank makes it clear that the data item is truly missing, as opposed to an omission by the data entry person, which is unfortunate but happens in practice; if true missing values were coded, identifying data entry omissions will be trivial and going back to charts (or questionnaires) — when possible — will be a foreseeable option to minimize the amount of missing information.

Section 3.2 will suggest a way to code missing values.

2.6 Date variables

Date variables are often problematic, as there is no uniform standard for date formats. Therefore, it is important to choose a standard and keep the same date format for each record (subject, patient) for all date variables in the project.

In Section 3.4, we will suggest a sensible and easy to use date format.

3. Entering your codebook through Excel spreadsheets

Very often, statisticians are sent data files that contain all the information listed in Section 2, but either through inconvenient codebooks (e.g. written in Word tables that are hard to access electronically by other programs, leading to "cut and paste" coding which needlessly duplicates efforts) or data file columns headers. Even though all the information necessary for data coding and analysis may be included, the information is so sparse and not easily accessed that it is not readily usable. It is much more efficient to include all the necessary information in a format that can be read in an automated way. The examples in Section 3.1 will illustrate a programmer-friendly codebook that remains easy to prepare.

For example, when SAS — perhaps the most commonly used statistical package for data coding — reads an Excel file and tries to obtain the variable names from the top row of the file, it expects a single string (no space, no bracket, no equal sign: nothing but a word!); consequently, it converts the content of top row cells by replacing unacceptable characters into underscores, shortens lengthy names and adds numeric suffixes — when necessary — to keep them unique; this often results in non-optimal and often hard to comprehend variables names, as shown in the examples in Table 1.

Original Excel column header	Variable name, when converted by SAS

ETHORIG (1=Caucasian) 2=Black 3=Asian 4=Natives	ETHORIG__1_Caucasian__2_Black_3_
SLICMAX.LM (total damage score)	SLICMAX_LM__total_damage_score_

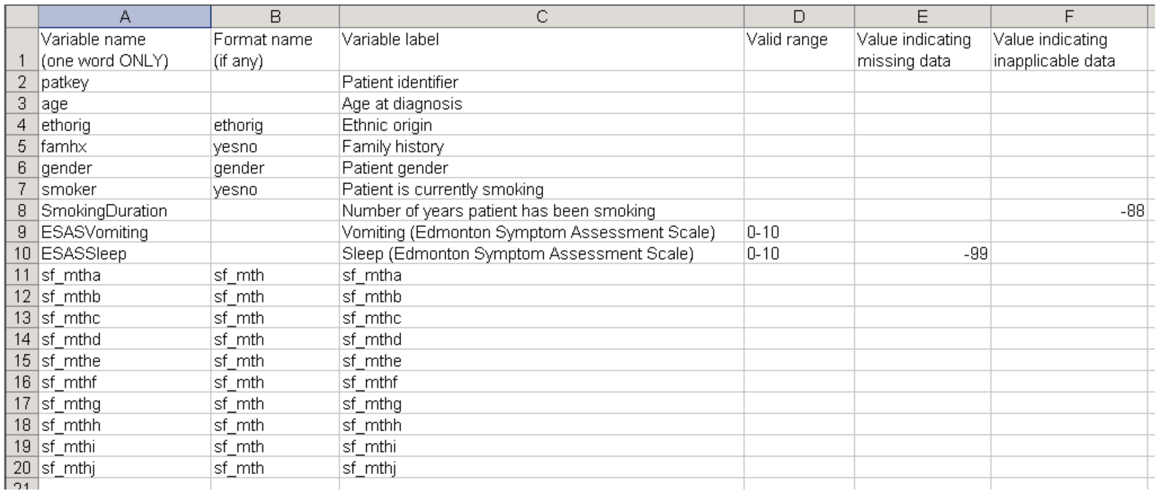
Table 1. Choosing friendly variables names.

The two shorter and yet meaningful alternatives **ethorig** and **SlicMax_LM** would have been accepted by SAS without any changes, meaning that the original codebook could be used when deciphering the outputs from the analyses.

These examples illustrate why keeping variable names simple is always a good idea! The additional bits of information (e.g. codes 1=Caucasian, 2=Black, etc. for the first variable, and text 'total damage score' for the second) should not be entered in variable names, but rather belong in value labels and variable labels, respectively, as will be illustrated shortly.

3.1 Labels

3.1.1 Variable labels



	A	B	C	D	E	F
1	Variable name (one word ONLY)	Format name (if any)	Variable label	Valid range	Value indicating missing data	Value indicating inapplicable data
2	patkey		Patient identifier			
3	age		Age at diagnosis			
4	ethorig	ethorig	Ethnic origin			
5	famhx	yesno	Family history			
6	gender	gender	Patient gender			
7	smoker	yesno	Patient is currently smoking			
8	SmokingDuration		Number of years patient has been smoking			-88
9	ESASVomiting		Vomiting (Edmonton Symptom Assessment Scale)	0-10		
10	ESASSleep		Sleep (Edmonton Symptom Assessment Scale)	0-10	-99	
11	sf_mtha	sf_mth	sf_mtha			
12	sf_mthb	sf_mth	sf_mthb			
13	sf_mthc	sf_mth	sf_mthc			
14	sf_mthd	sf_mth	sf_mthd			
15	sf_mthe	sf_mth	sf_mthe			
16	sf_mthf	sf_mth	sf_mthf			
17	sf_mthg	sf_mth	sf_mthg			
18	sf_mthh	sf_mth	sf_mthh			
19	sf_mthi	sf_mth	sf_mthi			
20	sf_mthj	sf_mth	sf_mthj			

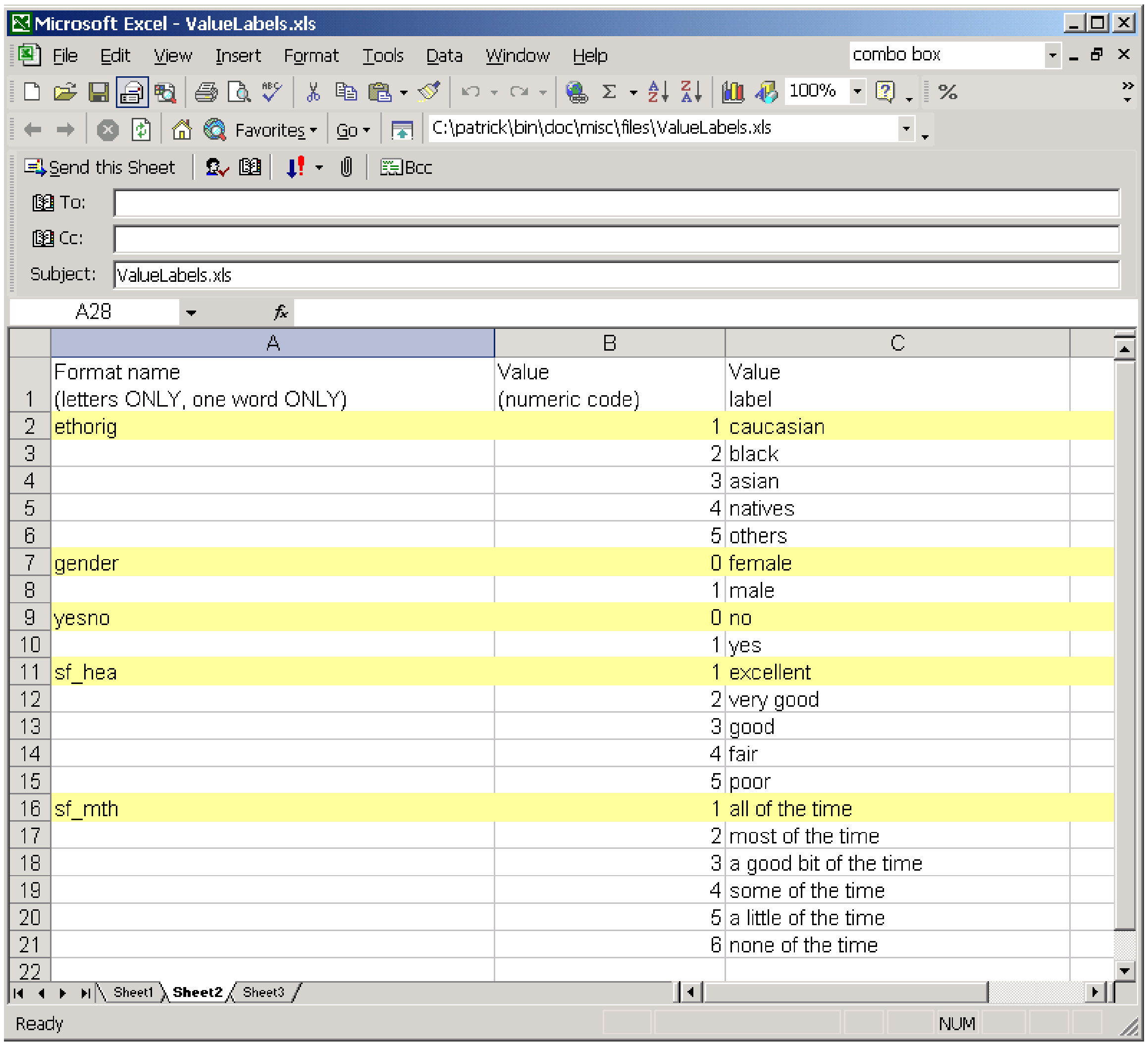
Figure 1. Variables names, labels and formats.

The variable labels section (or maybe sheet, if you use Excel and prefer to keep all information in a single file) of your codebook should contain the following three bits of information: variables names (identical to that used in the data entry page), format names and variable labels (see columns A, B and C, in Figure 1). Feel free to enter additional information in subsequent columns (e.g. measurement units — though measurement units could also be included in variable label, valid ranges for data, values for missing and inapplicable data, etc.)
Categorical variables that were entered with numeric codes should be identified as such through the means of an associated format name (column B, in Figure 1). Each format can be used more than once; in Figure 1, for example, variables *famhx* and *smoker* were

both yes/no variables, thus they were both coded as 0/1 variables, a format called *yesno*, yet to be described in the value labels section.

3.1.2 Value labels

The value labels section should consist of three columns: formats names and, for each format, an exhaustive list of numeric values and corresponding value labels.


Microsoft Excel - ValueLabels.xls
File Edit View Insert Format Tools Data Window Help
combo box
100%
Favorites Go
C:\patrick\bin\doc\misc\files\ValueLabels.xls
Send this Sheet
Bcc
To:
Cc:
Subject: ValueLabels.xls
A28
Sheet1 Sheet2 Sheet3
Ready
NUM

	A	B	C
1	Format name (letters ONLY, one word ONLY)	Value (numeric code)	Value label
2	ethorig	1	caucasian
3		2	black
4		3	asian
5		4	natives
6		5	others
7	gender	0	female
8		1	male
9	yesno	0	no
10		1	yes
11	sf_hea	1	excellent
12		2	very good
13		3	good
14		4	fair
15		5	poor
16	sf_mth	1	all of the time
17		2	most of the time
18		3	a good bit of the time
19		4	some of the time
20		5	a little of the time
21		6	none of the time
22			

Figure 2. Formats names and values labels.

Column headers were added to both values labels and variables labels sheets for the sake of clarity and ease of reference, but are not in fact absolutely necessary.

The order of the first three columns in each of these sheets, however, should be respected to ease later programming. Feel free to fill additional columns either for your personal use or for the statistician's team, in which case you will clearly indicate their intended use by either labeling them properly or by getting in touch with the statistical team.

Your value labels sheet may be conveniently reused in future projects, at least in part (as some categorical variables tend to be present in most studies, e.g. gender, smoking, yes/no variables, etc.).

3.2 Coding missing and inapplicable data

It is not absolutely mandatory to code missing values, as pointed out in Section 2.5. Nevertheless, such codes should be used as much as possible (it is a trade-off between work load and data bank completeness and clarity, the latter taking precedence whenever feasible).

When coded, missing values for categorical variables should be identified by a code *radically* different from non missing values: for example, a variable that would be coded as 0='not at all', 1='a little bit', …, 5='very much', a sensible choice would be to take a negative value, e.g. -9.

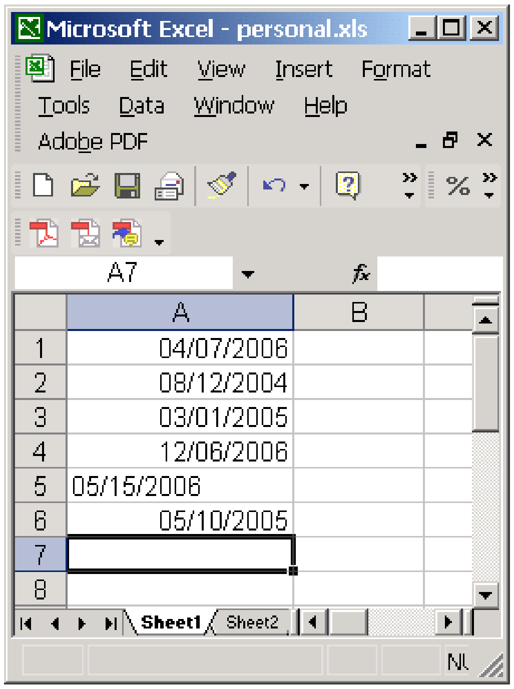
Similarly, for continuous variables (e.g. age as expressed in fractional years), impossible values should be used, diminishing the possibility of erroneously entering such codes: -88, -99, -999 are values that are often used. The choice is yours, but whatever values you choose to use for inapplicable data should be documented in your codebook (see columns E and F, Figure 1).

Note also that inapplicable (e.g., pregnancy related data among males) data should be coded differently from missing data. Codes for missing and inapplicable data can be entered in the variable labels worksheet for unformatted variables (e.g. age) or in the value labels worksheet if all variables using a given format were identically coded — which is highly recommended, of course!

4. Entering date variables

Date variables are often the cause of nightmares! Indeed, there is no standard way to enter dates; some prefer mm/dd/yyyy, as in 02/28/2007, others prefer dd/mm/yyyy, as in 28/02/2007, and enter their data accordingly, often without telling Excel in which format they are entering the corresponding variable. Excel is clever enough to recognize it's a date, but by default Excel will interpret them as dd/mm/yyyy.

Consequently, if you enter the dates April 7, 2006, August 12, 2004, March 1st, 2005, December 6, 2006, May 15, 2006, May 10, 2005 as in Figure 3, it looks like Excel accepts and understands all the dates you have entered. However, the value in row 5 being left-aligned while others were right-aligned should ring a warning bell!



	A	B
1	04/07/2006	
2	08/12/2004	
3	03/01/2005	
4	12/06/2006	
5	05/15/2006	
6	05/10/2005	
7		
8		

Figure 3. A date variable with incorrect date format.

Indeed, as Excel expects dates in the format dd/mm/yyyy (by default), it accepted the first four values (but misinterpreted them as July 4, 2006, December 8, 2004, January 3rd, 2005, June 12, 2006) and the date on row 5 could not be read as a date (month 15?!?!?), and was therefore aligned to the left and recorded as a string rather than a date!

To avoid such misinterpretations — which will remain problematic when your statistician reads it with a statistical software package — we highly suggest that prior to entering any dates, you indicate (in Excel or Access, for example) a format date to be used for the column where you intend to enter the dates. Furthermore, we recommend that you use a format that is unequivocal, e.g. the format dd-mon-yyy, that is, a format where the month will be displayed in words; it will not be more work for you as you can still enter your dates by entering month as numeric and using slashes between day & month and month & year: every time you enter a new date, Excel will turn it into the chosen format, and you will see right away whether it was correctly interpreted or not! Easy to use and much less error-prone than other formats! The best of all worlds!

To use that format, click the cell on top of the column, browse through **Format**, **Cells**, click **Date** in the "**Category:**" list box and finally click the format "14-mar-01" (or anything alike) in the "**Type:**" list box.

4.1 Validating date values

Avoiding date entry errors in Excel is made easier with the Data Validation tool.
If you enter your data in an Excel worksheet, click the top cell of the column where you plan to enter a date and follow these steps:

1. On the **Data** menu, click **Validation**, and then click the **Settings** tab.
2. In the **Allow** box, click **Date**.
3. Enter minimum and maximum feasible dates.

5. Useful Excel tips: AutoComplete and Pick from List

As announced in Section 2.4, we now suggest a few ways to enter alphanumeric variables (in Excel) that will reduce the risk of typographical errors and subsequent inconveniences (or worse!).

An Excel feature that makes entering data easier is **AutoComplete**. With this tool, all you need to do is to type in the first letters or digits of an entry into a cell; Excel will automatically scan the surrounding values in that column and complete the entry for you if it finds a like value. If you want to accept **AutoComplete**'s suggestion, simply press **Enter**, **Tab**, or any of the arrow keys. If the **AutoComplete's** suggestion is not what you need, keep entering the desired text, or use the delete key to cut off the letters that it suggested. You can toggle the **AutoComplete** feature on and off by choosing **Options** from the **Tools** menu and clicking on the **Edit** tab. Clicking in the **Enable AutoComplete for Cell Values** checkbox will allow you to turn **AutoComplete** on and off.

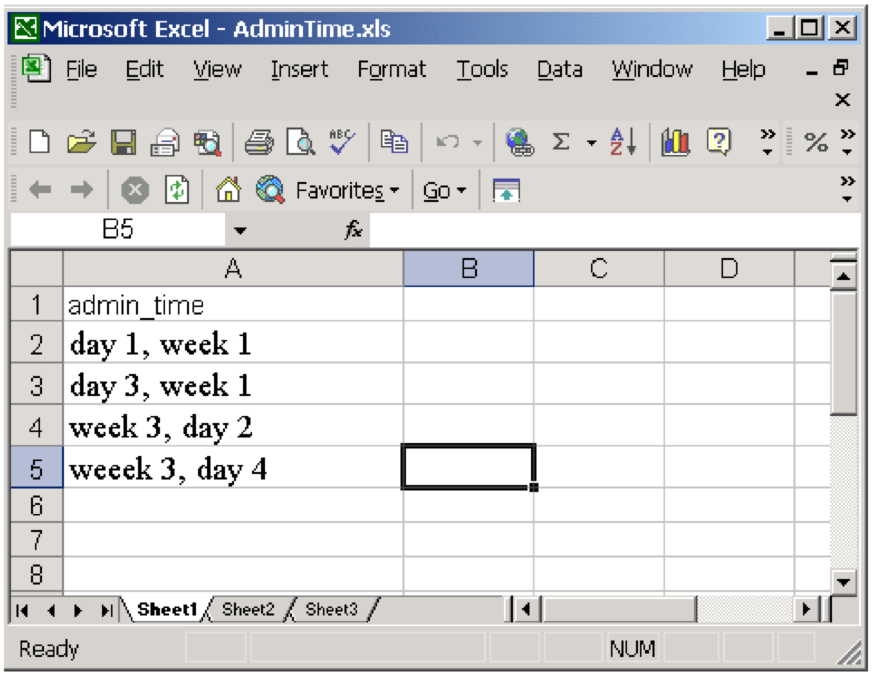
Pick From List could also be useful. It allows you to access a range of values that have already been entered into a given column and to pick a value from that list.
Right-click on the next cell to be filled and choose **Pick From List...** from the pop-up menu: it will produce a drop-down list below the cell showing the values already entered in the column; click a value to select it or type the **Esc** button to close the list.

A categorical variable can also be entered though a combo box. This works similarly to **Pick From List**, except that values offered in the list will come from a pre-determined list, not from the values already entered into the column. Choose the **Microsoft Excel Help** item from the **Help** menu, and enter "*Combo box*" in the text box; in the list of items found by the search tool, click the item "**Enter data in cell from a list you specify**" and follow the instructions.

6. Ready-to-use information

This is a minor but still noteworthy point: sometimes, one tries to compress information into as few variables as possible. Splitting information into more than one variable, however, is sometimes more natural and makes data entry easier (not to mention easier for data coding and analysis!).

Consider the example below, where a questionnaire was administered several times to each patient, with a variable called *admin_time* indicating the time of questionnaire administration for the corresponding record. The variable illustrated in the worksheet in Figure 4 contains all the information, but in a way that is not readily usable by the programmer.

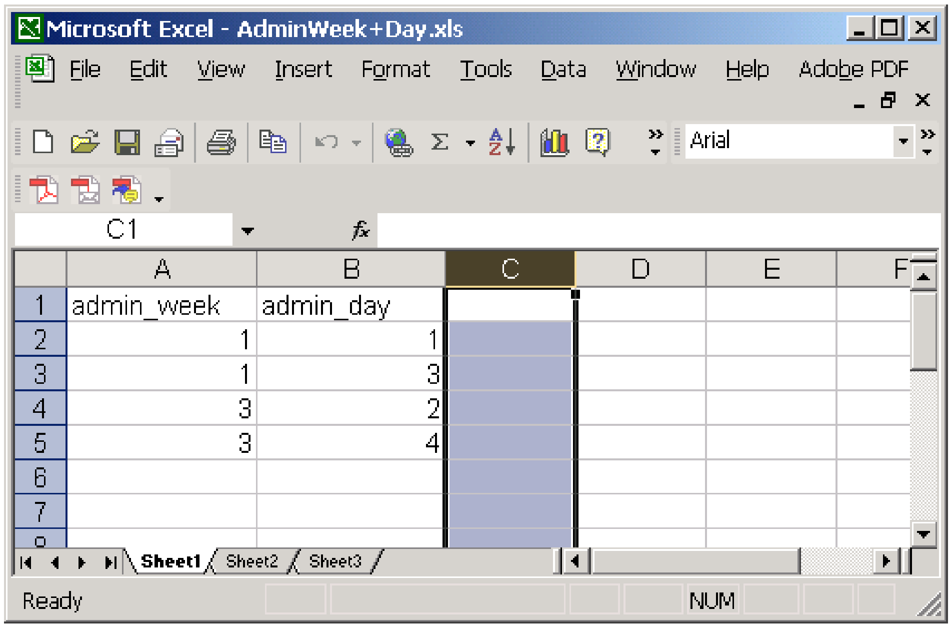

Microsoft Excel - AdminTime.xls
File Edit View Insert Format Tools Data Window Help
Favorites Go
B5
Sheet1 Sheet2 Sheet3
Ready
NUM

	A	B	C	D
1	admin_time			
2	day 1, week 1			
3	day 3, week 1			
4	week 3, day 2			
5	weeek 3, day 4			
6				
7				
8				

Figure 4. A variable containing information difficult to access through programming.

A good programmer will still be able to get the day and week numbers relatively easily from the lines 2 to 4: indeed, one would take the number after the string ‘day’ as the day number and the number after ‘week’ as the week number. Nothing impossible to deal with here! But consider now the 5th row, which unfortunately includes a typo (week was misspelled). The algorithm discussed above would fail in finding the word ‘week’, and week number would thus not be read!

The lack of a standard (e.g., always day first and week second, in this example) by itself is a good point against entering non numeric data, as suggested in Section 2 — not accounting for the augmented risk of typos. In this example, entering time of questionnaire administration into two columns, as shown in Figure 5, does better, both in terms of ease of data entry and statistical coding.


Microsoft Excel - AdminWeek+Day.xls
File Edit View Insert Format Tools Data Window Help Adobe PDF
Arial
C1
Sheet1 Sheet2 Sheet3
Ready
NUM

	A	B	C	D	E	F
1	admin_week	admin_day				
2	1	1				
3	1	3				
4	3	2				
5	3	4				
6						
7						

Figure 5. Information made easy to read and interpret.

7. A program to convert Excel codebook file to SAS code

An executable program to convert Excel codebook files to SAS code is available at http://www.med.mcgill.ca/epidemiology/Joseph/PBelisle/ExcelCodebook2SasCode.html

8. Final words

We believe that following these guidelines will greatly facilitate data entry and subsequent statistical analyses, not to mention cutting down on errors, which can often go undetected through the life of a project. Of course, we have presented our preferences, and others may have different opinions or habits. Therefore it is always best to consult with your statistical group prior to beginning any data base programming, and to devise your data entry system ahead of time in collaboration with them.

Nevertheless, we hope that you find this codebook cookbook useful, and welcome your comments for improvements.

Authors: Patrick Bélisle and Lawrence Joseph
Department of Clinical Epidemiology / McGill University Health Centre
Montréal, Québec CANADA
patrick.belisle@rimuhc.ca
http://www.med.mcgill.ca/epidemiology/Joseph/PBelisle
http://www.med.mcgill.ca/epidemiology/Joseph/