

Creating CSV Files

Recording Data in Spreadsheets

Use spreadsheet software to record and organize your data as tables. Do data entry using the spreadsheet; do analysis using R.

You will not need to use the fancy formatting capabilities of the spreadsheet software; just enter the data as plain text as in Figure ?? . Because of this, any spreadsheet package will do — Excel and Google docs are popular.

At the same time as you plan and then enter data in the spreadsheet, you should record documentation — the codebook — in a text file. Any word processor will do.

Coding The Variables

Once you have decided what variables to record, you need to decide how to **code** them, that is, how to represent the measurement. Sometimes the coding is obvious, but always it is worthwhile thinking about things before jumping in. variables!coding

Try to use mnemonic, common-sense codes whenever possible. For instance, M and F are sensible codes to record the sex of a subject. If there is an official code that is readily available, use that: Postal codes (e.g., IL, NY, MN), zip codes, airport codes, and so on. If there is a clear name — e.g., languages, species name if you are studying living things, or a political party if you are studying elections — you can use the name itself as the code. Make sure that you spell names consistently. In R, English and english and english are all different names. variable names names!of variables capitalization!in R

The airfare project involves five variables: distance, price, airline, origin, and destination.

It seems obvious to record the distance and price as numbers. Of course it would be silly to record some distances in miles and others in kilometers, or some prices in \$US and others in euros. You need to standardize.

For the origin and destination, the obvious thing might be to record the city name, e.g., Chicago or New York. But this leads to ambiguities when there is more than one airport serving a city. A better choice is the official airport code, a unique three-letter designation assigned to each airport. If, later on, you decide that the city name is more meaningful, you can always translate from airport code to city name.

For airlines, you can just use the name of the airline. You could also use an abbreviation, for instance NWA for Northwest Airlines. Since there might be many airfares for each airline, it's important to make sure that whatever representation you use, it is used consistently; you don't want to mix the use of "NWA," "Northwest," "Northwest Airlines," etc. Even though all of these are the same to a human reader, then computer would treat them as distinct entities.

	A	B	C	D	E	F
1	origin	dest	price	dist	airline	
2	ORD	MSP	172	410	United	
3	ORD	MSP	174	410	NWA	
4	ATL	CLT	535	246	NWA	
5	MSP	DWS	192	1568	NWA	
6	MSP	DWS	275	1568	United	
7	MSP	DW			US Air	
8						
9						
10						
11						

Figure 1: Filling in a table with the airfare data. Variable names are in the top row.

Naming the Variables

You need to pick a name for each variable — each column of the table — so that you can refer to that variable during your later analysis of the data. No two variables in one table should have the same name. It's convenient if the name is easy to type and reminds us what the variable is about. It simplifies the analysis phase — done in R — if the variable name is a valid object name in R. That is, there shouldn't be punctuation or spaces in the name. Stick to groups of letters and numbers for the names.

I'll illustrate with data as might be collected to study airline prices and the way they relate to the distance travelled, the origin and destination of the flight, etc. I'll use the obvious names: `dist`, `price`, `dest`, `origin`, `airline`. You could have used "destination" rather than "dest," and "distance" rather than "dist." This is a matter of personal preference.

Making the Table

The steps are simple:

1. Enter the variable names on the top row, one name in each column. Don't leave any blank columns.
2. Enter the data for each case as one row.
3. Save the spreadsheet.

Text, such as the name of the airline, should be entered straight; there's no need for quotation marks even when there are spaces. Avoid including any commas in your data.

There is no typographical formatting: no cells with color backgrounds, no bold-face or italics fonts.

Sometimes as you enter data, you discover that you need to record some notes, perhaps to indicate some data that are entered provisionally. A convenient way to do this is to add a column, perhaps with a heading of “comments.” You can then enter the notes as free-form text (but don’t use commas.) But if you find yourself entering notes for every case, such as the name of the person who collected the data for that case, perhaps you want to treat the information as another variable and code it appropriately.

Ordinal Data ordinal data data!ordinal coding!ordinal data ordinal data!and codebook

Some data, even if they are not numerical, have a natural ordering. For example, surveys often ask about the educational level of respondents, recording answers such as “not HS grad,” “HS grad,” “some college,” “college grad,” “graduate school.” It’s reasonable to say that a “HS grad” has a level of education between “not HS grad” and “some college.” This information about ordering can be valuable when analyzing data.

When coding data with a natural order, make sure to indicate the order in the codebook. It might be tempting to code the data as numbers, for instance coding “not HS grad” as 1, “HS grad” as 2, and so on. Avoid this temptation. It will be hard for you to remember the code and you will therefore make mistakes. Instead of recording the ordinal data with numerical codes, record it as short text labels and write down the information about the order in the documentation file. During the analysis process you can apply the information about the ordering.

Missing Data It often happens that data for a variable for one or more cases is missing. This might happen because the measurement could not be made or because a typographical or other error was made when originally recording the data and the true measurement cannot be recovered.

missing data!coding for NA!code for missing data coding!missing data

Such missing data should be explicitly identified as such. An effective way of doing this is to enter NA as the value for the missing data in the spreadsheet. If NA happens to be an actual code for non-missing data, then select another code for missing data, such as missing. Never use a missing data code in one variable that is a valid level in another variable: doing so makes it hard to read the data into R in a valid format.

Don’t confuse the idea of data that is missing with data that take a form different from your expectations. If you are doing a survey, a person who answers, “I don’t know,” or “I won’t tell you,” is actually providing information to you. Perhaps you want to code such data with DK or REFUSE. Save the NA designation for data that are genuinely missing, such as when the survey subject gives you an answer but you forgot it before writing it down.

When you read a table in to R, R will automatically treat the NA code as standing for missing data. missing data NA for missing data Section ?? describes what to do if you use a code other than NA for missing data.

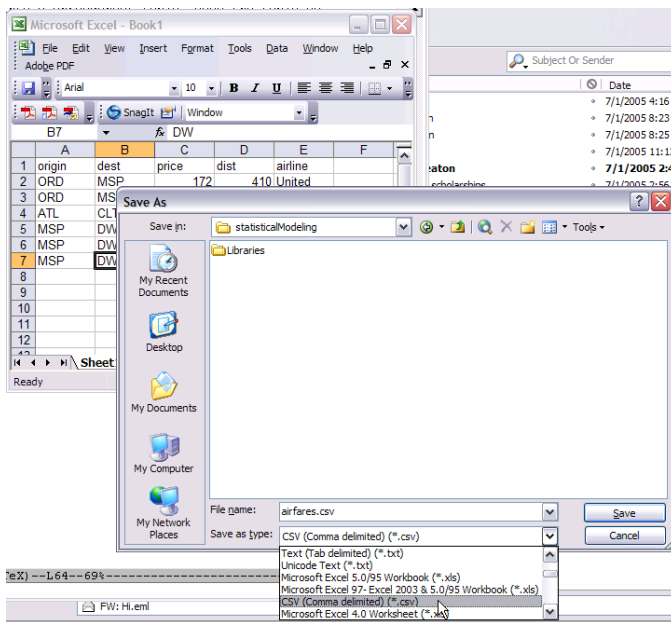


Figure 2: On a Windows computer, using the FILE/SAVE AS command to save the spreadsheet as a CSV file.

Saving the Spreadsheet

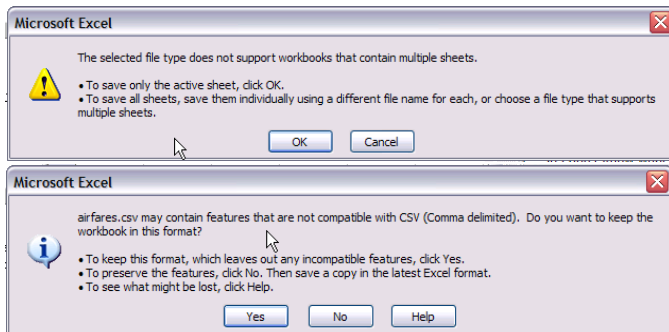
CSV file!creating CSV!advantages spreadsheet!propriety formats Once your data are entered into a spreadsheet table, save the table as a file. Usually the spreadsheet software will have a default file type that reflects the commercial imperatives of the software company to build and retain market share by creating incompatibilities with other software. Such proprietary formats are not used in this course. Instead, you should override the default file format, saving the data in a simple format that is compatible with a wide range of software packages.

spreadsheet!CSV format For the data used in this book are stored in the “comma-separated values” (CSV) file format. In this format each row of the table is written to a plain text file as a single line. The entries in different columns are separated by commas. The name of a CSV file is typically made to end with the .csv file-name extension, for example, `airfares.csv`.

An advantage of the CSV format is that it can be read by a wide variety of software, including R. CSV is pure text — no formatting. And while many spreadsheet programs allow you to have multiple “sheets” of data, a CSV file will store only one sheet. In practice, this is no limitation at all, since you can use multiple files to save multiple tables.

Excel spreadsheet XLS format!use CSV The process of saving an EXCEL spreadsheet

as a CSV-formatted file is shown in Figure ?? . When you save a spreadsheet into CSV format, you may be warned by the software that you can save only one sheet and that formatting information — such as fonts and background colors — will be lost. The messages might look like these:



Just say yes.

The Codebook

In addition to the spreadsheet file containing the table, it's important to create another file with a codebook for the table.

A **codebook** is simply the documentation of the conventions used in storing and coding your data. It often contains a short description of each variable. The point of having a codebook is to be able to remind ourselves and communicate with others what the variables mean and how they are stored. Many scientists are used to recording information in a lab notebook. It may be that much of the codebook consists of excerpts from the lab notebook.

The codebook can be an ordinary computer file — a text file or a word-processor document — that is stored along with the table in a separate file. Here's what a simple codebook for the airfare table might look like:

Codebook for airfares.csv file

dist — the geographic, direct distance from the origin city to the destination, measured in miles.

price — the price of the round-trip ticket, measured in dollars

orig — the official airport code for the originating airport. These are documented at <http://www.world-airport-codes.com>

dest — the official airport code for the destination airport

airline — the name of the airline.

Debugging Your Data

You will make mistakes collecting data and entering those data into a spreadsheet. The people from whom you get data will also make mistakes. Before you can properly analyze your the data stored in your table, you need to make sure that it reflects accurately the data you intended to record. Many of the tools for debugging your data are the same statistical tools you will use to analyze data; think of spotting bugs in the data as doing a mini-statistical analysis. Here are some basic ways to avoid “mechanical” errors such as transcription mistakes.

Three basic problems cover most of the difficulties neophytes have had importing newly created spreadsheet files.

1. Can't import the file at all, perhaps because the file is missing or isn't where it's supposed to be, or perhaps because it is formatted in a substantially different way than is required.
2. For categorical variables, there are mistakes in the levels of the variable, perhaps due to miscoding or typographical errors.
3. For quantitative variables, the values aren't properly treated numerically when read in to R.

Missing File or Bad Formatting

The `read.csv` and `read.csv` and `ISMdata` programs used for importing the data from a CSV file into R were described in section ???. Once you have saved your spreadsheet into a CSV file, open up the R program and import the data, for instance with

```
> a = ISMdata()
```

Then use the interactive file navigator to select the `mydata.csv` file.

Keep in mind that the object holding the data in R is called, in this example, `a`, not `mydata.csv` or `mydata`. The data have been imported from a file called `mydata.csv` into an R object called `a`.

If you are using anything other than `NA` to represent missing data, or if you are (unadvisedly) using `NA` as a code for something other than missing data, you can use the `na.strings` argument to `read.csv` or `ISMdata`. For instance, if you were using the codes `na` and `missing` to stand for missing data (in different variables perhaps) and using `NA` as the code for, say, “National Airlines,” then your file-importing statement would be:

```
> a = ISMdata(na.strings=c("na", "missing"))
```

```
missing data!na.strings ISMdata*!na.strings
```

Once you have imported the data, look at the names of the variables:

```
> names(a)
```

If the names are what you intended, the file has likely been imported in a reasonable way. If not, check whether you entered the names correctly as the first row of the spreadsheet and check whether there are spaces or other punctuation in the names entered in the spreadsheet. (If the names are not in a permitted format, R will try to modify them into something that is allowed.)

If you fix the problem in your spreadsheet, remember to re-save the spreadsheet in the CSV format, overwriting the old CSV file.

The most common problems that beginners encounter are these:

- You saved the data in some other format than CSV. Software such as Excel pushes you to save data in their proprietary format, XLS. Unless you specify CSV, you will get something else even if the name you give the file ends in `.csv`.
- Your spreadsheet is so far from being a simple rectangular data format that `read.csv` couldn't figure out what to do with it.

Bogus Categorical Levels

Categorical variables!mistakes in coding coding!mistakes

Mistakes can be made when coding a categorical variable or recording the data. For instance, you might decide to use F and M to record the sex of a survey subject, but accidentally use f in some of the cases, or mistype g instead of F.

Such problems with completely bogus levels of a variable are usually easily spotted. Once having imported the table to R, look at the levels of the factors. For instance, suppose you are working with a data frame `a` with variable named `sex`. You find out that the levels of `sex` are:

```
> levels(a$sex)
[1] "F" "M" "f" "g" "o"
```

Common sense, or comparing the output to the codebook, reveals the problem.

To fix the problem, you need to go back to the spreadsheet program and edit the file. Make sure to save it again after editing and to re-import it to R.

Editing of data is not something to be undertaken lightly. Fixing typographical mistakes is one thing, but constructing new data is another. In the above, it's reasonable to infer that F was intended for f, and even for g (since g is so close to f on the keyboard). The intention behind o is not so clear. Perhaps the best thing to do is replace o with NA and to record in your notebook the change you made so that you revise the change if that becomes appropriate.

Non-numeric Numbers

When reading in a column of numbers from a CSV file, the `read.csv` and `ISMdata` programs will sensibly produce a variable in numerical format. Sometimes things go wrong.

This happens particularly if one of the entries in that column is in character format: not a number. This might have been a typographical error, or perhaps a missing data marker not identified as such via the `na.strings` named argument. For instance, you might have entered a number as `98.S` instead of `98.5`. If you do, the whole column will be misinterpreted as a factor. `categorical variables!`numerical typos

You can spot such a problem with the `mode` operator `mode`. For example, if your data frame, `a`, has a variable named `count` which is numerical, test whether it has been read in properly with a statement such as

```
> mode(a$count)
[1] "numeric"
```

If the output is not `"numeric"`, look through the corresponding column in the spreadsheet for the non-numeric data entered.

Again, if you edit the spreadsheet, be sure to save it again and to re-import it to R.

Exporting Data from R

[You can skip this section until you need it.] Occasionally you may find that you have created a table in R and want to save it to a format that can be read by others. For example, you may have read in several tables and combined them using the database operations covered in Section ???. Or, you may have created a new variable as part of your analysis, and other researchers may want to use your new variable.

exporting data data!exporting CSV!creating a file

Exporting an R table to a CSV file is easy. Follow these steps:

1. Change directories to wherever you want to place the CSV file that is to be created.
2. Decide on a name for the CSV file. In this example, use `newtable.csv`.
3. Use the `write.table` `write.table` operator. To illustrate, the following statement saves the table stored in a data frame named `a`.

```
> write.table(a, "newtable.csv", sep=",")
```

The `sep` named argument determines what character is to be used as a delimiter between cells in the table. `delimiters!`in CSV files CSV!delimiters Use a comma in order to create a comma separated file. Note that the comma is in quotes; it is a character string, even if a very short one. The name of the target file is also in quotes, but the data frame is not; the point is to write out the contents of the data frame, not the name of the object holding the frame.