# Basic Dataset Exploration & File Preparation

Richard Williams, University of Notre Dame, https://www3.nd.edu/~rwilliam/
Last revised February 14, 2022

You should get to know a dataset before you begin heavily analyzing it. Variable coding may be problematic, e.g. response categories may not be properly ordered; or, a variable may be coded 1/2 when you want it coded 0/1. Missing data may not be handled correctly, e.g. missing data codes may be being treated as legitimate values; or it may not be clear to you why data are missing in the first place.

## Read the Dataset Documentation

At least skim through the study's documentation. It hopefully explains why it is a good dataset, and you'll want to note that if you use the data yourself. It may also tell you about any weaknesses in the data. There may be descriptions of additional variables that the collectors of the data tacked on at the end of the dataset; for example, a computed SES scale may be something you want to use yourself. The correct way to weight the data is often described. The questionnaire can help you be clear on exactly how questions were worded and how the skip patterns work; it can be frustrating if half the data are missing on a variable and you don't know why (e.g. maybe only married people were asked the question).

## Document Your Own Work

- Be sure to document any manipulations you make to the dataset. I've had people come to me with scales they have computed, and they've lost the code that showed how the calculations were done. Or, they have a variable like gender that is coded 0/1, but they can't tell me if 1 = male or 1 = female. I often give my programs names like run01.do, run02.do, etc. so I know and can reproduce the order in which I did things.

- Related to this, you may want to create new variables rather than manipulate existing ones. You'll often want clearer variable names than are used in the data set anyway. Also, if you use the original name for a variable, but have substantially recoded it, others may get confused when they try to replicate your work.

- Including comments in your code may be very helpful. You may understand why you did something now, but six months from now it could be a total mystery.

- In any event, whether you create new variables or not, make sure you can get back to the values in the original dataset.

- Remember too that there is increasing emphasis on replicability of results. Someone may want your data some day, or want to know how you did something. If you can't show what you did it may undermine your credibility. I'm a big fan of Trenton Mize at Purdue partly because he does a great job of making it easy to replicate his results. https://www.trentonmize.com/research .

## Use fre, not tab1, for frequencies

You'll no doubt want to run frequencies on many of your variables. For frequencies, I generally prefer Ben Jann's user-written `fre`, because it gives both the numeric code and the value label for the code (`tab1` only gives you one or the other). For example:

```
. webuse nhanes2f, clear
. tab1 race

-> tabulation of race

       Race |      Freq.     Percent        Cum.
------------+-----------------------------------
      White |      9,051       87.56       87.56
      Black |      1,086       10.51       98.07
      Other |        200        1.93      100.00
------------+-----------------------------------
      Total |     10,337      100.00

. fre race

race -- Race
-------------------------------------------------------------
                |      Freq.    Percent      Valid       Cum.
----------------+--------------------------------------------
Valid   1 White |       9051      87.56      87.56      87.56
        2 Black |       1086      10.51      10.51      98.07
        3 Other |        200       1.93       1.93     100.00
        Total   |      10337     100.00     100.00
-------------------------------------------------------------
```

To get fre, type

```
ssc install fre
```

More generally, to find user-written commands that may be of interest to you, you can use the `findit` command. With `findit` you can specify a specific program or package of programs (e.g. `fre` or `spost13_ado`) or a topic area you are interested in, e.g. user-written commands for dealing with collinearity.

```
findit fre
findit spost13_ado
findit collinearity
```

## Include options with tab2

With two-way cross-tabulations, you can ask for additional output. In particular I like to get a chi-square statistic, which indicates how strongly the variables are related, and the row percentages. (Many other statistics are available.) In the following example the chi-square statistic shows that race and diabetes are related, i.e. the likelihood of having diabetes differs by race. More specifically, the row percentages show you that, while about 8% of the Black respondents have diabetes, only about 4.5% of the White and Other respondents do.

```
. webuse nhanes2f, clear
. tab2 race diabetes, lrchi2 row

-> tabulation of race by diabetes

+----------------+
| Key            |
|----------------|
|   frequency    |
| row percentage |
+----------------+

           |    Diabetes status
      Race | Not diabe   Diabetic |     Total
-----------+----------------------+----------
     White |     8,645        404 |     9,049
           |     95.54       4.46 |    100.00
-----------+----------------------+----------
     Black |     1,000         86 |     1,086
           |     92.08       7.92 |    100.00
-----------+----------------------+----------
     Other |       191          9 |       200
           |     95.50       4.50 |    100.00
-----------+----------------------+----------
     Total |     9,836        499 |    10,335
           |     95.17       4.83 |    100.00

 Likelihood-ratio chi2(2) =  21.7902   Pr = 0.000
```

## Make sure missing data are being handled correctly

Before you do any extensive analysis with your data, you should make sure missing data is coded correctly. The Stata missing value codes are ., .a, .b, .c,…, .z (i.e. . and .a to .z). Even if you downloaded your data in Stata format, the missing data codes may not be correct. For example,

```
. use https://www3.nd.edu/~rwilliam/statafiles/fixcoding, clear
. fre var1

var1
-------------------------------------------------------------------------------
                            |      Freq.    Percent      Valid       Cum.
----------------------------+--------------------------------------------------
Valid   1  Strongly Disagree |         54      24.11      24.11      24.11
        2  Disagree          |         75      33.48      33.48      57.59
        3  Agree             |         29      12.95      12.95      70.54
        4  Strongly Agree    |         42      18.75      18.75      89.29
        97 Don't Know        |          8       3.57       3.57      92.86
        98 Refused           |          5       2.23       2.23      95.09
        99 Not Applicable    |         11       4.91       4.91     100.00
        Total                |        224     100.00     100.00
-------------------------------------------------------------------------------

. sum var1

    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
        var1 |        224     12.5625    29.72513          1         99
```

The values 97, 98, and 99 are missing data codes. That might be correct coding for a program like SPSS, but in Stata those codes are treated as legitimate values, which totally distorts statistics involving the variable, e.g. the mean and standard deviation are wrong here. OLS or logistic regression results could also be way off if you don't fix the MD coding.

The `mvdecode` command is one of the many ways to solve the problem (the `recode` command is another) :

```
. mvdecode var1, mv(97=.a\ 98 = .b\ 99=.c)
        var1: 24 missing values generated
. fre var1

var1
------------------------------------------------------------------------------
                             |      Freq.    Percent      Valid       Cum.
-----------------------------+------------------------------------------------
Valid   1  Strongly Disagree |         54      24.11      27.00      27.00
        2  Disagree          |         75      33.48      37.50      64.50
        3  Agree             |         29      12.95      14.50      79.00
        4  Strongly Agree    |         42      18.75      21.00     100.00
        Total                |        200      89.29     100.00
Missing .a                   |          8       3.57
        .b                   |          5       2.23
        .c                   |         11       4.91
        Total                |         24      10.71
Total                        |        224     100.00
------------------------------------------------------------------------------

. sum var1

    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
        var1 |        200       2.295    1.083441          1          4
```

Much better! Further, suppose var1 thru var20 are consecutive variables in the data set and are all coded the same way. We might then be able to say

```
mvdecode var1-var20, mv(97=.a\ 98 = .b\ 99=.c)
```

Or, better yet, suppose all variables in the data set use the same missing value codes. You could then say

```
mvdecode _all, mv(97=.a\ 98 = .b\ 99=.c)
```

If we want, we can also tidy up the value labels a bit. `var1` uses a value label called `agreement` (using the same value label for several variables that share the same values is often convenient). We can get rid of the old labels and add the new with the commands

```
. label define agreement 97 "" 98 "" 99 "", modify
. label define agreement .a "Don't Know" .b "Refused" .c "Not Applicable", add
. fre var1

var1
-------------------------------------------------------------------------------
                              |      Freq.    Percent      Valid       Cum.
------------------------------+------------------------------------------------
Valid    1  Strongly Disagree |         54      24.11      27.00      27.00
         2  Disagree          |         75      33.48      37.50      64.50
         3  Agree             |         29      12.95      14.50      79.00
         4  Strongly Agree    |         42      18.75      21.00     100.00
         Total                |        200      89.29     100.00
Missing .a Don't Know         |          8       3.57
         .b Refused           |          5       2.23
         .c Not Applicable    |         11       4.91
         Total                |         24      10.71
Total                         |        224     100.00
-------------------------------------------------------------------------------
```

Other notes:

- Never just assume you did things right! Check things out before and after like I did.
- The missing data codes were pretty obvious in this case. Other times they won't be. Try to check the dataset documentation if you can.
- It is nice when every variable uses the same MD codes, but that doesn't have to be the case. For example, 99 may be a missing value for one variable and a valid value for another.
- Sometimes all missing data are just coded ., the system missing value. That is often fine, but at other times it is helpful to know why data are missing. If you use Stata's multiple imputation commands it is very important that you use different MD codes for different types of MD. Eventually, you may decide that different types of missing data will be treated differently in your analysis.
- See `help mvdecode` for more information and examples.
- Chuck Huber has a nice 2-minute video on "How to convert missing value codes to missing values". I prefer to directly write out code when I can, but sometimes the menu-driven approach he shows is better or easier. See https://www.youtube.com/watch?v=6HV2773-dVM.

## Convert String Variables to Numeric Variables

Sometimes variables are in string format rather than numeric format. (Further complicating things, when you are viewing the data it isn't always obvious whether a variable is string or numeric.) For example, a variable called sex might be coded 1 = female, 0 = male, which is a numeric coding. Or, it might be coded "female" and "male" which makes it a string variable, i.e. words are used instead of numbers to code values. Stata usually wants numeric variables in analyses, so string variables must be converted to numeric variables.

You can often use the `encode` command to solve this.

```
. use https://www3.nd.edu/~rwilliam/statafiles/fixcoding, clear
. fre genderstr

genderstr
-------------------------------------------------------------
               |      Freq.    Percent      Valid       Cum.
---------------+---------------------------------------------
Valid   female |        116      51.79      51.79      51.79
        male   |        108      48.21      48.21     100.00
        Total  |        224     100.00     100.00
-------------------------------------------------------------

. des genderstr

Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------------------
genderstr       str6    %9s
```

In the above, `genderstr` is a string variable with two values, "male" and "female." Using `encode`, we can easily create a numeric variable with the values correctly labeled (the value label will have the same name as the generated variable, in this case `gender`):

```
. encode genderstr, gen(gender)
. fre gender*

genderstr
-------------------------------------------------------------
               |      Freq.    Percent      Valid       Cum.
---------------+---------------------------------------------
Valid   female |        116      51.79      51.79      51.79
        male   |        108      48.21      48.21     100.00
        Total  |        224     100.00     100.00
-------------------------------------------------------------

gender
-----------------------------------------------------------------
                 |      Freq.    Percent      Valid       Cum.
-----------------+-----------------------------------------------
Valid   1 female |        116      51.79      51.79      51.79
        2 male   |        108      48.21      48.21     100.00
        Total    |        224     100.00     100.00
-----------------------------------------------------------------

. des gender*

Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------------------
genderstr       str6    %9s
gender          long    %8.0g      gender
```

The variable gender is now numeric, with value label gender, and 1 = female, 2 = male.

Numeric codes are assigned alphabetically, and female comes before male so it got coded 1. But suppose what you really want is a variable coded 0 = male, 1 = female. You can do this by defining the value label before you generate the numerical variable:

```
. label define female 0 "male" 1 "female"
. encode genderstr, gen(female)
. fre gender female

gender
-------------------------------------------------------------
                  |      Freq.    Percent      Valid       Cum.
------------------+------------------------------------------
Valid   1 female  |         116      51.79      51.79      51.79
        2 male    |         108      48.21      48.21     100.00
        Total     |         224     100.00     100.00
-------------------------------------------------------------

female
-------------------------------------------------------------
                  |      Freq.    Percent      Valid       Cum.
------------------+------------------------------------------
Valid   0 male    |         108      48.21      48.21      48.21
        1 female  |         116      51.79      51.79     100.00
        Total     |         224     100.00     100.00
-------------------------------------------------------------
```

Stata has some good resources describing how to do this:

https://www.youtube.com/watch?v=Js_i3wI2-jY
https://www.youtube.com/watch?v=ZRWHjdIZyxo
https://www.stata.com/support/faqs/data-management/numeric-variables-input-as-string/

You can also just read the help for the `encode` command. The `destring` command is also helpful in some cases.

## Create Binary or Ordinal Variables from Continuous Measures

Sometimes you have continuous measures that you want to convert to binary or ordinal variables. For example, `birthweight` (a continuous variable) might be used to create a variable called `lbw`, where 1 = low birth weight, 0 = not low birth weight. Or, `income` (measured in thousands of dollars) might be used to create a variable where 0 = No Income, 1 = $1 to $4999, 2 = $5000-$14,999, … 9 = $100,000 and above. Or, you might just want to take a variable that is coded 1/2 and create a new variable that is coded 0/1. Chuck Huber shows how to do this at

https://www.youtube.com/watch?v=XWVaXN2KwmA

Huber illustrates how to do this via menus, but once you know how, it is probably easier to use the `recode` command directly.

## Handling More Complicated Data Structures

There are several types of datasets that raise their own special issues. Panel data involve the same cases measured at multiple points in time, e.g. children interviewed at ages 8, 10, 12, 14, and 16. Multilevel data involves subjects nested within other structures, e.g. you might have a

sample of schools which in turn has a sample of students from within each school. Complex survey designs have cases that did not all have equal probability of selection. You might get a data set where multiple imputation of missing data has already been done.

To get these data ready to use, you might eventually have to use commands like `xtset`, `svyset`, `miset`, `mi import`, `reshape`, and others.

I won't try now to summarize all the issues involved with these sorts of data. The course has entire handouts dealing with these topics, and you may want to read ahead a bit to find out how to deal with them.

## Other Useful Resources

The above are situations I have frequently encountered with students. There are many other common issues you might encounter when getting your data ready. Stata lists several resources for Data Management (as well as statistical analyses) at

https://www.stata.com/links/video-tutorials/

https://www.stata.com/support/faqs/

Whenever you use a Stata video or FAQ, I encourage you to read the help files for the commands shown, as they may have additional useful capabilities not otherwise shown in the Video or FAQ.