Dealing with Strings and Dates in Stata

Strings

- Strings are a type of variable (short for *character strings* or *strings of characters*)

- Strings often represent non-numeric identifying information (addresses or names)

- Strings are also how many categorical variables are coded in outside data sets (“Male” vs. “Female” or “Red” vs. “Blue” vs. “Green”)

- Capitalization matters in strings: “Cat” is different than “cat”

- Stata is effective at loading strings, but most analytic commands can not be performed on strings

String Manipulation

- There are a **large** subset of commands that deal with manipulating strings
- We'll look at three here:
    - **`strtrim`**`(s)` - trims leading and trailing whitespace from string `s`
    - **`word`**`(s,n)` - selects the nth word of a string `s`
    - **`strpos`**`(s, t)` - returns the position of the string `t` in the bigger string `s`

strtrim(s)

```
display strtrim("          front-spaced")

display strtrim("back-spaced                ")

display strtrim("         both-spaced        ")
```

- This command is a good starting point when processing raw data from Excel or csv files
- Many times extra spaces are added to the front or end of messy variables

word(s,n)

```
sysuse auto, clear

generate first_word = word(make, 1)

generate second_word = word(make, 2)

list first_word second_word
```

- This command is good for separating names or generally parsing information from a string variable

`strpos(s, t)`

```
sysuse auto, clear

generate first_word = word(make, 1)

generate second_word = word(make, 2)

list first_word second_word
```

- This command returns the index of the first location of a string t in a string s, but also returns 0 if there is no

`destring(s), replace`

- During data loading or processing, variables that should be numbers can be incorrectly read as strings
- To transform these variables back to numbers, use the `destring` command
- An alternative method is to use `generate var_name = real(s)`

`encode`

- Similarly, variables that should be categorical are often read in as strings

```
encode variable_name, generate(new_variable)
```

- This command creates a new **number** variable from a string variable, with level labels that correspond to the original string variable

- This is a quick and easy way to generate correctly labeled numerical coding of string variables

Dates

- Stata distinguishes between `date` type variables (measured on the order of a calendar day) and `datetime` type variables (measured down to the millisecond in time)

- In Stata, `date` and `datetime` variables are kept in memory as `double` type numbers, representing the number of days (`date`) or milliseconds (`datetime`) from January 1st, 1960.

- This form is called **SIF** or **S**tata **I**nternal **F**orm, and allows Stata to perform simple operations on dates (addition, subtraction, summaries, etc).

SIF

- To obtain a SIF `date` of today's date, for example:

```
generate todays_date = date("26/7/2017", "DMY")
```

- We can see that this is represented as the number of days from 01/01/1960
- The "`DMY`" argument to the date command tells Stata what the structure of the input string will be
- There are many other possible structures, for example:
    - `MDY` - month day year ("August 18, 2034")
    - `MD20Y` - month day 2-digit year ("12/3/05") *For 1900's use `19Y`*
    - `YDM` - year day month ("2011-12-03")

HRF

- SIF (Stata Internal Form) dates are useful for sorting and performing operations on date variables

- However, they are impossible for humans to parse

- As a result, there is date format **HRF** (Human Readable Form) which presents dates in readable form

- To apply a format that makes todays date readable:

```
format todays_date %td
```

Date Operations

- Suppose we have a birthdate and an observation date, and want to determine age (in years) at observation
- First we would want to translate these dates into **SIF** and then perform a subtraction
- This would give us “days old” which we could then translate into years old and **`int`**
- If we were interested in more specific second-by-second measurements we could use `datetime` variables and the **`clock`** command (instead of **`date`**)

Other Date Formats

- **Good News**: Stata automatically synchronizes `date` and `datetime` variables from Excel dates and times when using the **`import`** `excel` command

- **Bad News**: SPSS, R, or SAS dates all use slightly different encoding mechanisms for their dates.

  - To translate from these formats to SIF, consult the extended **`datetime`** help in the user's manual

Exercises (1)

There's been some sort of processing disaster! We have a dataset (strings.csv) which is supposed to contain information about a drug study but for each observation all the variables have been combined together as one long string.

The intended columns are each separated by a space, and are:

1. ID number
2. Breath Score Before Drug
3. Breath Score After Drug
4. Color Group
5. Reported Gender
6. Treatment Group
7. Birth Date
8. End of Study Date

Exercises (2)

1. Using your knowledge of string and date processing, create a do file (string.do) which processes this dataset:

    A. Load the strings.csv file – there may be something tricky, *check the help file to find out how to restrict rows or columns*

    B. Remove all observations from the **yellow** group.

    C. Create an appropriately named variable for each column

    D. Make sure that categorical variables are properly coded

    E. Make new birth date and end of study date variables in SIF with HRF formats

    F. Calculate an age at end of study variable (in years)

    G. Properly label each variable and categorical variable levels

    H. Save your clean dataset as drug.dta!