1. Introduction

1.1 Dataset: Observations and Variables

This section discusses data structure of a dataset with respect to observations and variables.

1.1.1 Data Structure of a Dataset

A dataset is a data table that has a set of observations. An observation, often called case, is a collection of information of a unit of analysis. Individual information on the attributes of a unit of analysis is stored in a variable.[1] Imagine a worksheet in Excel that arranged by row (observation) and column (variable).

Figure 1.1 illustrates how a dataset looks like. The left visualizes concepts of observations and variables. The right shows a part of an actual STATA dataset.

Figure 1.1 Observations and Variables in a Dataset

Dataset		var_1	var_2	⋯	var_k
	obs_1	.	.	⋯	.
	obs_2	.	.	⋯	.
	⋯⋯	⋯	⋯	⋯	⋯
	obs_n	.	.	.	.

id	age0	age	male	interest
1025	29	1	0	1.00
1026	40	3	1	3.50
1027	27	1	0	.
1028	34	2	.	5.00
1029	35	2	1	4.00
…	…	…	…	… …
1226	50	4	1	3.25

It is highly recommended to have unique identification in a dataset in order to trace observations back and forth.

1.1.2 Rules of Naming

It is important to have proper names of files, variables, macros, functions, and labels in data analysis. In particular, variable name is most critical since data analyses are based on variables.

- Use characters (a- z and A-Z), numbers (0-9), or underscore (_) only.[2]
- Begin with a letter.[3]
- The shorter, the better. Do not exceed 10 characters unless necessary.[4]
- Avoid reserved words or keywords (e.g., command and function).
- Use meaningful names associated with contents of the variable.

[1] Observation is also called record or entity, while variable may be called field or attribute.
[2] Do not use special characters such as -, space, ~, !, @, #, $, %, ^, &, *, (,), {, }, [,], <, >, ?, and /.
[3] It is because underscore is often used in system variables such as _N, _n, _pi, _b, _coef, and _cons.
[4] STATA allows up to 32 characters as a variable name.

- Make it consistent and systematic.[5]
- Use lower cases unless necessary or required.
- Use underscore instead of space
- Use a value of the dummy variable.

1.1.3 Good and Bad Names

Most common mistakes in naming are allowing blank (e.g., `US citizen`), beginning with a number (e.g., `2002_sale`), and using a very long name (e.g., `How_would_you_…`). Table 1.1 compares good and bad examples of variable names.

Table 1.1 Good and Bad Variable Names

Good Example	Bad Example	Description
`gnp2002`	`gnp-2002; gnp#2002`	Avoid special characters
`real_int`	`real interest rate`	Use underscore
`score1; gnp2003`	`1st_score; 2003gnp`	Begin with a character
`reg_out; glm1`	`REG; glm; ttest`	Avoid reserved words
`invest; interest`	`xxx; yyy; zmdje;`	Use meaningful names
`male; black`	`gender; race`	Use a value of dummy
`score1; score2;…`	`math; math_1; math02`	Consistent and systematic
`citizen`	`Are_you_a_US_citizen?`	The shorter, the better
`income; intUS03`	`INCOME; Int_us2003;`	Use lower cases

Naming is a beginning point of data analyses. Bad naming may frequently bother you during the analyses.

1.2 STATA Basics

STATA is available in a variety of platforms and flavors. STATA runs under UNIX, LINUX, Microsoft Windows and Apple Macintosh OS.

1.2.1 Three Flavors

Stata has three different flavors. Stata/SE (Special Edition) is most powerful in that it can handle large data sets and matrices in a fast and safe manner. Intercooled Stata, a standard version, provides moderate capacity for ordinary users. Small Stata, a limited edition, is not available in UNIX machines. Table 1.1 summarizes major differences among the three flavors. This book mainly focuses on STATA/SE (release 8 and 9) under Microsoft Windows.

Table 1.2 STATA Three Flavors

Maximum	Special Edition	Intercooled Stata	Small Stata
Observations	Limited by memory	Limited by memory	1,000
Variables	32,766	2,047	99
Dataset Width	393,192	24,564	200

[5] You can benefit from using array and wild card as in score1-score10, score??, vote*.

Command	1,081,527 characters	67,800 characters	8,697 characters
Macro	1,081,511 characters	67,784 characters	8,681 characters
String Variable	244 characters	80 characters	80 characters
Matrices	11,000 by 11,000	800 by 800	40 by 40
One-way Table	12,000	3,000	500
Two-way Table	12,000 by 80	300 by 20	160 by 20

STATA puts a dataset into computer memory (including virtual memory), but it does not automatically use all the memory available in your computer. STATA/SE by default assigns 10MB for dataset. When reading a large dataset, you may need to adjust memory size, maximum number of variable, and/or matrix size using the ***.set memory***, ***.set maxvar***, and ***.set matsize*** commands.[6]

```
. set memory 150m, permanently
. set maxvar 10000
. set matsize 2000
```

You may also use virtual memory to have enough room for a dataset at the expense of processing speed.

```
. set virtual on
```

1.2.2 Variable Types

STATA supports six variables types, which are grouped into real number, integer, and string. Default type is `float`, single precision real number. Date type is deal with the string type and conversion functions.

Table 1.3 STATA Variable Types

Keyword	Type	Bytes	Format	Range
`float`	Real	4	%9.0g	$1.70141173319\times(-10^{38} \sim 10^{36})$ (8.5 digits of precision)
`double`	Real	8	%10.0g	$8.9884656743\times(10^{307} \sim 10^{308})$ (16.5 digits of precision)
`byte`	Integer	1	%8.0g	-127 ~ 100
`int`	Integer	2	%8.0g	-32,767 ~ 32,740
`long`	Integer	4	%12.0g	-2,147,483,647 ~ 2,147,483,620
`str#`	String	#	-	str1 through str244*

You need to use proper variable types in order for efficient memory management. For instance, the ***byte*** type (1 byte) is best for five-point Likert scale. Use ***int*** (2 bytes) rather than ***long*** (4 bytes), and ***float*** (4 bytes) rather than ***double*** (8 bytes), unless required.

1.2.3 Default Extensions

Table 1.4 summarizes default extensions used in STATA. These default extensions are often omitted.

[6] However, increasing memory size does not always improve the overall performance of STATA. The optimal memory size depends upon computing resources and the size of the dataset.

Table 1.4 STATA Default Extensions

Default	File Types	Related Commands
.dta	STATA format dataset	`.use` and `.save`
.do	STATA do-file	`.do` and `.doedit`
.ado	Automatically loaded do-file	`.doedit`
.log	Log file in the text format	`.log`
.smcl	Log file in the SMCL format	`.cmdlog`
.raw	ASCII text file	`.infile`, `.infix`, and `.insheet`
.out	Files saved by the ***.outsheet***	`.outsheet`
.dct	ASCII data dictionary	`.infix`
.gph	Graph image	`.graph`

1.2.4 Length of Names and Labels

Table 1.5 summarizes the maximum length of names and labels.

Table 1.5 Length of Names and Labels

Keyword	Maximum Length	Notes
Variable Name	32 characters	Function name?
String Variable	244 characters	
Dataset Label	80 characters	`.label data "…"`
Variable Label	80 characters	`.label variable var_name "… "`
Value Label Name	32 characters	`.label define lbl_name # "…";`
Value Label	32,000 characters*	`.label values var_name lbl_name`
Language Label		`.label language lang_name`
Local Macro Name	31 characters	`.local mac_name "…"`
Global Macro Name	32 characters	`.global mac_name "…"`
Macro Variable	1,081,511 characters	

* The intercooled allows only 80 characters.

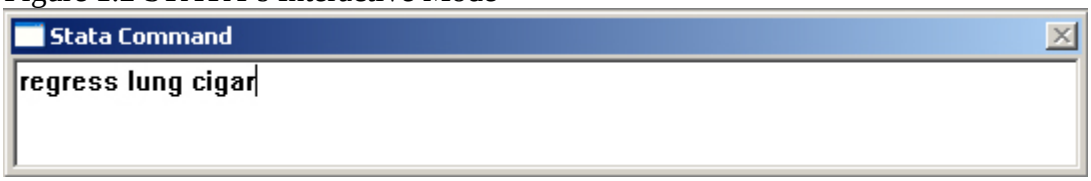
1.3 STATA Interface

There are three ways to communicate with STATA: Interactive mode, non-interactive mode, and point-and-click.

1.3.1 Interactive mode

STATA is a command-driven application. This interactive mode enables users to communicate with STATA step by step. Users need to type in a command and hit ENTER to run the command. Then, STATA interprets the command, processes the job, and return its result to users (Figure 1.1).

Figure 1.2 STATA's Interactive Mode



STATA systematic grammar structure and abbreviation rules makes it efficient and flexible to perform many simple tasks. STATA must come in pretty handy.

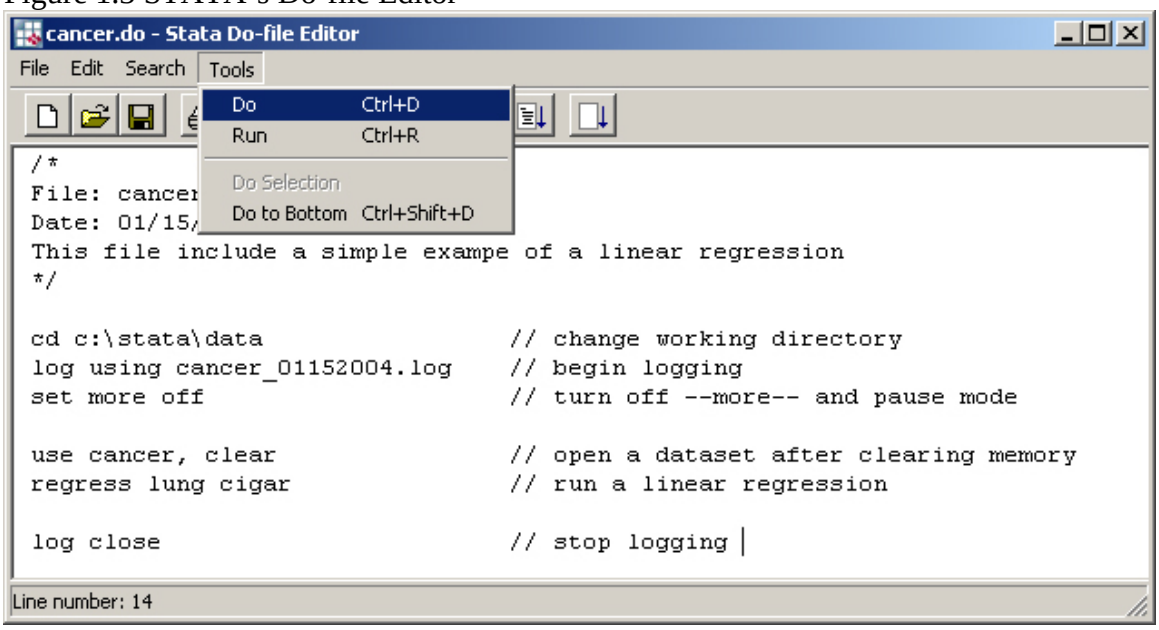
Unlike compilers, STATA command interpreter keeps analysis results in memory even after executing commands so that users can conduct necessary follow-up analyses without running entire analyses again.

1.3.2 Non-interactive mode (batch mode)

The non-interactive mode executes a set of commands written in a text file. Classical statistical software like SAS uses this mode of communication. Instead of running individual commands one by one in the interactive mode, users may write a ***.do*** file, a batch file, in which a set of commands are organized. Writing a do file is efficient especially when a bundle of commands needs to be repeated many times.[7]

In order to open the STATA Do-file editor, click Window→Do-file Editor or pressing Ctrl+8. Alternatively, run the ***.doedit*** command or click the Do-file editor icon  . You may also use a text editor like Notepad to write a do file.

Figure 1.3 STATA's Do-file Editor



Once a .do file is ready, you may execute the batch job by running the ***.do*** command in the command window. Alternatively, you may choose Tools→Do menu (Ctrl+D) or click

[7] Another type of programs is the ***.ado*** file. In fact, many STATA commands are based on ***.ado*** files. Although StataCorp provides basic ***.ado*** files, users also can write their own ***.ado*** programs to add their own commands to STATA. The ***.do*** files include typical STATA commands, while ***.ado*** programs need to be written in the STATA ado language. This book does not address ***.ado*** programming.

 in the Do-file Editor window. When you wish to execute only a part of commands, highlight the block of commands using a mouse, and choose Tools→Do Selection menu.

1.3.3 Point-and-Click (Graphical User Interface)

STATA's point-and-click provides graphic user interface environment, where users pull down menus and select a proper menu of a command to invoke the dialog box. STATA echoes the command on the basis of information provided in dialog boxes.

In order to invoke a proper dialog box, run the ***.db*** command or use shortcuts. For instance, you may run `.db save` command or press Ctrl+S (pressing S key while the Ctrl key is pressed), which is equivalent to clicking FILE→Save.

1.4 STATA Commands

1.4.1 Command Conventions

There are several conventions for STATA commands.

- Commands are lowercased.
- Commands, variable names, and options can be abbreviated.
- No character is required at the end of a command.
- A command and its options should be separated by a comma.
- There is no comma between variables and between options.
- A dependent variable precedes a set of independent variables.

1.4.2 Abbreviations

STATA commands, variable names, and options can be abbreviated to the shortest string of characters as long as they are uniquely identified. The minimum abbreviations are underlined in help and manuals (e.g., ***tabulate***). However, some commands like the ***.replace*** cannot be abbreviated. Users also use wildcards such as ?, *, and ~ when abbreviating variable names (see Table 1.5).

1.4.3 Command Structure

A STATA command in general consists of,

- A command (with subcommands)
- A list of variables (dependent and independent variable)
- Qualifiers (***in*** and/or ***if***)
- Option(s)

A command may or may not have their subcommands. A command may have a series of options as follows.

```
. list state lung cigar, nolabel noobs separator(10)
```

1.4.4 Listing Variables

You may list all variables to be used. Omitting a list of variables implies all variables in a dataset. STATA allows various ways of listing variables using wildcards (Table 1.5).

Table 1.6. Wildcards

Wildcards	Descriptions	Examples
?	Any character	d?
*	Any characters	re*
~	zero or more characters	mil~um
-	Specifying range of variables	gender-rank

For example, `d?` means the variables beginning with d and ending with any single character and number (e.g., da, db, dc… d1, d2, d3…), while `re*` indicates any variables beginning with `re` (e.g., `retain` and `return`). The `in~t` means any variables beginning with `in` and ending with `t`. (e.g., `invent` and `interest`) The `gender-rank` indicates all variables from `gender` through `rank` of the variable list in a dataset. Followings are some examples of using wildcards in listing variables.

```
. list
. list state d? re*
. list state-lung in~t
```

The ***in*** and ***if*** qualifiers specify a subset of a dataset to which a command is applied.

1.4.5 Selecting Observations

The ***if*** and ***in*** qualifiers specify a subset of a dataset to which a command is applied. The ***if*** qualifier selects observations that meet the conditions imposed. You may use & (and) and/or | (or) relational operators to provide more than one condition.

```
. list if area==3
. list state cigar lung if (area==4) & (lung >= 10)
```

The ***in*** qualifier directly specifies the range of observations. You may use observation numbers (record numbers) or some symbols indicating particular observations (Table 1.6). Note the "/" separates beginning and ending observation numbers.

```
. list in 10
. list in 10/50
. list cigar-kidney in f/10
. sum bladder cigar in f/l
```

Table 1.7 Symbols of the ***in*** Qualifier.

Symbols	Example	Meaning
#	in 10	The 10th observation
-#	in -10	The 10th observation from the last
1 (or f)	in 1/10; in f/10	From the **first** observation through the 10th
-1 (or l)	in 15/-1; in 15/l	From the 15th observation through the **last**

However, you may not list more than one observation numbers without the / operator, nor specify observation numbers as well as the range of observations at the same time.

1.5 Commands, Function, and Operators

1.5.1 Basic Commands

Table 1.8 summarizes the basic commands frequently used in STATA.

Table 1.8

Commands	Description
.display	Echo strings and values of scalar expressions
.use, .save	Load and save a dataset
.describe	Describe dataset in memory
.summarize	Summary statistics
.tabulate	One-way and two-way table of frequencies
.list	List values of variables
.edit, .browse	Edit and view a dataset in Data Editor
.generate, .egen	Generate variables
.replace, .recode	Modify and recode variables
.count	Count the number of observations
.version	Return release number and set the command interpreter
.memory, .set memory	Check and set memory size
.format	Specify variable display format
.lookfor	Search for sting in variable names and labels
.quietly, .noisily	Suppresses and turns back STATA output

1.5.2 Operating System Commands

Table 1.9 summarizes useful operating system commands. Note that the .pwd and .rm are available only under Macintosh OS and UNIX, respectively.

Table 1.9 STATA Operating System Commands

Command s	Descriptions
.cd (.pwd in Mac OS)	Change a directory
.copy	Copy files
.dir (or ls)	List directories and files
.erase (.rm in UNIX)	Remove files
.mkdir	Create a directory
.shell	Invoke operating system temporarily
.type	View contents of a text file

1.5.3 Operators and Symbols

Table 1.10 illustrates various operators used in STATA. Note that the equal operator is not "=" (assignment), but "==."

Table 1.10 STATA Operators

Types	Operators
Arithmetic Operator	+, -, *, /, ^ (power)
Relational Operator	>, >=, <, <=, == (equal), != (not equal)
Logical Operator	& (and), | (or), ! (not)
Assignment	=
Concatenation	+
Backward Shift	L#.variable

There are also several symbols that are frequently used in STATA (Table 1.11). Note that the /// is useful when a command is too long to be listed in a line.

Table 1.11 Useful Symbols

Symbol	Descriptions
/	Specifying range of observations in the ***in*** qualifier
// …	Comment in programming
///	join the next line with the current line in do and ado programs
/* …*/	Comment in programming
* …	The same as // …

1.5.4 Functions

Table 1.12 and 1.13 list commonly used mathematical and string functions. Table 1.14 in section 1.7 summarizes major probability distribution functions.

Table 1.12 Mathematical Functions

Functions	Descriptions
abs(x)	Absolute value
sin(x),cos(x),tan(x)	Sine, cosine, tangent
ceil(x), floor(x)	Unique value
int(x), round(x)	Truncations
comb(n, k)	Combinational function
exp(x)	Exponential function
Ln(x) or log(x)	Natural logarithm
logit(x),invlogit(x)	Log of the odd ratio and its inverse
max(x), min(x)	Maximum and minimum values
mod(x,y)	Modulus of x with respect to y
sign(x)	Sign
sqrt(x)	Square root
sum(x)	Sum

Table 1.13 String Functions

Functions	Descriptions
char(n)	Character corresponding to ASCII code n
index(s, key)	Position in s at which key is first found; otherwise zero
length(s)	The length of a string

`lower(s)`	Lowercase string
`ltrim(s)`	A string without leading blanks
`real(s)`	To convert a string to a number
`Reverse(s)`	A reversed string
`rtrim(s)`	A string without trailing blanks
`string(n)`	To convert a number to a (formatted) string
`substr(s, n1, n2)`	Substring of s starting at n1 for a length of n2
`trim(s)`	String without leading and tailing blanks
`upper(s)`	Uppercase string
`word(s)`	The number of words in a string

1.6 Repeating a Command (*.bysort* and *.by*)

You may wish to run the same command on each group instead of the entire dataset. Consider the following commands.

```
. sum cigar lung if area==1
. sum cigar lung if area==2
```
…

This approach works, but it will be burdensome when there are many groups. Here is the rationale the ***.bysort*** (or ***.bys***) and ***.by*** commands are needed. However, not every STATA command can be used with the ***.bysort*** and ***.by*** commands.

1.6.1 The `.bysort` Command

The ***.bysort*** repeats STATA command on each group without the ***if*** qualifier. The ***.bysort*** command first sorts the variable in an ascending order, and then repeats the command on groups. Note that colon (:) separates the ***.bysort*** or ***.by*** from the command to be repeated.

```
. bysort area: sum cigar lung
-------------------------------------------------------------------------------
-> area = 1
-------------------------------------------------------------------------------
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
       cigar |         8    27.94625    2.297881      23.78       31.1
        lung |         8    21.72375    4.262283      12.11      25.95
-------------------------------------------------------------------------------
-> area = 2

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
       cigar |        12    23.70667    2.762431      19.96      27.91
        lung |        12    18.31667     3.68153      12.12       22.8
...
```

1.6.2 The `.by` Command

The ***.by*** command with the ***sort*** option is equivalent to the ***.bysort***.

```
. by area, sort: sum cigar lung
```

Alternatively, you may omit the ***sort*** (or ***s***) option, if you sort the variable in advance.

```
. sort area
. by area: sum cigar lung
```

1.7 Using the *.display* Command

The ***.display*** (or ***.di***) command displays strings and values of various scalar expressions. This command also echoes outputs of a program.

1.7.1 Displaying Strings and Values of Variables

The following is an example of displaying a string and values of system variables. Note that the `_pi` below is a system variable.

```
. display “Pi is “ _pi
Pi is 3.1415927
```

Next example displays values of two variables using explicit subscripts. The number in a bracket indicates the observation number (record pointer).

```
. display state[12] cigar[12]
```

1.7.2 Using As a Hand Calculator

This command enables users to use STATA as a calculator. The followings show how various expressions can be used in this command.

```
. display 5*5*3.14
. display (1.3)^(1/12)-1
. di (6.4-5.0)/sqrt(10)
```

1.7.3 Using Probability Distributions

One of the biggest benefits of the ***.display*** command is that users can get p-values without referring probability distribution tables. The various probability distribution functions are used in the expressions of this command (Table 1.14). Consider the following examples.

```
. di normal(1.96)
. di (1-normal(1.96))*2
```

The ***.normal(z)*** returns the cumulative probability of the standard normal distribution. So the second command gives you the two-tailed p-value of the z score 1.96.

The ***ttail(df , t)*** returns the reverse cumulative (upper-tail only) Student's t distribution. The first example below returns the two-tailed p-value of the t value 2.086 with degree of freedom 20.

```
. disp ttail(20, 2.086)*2
```

The ***chi2tail(df, c)*** gives you the reverse (upper-tail) cumulative probability of the chi-squared distribution. Similarly, the ***Ftail(df1, df2, F)*** returns reverse (upper-tail) cumulative probability of the F distribution. Note that the F is uppercased and that the first number is the degree of freedom for numerator.

```
. disp chi2tail(10, 18.307)
. disp Ftail(5, 10, 3.325)
```

The t, chi-squared, and F scores used above, in fact, are critical values of the distribution at the .05 level. Thus, all examples produce .05.

Table 1.14 Major Probability Distribution Functions

Functions	**Descriptions**
`binomial(n, k, p)`	Binomial probability distribution of ***k*** or more successes in ***n*** trials
`binormal(h, k, p)`	Joint cumulative distribution of bivariate normal
`chi2(d, x)`	Cumulative chi-squared distribution
`chi2tail(d, x)`	Reverse cumulative (upper-tail) chi-squared distribution
`F(d1, d2, f)`	Cumulative F distribution
`Fden(d1, d2, f)`	Probability density function of the F distribution
`Ftail(d1, d2, f)`	Reverse cumulative (upper-tail) F distribution
`normal(z)`	Cumulative standard normal distribution
`normalden(z)`	Standard normal density
`normalden(z, s)`	Rescaled standard normal density
`tden(d, t)`	Probability density function of Student's t distribution
`ttail(d, t)`	Reverse cumulative (upper-tail) Student's t distribution