An Introduction to Stata for Health Researchers

Fifth Edition

SVEND JUUL
Department of Public Health
Aarhus University
Aarhus, Denmark

MORTEN FRYDENBERG
Aarhus, Denmark

A Stata Press Publication
StataCorp LLC
College Station, Texas

Second edition 2008
Third edition 2010
Fourth edition 2014
Fifth edition 2021

Published by Stata Press, 4905 Lakeway Drive, College Station, Texas 77845
Typeset in LaTeX 2$_\varepsilon$
Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

Print ISBN-10: 1-59718-315-6
Print ISBN-13: 978-1-59718-315-4
ePub ISBN-10: 1-59718-316-4
ePub ISBN-13: 978-1-59718-316-1
Mobi ISBN-10: 1-59718-317-2
Mobi ISBN-13: 978-1-59718-317-8

Library of Congress Control Number: 2021933404

Contents

Preface to the fifth edition

This fifth edition updates the fourth edition to reflect the changes in Stata 14, released in 2015; Stata 15, released in 2017; Stata 16, released in 2019; and Stata 17, released in 2021.

Since the fourth edition of the book, many nice things have happened with Stata, and many of these changes are also reflected in the book. In several ways, Stata has become more user-friendly, for example, with an improved Do-file Editor.

Stata now has commands for working with both the 9th and the 10th releases of the International Classification of Diseases, and we have included them in the book. We also included a chapter on the much-improved commands for power, precision, and sample-size analysis. With release 14, Stata introduced Unicode, giving the opportunity to use a wealth of characters beyond the Latin alphabet, and the consequences of that affect several sections in the book. With release 17, Stata introduced tools to tailor publication-ready tables, and we wrote an introduction to these tools. Finally, we made a complete revision of the important (we think) chapter 9, *Taking good care of your data.*

This is an introductory book aimed at people working in health research, and we had to make several decisions about what to include and what to omit. If you miss something that is not described in the book, it does not necessarily mean that Stata cannot do it. Use the `help` command and the PDF manuals to learn more.

During the process, Bill Rising and Kristin MacDonald at StataCorp gave several useful suggestions to improve the quality of the book, and Lisa Gilmore coordinated everything.

User reactions are welcome and can be good inspiration for further improvements, so please feel free to send comments to sj@ph.au.dk or mfstat@mollerfryd.dk.

Aarhus, Denmark
July 2021

Svend Juul and Morten Frydenberg

(Pages omitted)

3 Command syntax

3.1 General syntax rules

Stata's language rules are described in detail in [U] **11 Language syntax**; to get there, type

```
. help language
```

Stata is case sensitive, and all official Stata command names are lowercase. `list` is a valid command, but `List` is not. Variable names may include lowercase and uppercase letters, but `sex` and `Sex` are two different variable names. Throughout this book, we use lowercase variable names.

Variable names can have up to 32 characters, but Stata often abbreviates long variable names in output, so we recommend avoiding more than, say, 10 characters. Numbers (0–9), letters (A–Z, a–z, and any Unicode letter), and _ (underscore) are valid characters in variable names. Names must start with a letter and can contain an underscore, but starting with an underscore should be avoided because many Stata-generated temporary variables begin with an underscore. The following are valid variable names:

```
a q17 q_17 pregnant sex
```

3.2 Syntax diagrams

A syntax diagram is a formal description of the elements in a Stata command. The notation used is described in [R] **Intro**, which you will find in the beginning of the *Base Reference Manual* [R]. The general syntax of typical Stata commands can be written like this:

[*prefix*:] *command* [*varlist*] [*if*] [*in*] [*weight*] [, *options*]

For example, the syntax for `summarize` is

`summarize` [*varlist*] [*if*] [*in*] [*weight*] [, *options*]

options	description
Main	
`detail`	display additional statistics
`meanonly`	suppress the display; calculate only the mean; programmer's option
`format`	use variable's display format
`separator(#)`	draw separator line after every # variables; default is `separator(5)`
display_options	control spacing, line width, and base and empty cells

Find a more detailed description of the syntax of `summarize` by typing

```
. help summarize
```

Thin square brackets, [ ], mean that the item is optional, so the only mandatory part of the `summarize` command is the command name itself. Square brackets may also be part of the syntax, in which case they are shown in the typewriter font, as in

```
tab2 case ctrl [fweight=pop]
```

Curly brackets, { }, mean that you must specify one of the options but not both options, as in

`numlabel` [*lblname-list*]`,` {add | remove}

Here you must specify either `add` or `remove`.

Command and option names can be abbreviated; in the syntax diagram, underlining shows the minimum abbreviation. We use few abbreviations. Although they make commands faster to write, they make them more difficult to read. Table 3.1 shows some example `summarize` commands:

Table 3.1. Example `summarize` commands

prefix	*command*	*varlist*	*qualifiers/weights*	*options*	Comments
	`summarize`	`_all`			`_all`: all variables
	`summarize`				All variables
	`sum`				Abbreviated
	`summarize`	`sex age`			Two variables
	`summarize`	`sex-weight`			Variable range
	`summarize`	`pro*`			All variables starting with `pro`
	`summarize`	`*ro*`			All variables containing `ro`
	`summarize`	`??ro?`			5-letter variables; `ro` as 3rd and 4th characters
	`summarize`	`age`	`if sex==1`		Males only
	`summarize`	`bmi`	`in 1/10`		First 10 observations
	`summarize`	`bmi`	`[fweight=n]`		Weighted observations
	`sort`	`sex`			Separate table for each `sex`; data must be sorted first
`by sex:`	`summarize`	`bmi`			
	`summarize`	`bmi`		`, detail`	Option: `detail`

3.3 Lists of variables and numbers

Variable lists

A variable list (*varlist*) defines one or more variables to be processed. Here are some examples:

(nothing)	Sometimes means the same as `_all`
`_all`	All variables in the dataset
`sex age pregnant`	Three variables
`pregnant sex-weight`	`pregnant` and the consecutive variables from `sex` to `weight`
`pro*`	All variables starting with `pro`
`*ro*`	All variables containing `ro`
`??ro?`	five-letter variables with `ro` as third and fourth characters

When generating new variables, you can refer to the 17 variables `q1`, `q2`, ..., `q17` as `q1-q17`. When referring to existing variables `q1-q17`, you will get `q1`, `q17`, and the variables that come between them in the dataset, which are not necessarily `q2`, `q3`, ..., `q16`. `summarize` and `describe` are useful commands to see the ordering of variables in the dataset.

In commands that have a dependent variable, it is listed first in the variable list:

`. oneway bmi sex`	`bmi` is the dependent variable
`. regress bmi sex age`	`bmi` is the dependent variable
`. scatter weight height`	Scatterplot, `weight` is the y axis
`. tab2 expos case`	The first variable defines the rows

Numeric lists

A numeric list (*numlist*) is a list of numbers with some shorthand possibilities:

`1(3)11`	means	`1 4 7 10`
`1(1)4 4.5(0.5)6`	means	`1 2 3 4 4.5 5 5.5 6`
`4 3 2 7(-1)1`	means	`4 3 2 7 6 5 4 3 2 1`
`1/5`	means	`1 2 3 4 5`
`4/2 7/1`	means	`4 3 2 7 6 5 4 3 2 1`

Numeric lists have many uses; for example, they can

- display person-time and incidence rates in 0.5-year intervals up to 5 years:
```
. stptime, at(0(0.5)5) by(drug)
```
- show a graph with y-axis labels at 0 10 20 30 40:
```
. scatter mpg weight, ylabel(0(10)40)
```
- generate age groups 0–4, 5–14, 15–24, ..., 75–84, 85+:
```
. egen agegrp = cut(age), at(0 5(10)85 200)
```

Numeric ranges

Numeric lists should not be confused with numeric ranges. The following are ranges:[1]

```
. list in 1/10
. recode age (45/max=3)(25/45=2)(0/25=1), generate(agegr)
```

3.4 Qualifiers

Qualifiers are common to many commands, while most options are specific to one command or a few commands.

The if qualifier

The `if` qualifier is used with logical expressions to select the observations to which a command applies. Here are a few examples (`sex` has the value `1` for males):

`. summarize age if sex==1`	Males only
`. summarize age if sex!=1`	Males excluded
`. list id age if age<=25`	Young only
`. replace npreg = . if sex==1`	set `npreg` to missing for males
`. list sex age weight if sex==1 & age<=25`	Young males only
`. keep if sex==1 | age<=25`	Males or young
`. keep if !(sex==1 | age<=25)`	All others

Two types of operators are used in logical expressions, as shown in table 3.2.[2]

Table 3.2. Operators in logical expressions

Relational operators		Logical operators	
>	Greater than	!	Not
<	Less than	&	And
>=	Greater than or equal to	|	Or
<=	Less than or equal to		
==	Equal		
!=	Not equal		

The double equal sign (==) in relational expressions has a meaning different from that of the assignment equal sign, as in

```
. generate bmi = weight/(height^2)
```

1. You may wonder why we chose to let the `recode` command start with the highest values. See an explanation in section 7.4.
2. Previously, the tilde (~) was used for "Not", and it still works, but it is seldom used.

Logical expressions are evaluated to be true or false. A value of 0 means false, and any other value, including missing values, means true. Technically, missing values are large positive numbers and are evaluated as such in logical expressions. This issue is described in more detail in section 4.2.

With complex logical expressions, use parentheses to control the order of evaluation:

```
. list if ((sex==1 & wt>90) | (sex==2 & wt>80)) & ! missing(wt)
```

Omitting the parentheses might give a different selection, but the outcome may be difficult to predict. Use parentheses to make the syntax transparent to yourself; then it will work correctly. A possibly more transparent way to handle complex selections is to generate a help variable (`heavy`):

`. generate heavy = 0`	Initialize help variable (`heavy`)
`. replace heavy = 1 if sex==1 & wt>90`	Include males > 90 kg
`. replace heavy = 1 if sex==2 & wt>80`	Include females > 80 kg
`. replace heavy = . if missing(wt)`	Do not include if `wt` is missing
`. list if heavy==1`	These are the heavy ones

The in qualifier

The `in` qualifier is used to select the observations to which a command applies. It is especially useful for listing or displaying a subset of observations. Below are three examples:

`. list sex age weight in 23`	23rd observation
`. list in 1/10`	All variables; observations 1–10
`. browse sex-weight in -5/-1`	See last 5 observations in the Data Browser

The last observation is identified by `-1`, and `-5/-1` means the last five observations. Note that the sort order of the dataset may change, so you should not rely on the observation number to identify a specific observation.

3.5 Weights

Weighting observations

Weights can be used to multiply observations when the input is tabular. Suppose that you see the following table in a paper and want to analyze it further:

	Cases	Controls
Exposed	21	30
Unexposed	23	100
Total	44	130

The `input` command (see section 5.2) lets you enter the tabular data directly:

```
input expos case pop
  1 1 21
  1 0 30
  0 1 23
  0 0 100
end
```

Now you can analyze the data by weighting with `pop`:

```
. tab2 expos case [fweight=pop], chi2
. logistic case expos [fweight=pop]
```

The square brackets around the weight expression are shown in typewriter font. They are part of the syntax; here they do not mean optional.

`fweight` indicates frequency weighting. For information about other types of weighting, see

```
. help weight
```

3.6 Options

Options are specific to a command, and you must use the `help` command or look in the PDF documentation to see the available options. Options come last in the command, and they are preceded by a comma. Usually, there is no more than one comma per command, but complex graph commands may include more; see chapter 16.

The `nolabel` option is common to many commands. If value labels have been assigned to a variable, Stata usually displays the value label rather than the code in tables and listings. The `nolabel` option lets you see the code instead of the label:

```
. tab1 race

-> tabulation of race

       Race |      Freq.     Percent        Cum.
------------+-----------------------------------
      White |         96       50.79       50.79
      Black |         26       13.76       64.55
      Other |         67       35.45      100.00
------------+-----------------------------------
      Total |        189      100.00

. tab1 race, nolabel

-> tabulation of race

       Race |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |         96       50.79       50.79
          2 |         26       13.76       64.55
          3 |         67       35.45      100.00
------------+-----------------------------------
      Total |        189      100.00
```

3.7 Prefixes

Only the `by` *varlist*`:` prefix is shown here, but in later chapters, we will illustrate others, for example, `quietly` and `statsby`.

The by varlist: prefix

The `by` *varlist*: prefix makes a command perform calculations or display results for strata of the data. Data must be sorted by the stratification variables. The following commands lead to two `summarize` tables, one for each sex:

```
. sort sex
. by sex: summarize age height weight
```

There are two ways to produce the same results with one command:

```
. bysort sex: summarize age height weight
. by sex, sort: summarize age height weight
```

3.8 Other syntax elements

Text strings with quotes

Stata uses double quotes around text strings. This applies to filenames, file paths, and labels, as shown here:

```
. label define sex 1 "male" 2 "female" 9 "sex unknown"
. use "~/Documents/ishr5/lbw1.dta"
```

Actually, quotes are not needed around text strings without spaces, but in this book, we most often use them to improve readability. The above commands could be written

```
. label define sex 1 male 2 female 9 "sex unknown"
. use ~/Documents/ishr5/lbw1.dta
```

Comments

The following are interpreted as comments, so you can include short explanations in do-files and ado-files:

- Lines beginning with `*`.
- Text following `//`. If `//` are not the first characters in the line, they must be preceded by a space to work properly.

Comments make your do-files more readable; Stata does not care what you write:

```
. * ~/ado/personal/profile.do executes when opening Stata
. // ~/ado/personal/profile.do executes when opening Stata
. summarize bmi, detail            // Body mass index
```

(Pages omitted)

11 Regression analysis

This chapter describes the fundamentals of linear regression and logistic regression. However, many other regression models are available in Stata. Chapter 12 discusses Poisson regression and Cox regression, and the general principles apply to them, too.

We will not go into details of how to analyze data by regression models, but instead, we will focus on some of the features in Stata: working with categorical explanatory variables, testing hypotheses, and using Stata's postestimation facilities.

Stepwise selection procedures are available in Stata, but they will generally lead to invalid estimates, confidence intervals, and p-values. Therefore, we will not describe them. Find a discussion of the problems at https://www.stata.com/support/faqs/statistics/stepwise-regression-problems/, *What are some of the problems with stepwise regression?*

A regression model expresses the dependency of one variable (the response, outcome, or dependent variable) on one or more other variables (predictors, regressors, or independent variables). Short introductions to regression models can be found in many standard textbooks, such as Kirkwood and Sterne (2003). If you want to apply more than the simplest regression models, you should consult books dedicated to the subject, such as Vittinghoff et al. (2012) and Hosmer, Lemeshow, and Sturdivant (2013).

11.1 Linear regression

We are going to use the `lbw.dta` dataset but with added numerical labels; see section 10.1:

```
. cd "~/Documents/ishr5"

. use lbw1.dta
(Hosmer and Lemeshow data - with labels)

. codebook, compact

Variable   Obs Unique      Mean  Min   Max  Label
-------------------------------------------------------------------------------
id         189    189  121.0794    4   226  Identification code
low        189      2  .3121693    0     1  Birthweight < 2500g
age        189     24   23.2381   14    45  Age of mother
lwt        189     76  129.8201   80   250  Last prepregnancy weight (lb)
race       189      3  1.846561    1     3  Race
smoke      189      2  .3915344    0     1  Smoked during pregnancy
ptl        189      4  .1957672    0     3  Premature labor history (count)
ht         189      2  .0634921    0     1  Has history of hypertension
ui         189      2  .1481481    0     1  Presence, uterine irritability
ftv        189      6  .7936508    0     6  Number of visits to physician du...
bwt        189    133  2944.286  709  4990  Birthweight (grams)
-------------------------------------------------------------------------------
```

We will start by looking at a simple linear regression model, which in Stata is fit using the `regress` command.[1] Let us consider the model

$$\texttt{bwt} = \beta_0 + \beta_1 \times \texttt{lwt} + \textit{error}$$

Here `bwt` (birthweight in grams) is the dependent variable, and `lwt` (weight at last menstrual period in lb) is the predictor. *error* is the unexplained random variation; it is assumed normal with mean zero and standard deviation σ. The estimates for β_0, β_1, and σ are easily found with the `regress` command:

```
. regress bwt lwt

      Source |       SS           df       MS      Number of obs   =       189
-------------+----------------------------------   F(1, 187)       =      6.69
       Model |  3449062.64         1  3449062.64   Prob > F        =    0.0105
    Residual |  96466235.9       187  515862.224   R-squared       =    0.0345
-------------+----------------------------------   Adj R-squared   =    0.0294
       Total |  99915298.6       188  531464.354   Root MSE        =    718.24

------------------------------------------------------------------------------
         bwt | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
         lwt |   4.429993   1.713244     2.59   0.010     1.050222    7.809763
       _cons |   2369.184   228.4671    10.37   0.000     1918.479    2819.888
------------------------------------------------------------------------------
```

The estimate for β_0 (2369.184) is found in the `_cons` line, and the estimate for β_1 (4.429993) is found in the `lwt` line. Confidence intervals for β_0 and β_1 are found in the two rightmost columns.

The estimated relationship is

$$\text{mean } \texttt{bwt} = 2369.2 + 4.43 \times \texttt{lwt}$$

A weight difference of 10 lb between two mothers corresponds to an expected difference in birthweight of 44.3 grams; the 95% confidence interval is [10.5, 78.1]. Under the model, that is, assuming linearity, the slope is significantly different from 0 ($t = 2.59$; $p = 0.010$). `_cons` is the constant or intercept, that is, the predicted outcome when all predictors are 0. Here it is the predicted birthweight (2,369 grams) for a child whose mother weighed 0 lb. Naïve extrapolations outside the observed ranges of the predictors obviously can lead to nonsense.

The estimated standard deviation of the error term, σ, is displayed in the upper-right block of output as `Root MSE`; here it is 718.24. From σ, we can express the variation around the estimated regression line as a 95% prediction interval using $\pm 1.96{\cdot}\sigma$, here ± 1408 grams. The `regress` command does not calculate the confidence interval around σ, but an unofficial program, `cisd`, does it for you. You can download the program by typing

```
. ssc install cisd
```

`cisd` is a postestimation command (see section 11.2), and it must be preceded by a `regress` command. Here we run it `quietly` because we have already seen the `regress` output:

```
. quietly regress bwt lwt
```

1. The `regress` command assumes that the precise value is observed. If you know only that the value lies within an interval, that is, you have interval-censored data, you can use `intreg`.

```
. cisd
SD(error): 718.23549
95% CI: ( 652.23371 ; 799.21579 )
```

In figure 11.1, a scatterplot with a regression line illustrates the association. You can find a do-file with the full graph command (`gph_fig11_1.do`) at this book's website, but here we show the minimum graph command:

```
. twoway (scatter bwt lwt) (lfit bwt lwt)
```

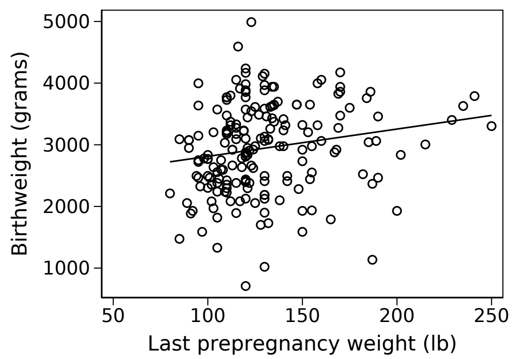

Figure 11.1. Scatterplot with a regression line

We can fit a multiple regression model, that is, a model involving more than one predictor:

$\texttt{bwt} = \beta_0 + \beta_1 \times \texttt{lwt} + \beta_2 \times \texttt{age} + \textit{error}$

```
. regress bwt lwt age
```

Source	SS	df	MS			
				Number of obs	=	189
				F(2, 186)	=	3.65
Model	3773616.52	2	1886808.26	Prob > F	=	0.0279
Residual	96141682.1	186	516890.764	R-squared	=	0.0378
				Adj R-squared	=	0.0274
Total	99915298.6	188	531464.354	Root MSE	=	718.95

bwt	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
lwt	4.181336	1.743425	2.40	0.017	.7419072	7.620765
age	7.971652	10.06015	0.79	0.429	-11.87501	27.81831
_cons	2216.218	299.2759	7.41	0.000	1625.807	2806.63

From this, we can find the estimated relationship:

$\text{mean } \texttt{bwt} = 2216.2 + 4.18 \times \texttt{lwt} + 7.97 \times \texttt{age}$

The interpretation is that for each pound of maternal prepregnancy weight, the mean birthweight increased by 4.18 grams when adjusted for maternal age.

11.2 Regression postestimation

One of the nice features in Stata is that the default output from a regression analysis is limited to the most essential information, but afterward, it is possible to supplement an analysis with additional information, such as calculating diagnostics (residuals, leverages, etc.), testing specific hypotheses, displaying variance inflation factors, and displaying correlations between estimates. This is done with postestimation commands, and we will show some examples. Read about postestimation commands in general by typing

```
. help estimation
```

and about specific commands by typing

```
. help regress postestimation
```

One of the assumptions behind the regression model above is that the error term should follow a normal distribution. This is validated by making a Q–Q plot or a histogram of the residuals. To do this, we need a new variable containing the residuals.[2] This is easily generated by typing

```
. quietly: regress bwt lwt age
. predict rbwt if e(sample), residual
```

The `predict` command will generate a new variable, `rbwt`, containing the residuals. We ran the `regress` command `quietly` because we had already recorded the output, but we wanted to make sure that we used information from the right regression model. The restriction `if e(sample)` ensures that the residuals are only calculated for the observations included in the preceding regression analysis. Now we can validate the assumption about normal errors with figure 11.2:

```
. histogram rbwt, normal name(p1)
. qnorm rbwt, name(p2)
. graph combine p1 p2
```

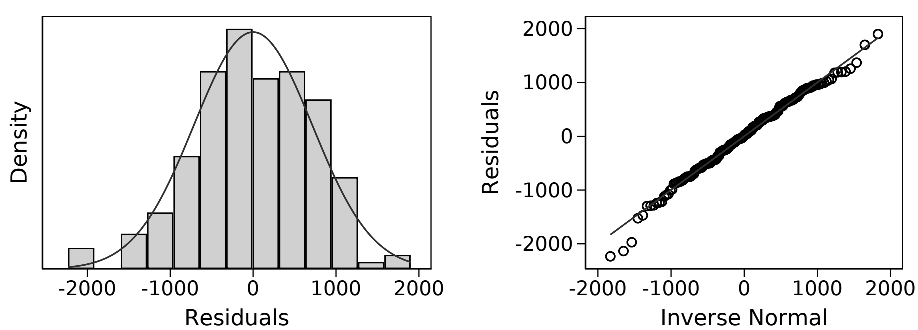

Figure 11.2. Histogram and Q–Q plot of residuals

The `predict` command can also generate predicted values (the `xb` option), standardized residuals (the `rstandard` option), and leverages (the `leverage` option). You can use these options

2. Ideally, you should check the assumption by plotting the standardized residuals, but we prefer to see the deviations on the original scale.

to make diagnostic plots such as residual versus fitted, residuals versus predictor, and leverage versus squared residual, but it is easier to use the postestimation plot commands that are already in Stata: `rvfplot`, `rvpplot`, and `lvr2plot`; see `help regress postestimation plots`. For the above model, we can make the diagnostic plots shown in figure 11.3 by using these commands:

```
. rvfplot, name(p1) yline(0)
. rvpplot lwt, name(p2) yline(0)
. rvpplot age, name(p3) yline(0)
. lvr2plot, name(p4) ylabel(0(0.25)1)
. graph combine p1 p2 p3 p4
```

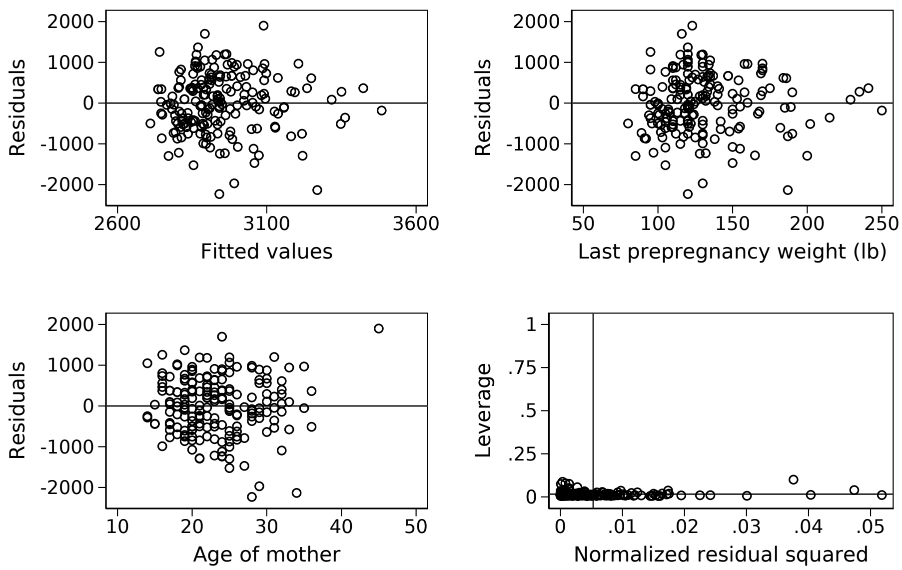

Figure 11.3. Diagnostic plots for `bwt` = β_0 + β_1×`lwt` + β_2×`age` + *error*

None of the first three plots indicate any serious problems with the assumption of linearity. The leverage-versus-residuals plot shows that no data points have especially high importance (high leverage) for the results and that no observed `bwt` differs extremely from the fitted value. Because figure 11.2 shows only a minor deviation from a normal distribution, we conclude that the linear regression model is appropriate.

Another postestimation command is `lincom`; it calculates linear combinations of the parameters in the model. For example, if we want to estimate the expected birthweight of a child born to a 25-year-old woman who weighed 125 lb, that is, $\beta_0 + \beta_1 \times 125 + \beta_2 \times 25$, we type

```
. lincom _cons + lwt*125 + age*25
 ( 1)  125*lwt + 25*age + _cons = 0
```

bwt	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
(1)	2938.177	56.33194	52.16	0.000	2827.045	3049.308

We could, of course, have found the 2,938.177 grams by hand, but `lincom` also supplies a 95% confidence interval. Here the test is of no interest because it evaluates the hypothesis that the birthweight for a child whose mother weighed 125 lb is 0 grams. If we had wanted to test the hypothesis that the expected birthweight is 3,000 grams, we could have written

```
. lincom _cons + lwt*125 + age*25 - 3000
```

Suppose that we want to estimate the difference in expected birthweight between two babies, the mother of one of them weighing 15 lb more and being 7 years older than the mother of the other baby. Doing the calculation by hand, we get $4.18 \times 15 + 7.97 \times 7 = 118.5$ (grams). Using `lincom`, we obtain

```
. lincom lwt*15 + age*7
 ( 1)  15*lwt + 7*age = 0
```

bwt	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
(1)	118.5216	70.5696	1.68	0.095	-20.6981	257.7413

The list of postestimation commands also includes `test` and `testparm` for testing specific hypotheses (see the next section for some examples), `margins` for calculating predictive margins,[3]

Use `pwcompare` for making pairwise comparisons between levels in a categorical variable, and `estat` for calculating the covariance matrix or the correlation matrix of the estimates. It is a good idea to consult the list of available postestimation commands for the specific type of model that you want to apply.

11.3 Categorical predictors—factor variables

Until now, we have considered continuous predictors like `age` and `weight`, but often we have categorical predictors, that is, variables that divide observations into categories, such as `race`:

3. The `margins` command is very versatile, but we chose not to include it in this book. See Mitchell (2021) for more information.

```
. tab1 race
-> tabulation of race

       Race |      Freq.     Percent        Cum.
------------+-----------------------------------
   1. White |         96       50.79       50.79
   2. Black |         26       13.76       64.55
   3. Other |         67       35.45      100.00
------------+-----------------------------------
      Total |        189      100.00
```

Categorical variables are stored in the dataset as numerical variables with which we associate labels; see section 6.1. `race` is on a nominal scale, and the codes themselves (1, 2, 3) are just codes, so there is no point in using them as values in any analysis. However, it is often vital that you know the actual codes, which may be displayed by typing `codebook race` or `label list`. In section 10.1, we used the `numlabel` command to add the codes to the value labels in the original `lbw.dta` dataset.

> As described in section 6.1, Stata most often displays value labels rather than codes, and if you specify the `nolabel` option, Stata displays the code but not the label. However, in regression commands and other estimation commands, the `nolabel` option is not allowed, but you can use `nofvlabel` instead.
>
> To display both the code and the label, you can use the `numlabel` command to include codes in the value labels. We did that when we generated the `lbw1.dta` dataset in section 10.1.
>
> ```
> . numlabel, add
> ```
>
> To hide the code again, type
>
> ```
> . numlabel, remove
> ```

You can work with categorical variables in several ways, all based on factor variables. Factor variables are virtual indicator variables that are generated from categorical variables. We (and Stata) use these terms:

- A *categorical variable* has a finite number of categories. To generate factor variables, a categorical variable must take only nonnegative integer values.
- An *indicator variable* only takes the values 0, meaning false, or 1, meaning true.
- A *virtual variable* does not really exist as part of the dataset, but in some situations, it appears to exist.

With some commands, we can see the corresponding factor variables by specifying an `i.` prefix:

```
. list race i.race in 1/3, nolabel

     +-------------------------------+
     |              1.     2.     3. |
     | race      race   race   race  |
     |-------------------------------|
  1. |    2         0      1      0  |
  2. |    3         0      0      1  |
  3. |    1         1      0      0  |
     +-------------------------------+
```

Here we show three different prefixes that can be used when specifying a categorical variable in a regression model:

```
. regress bwt i.race lwt
. regress bwt b2.race lwt
. regress bwt bn.race lwt, noconstant
```

In the first version, Stata automatically uses the first (lowest) level as the base or reference level.

We prefer specifying the base level explicitly in the regression command for the variables of primary interest, and we can do that by using the second version, where `b2.race` means that the category coded 2 will be the base level. We will use this principle in the book.

In the third version, `bn.race` specifies that we want no base level for `race`; this version also requires the `noconstant` option.

To illustrate, we will look at the simple model for birthweight with the predictors `lwt` and `race`, assuming the same slope for the three races as illustrated in figure 11.4. We chose to center `lwt` at 130 lb to have a sensible interpretation of the constant (`_cons`) in the model:

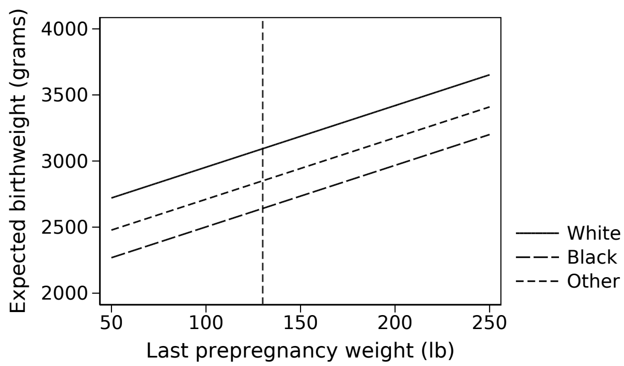

Figure 11.4. The same slope for all races; 130 lb is the reference level for `lwt`

If we want to use blacks (`race=2`) as the reference for a comparison of the three races, we fit the model with

```
. generate lwt130 = lwt - 130
. regress bwt b2.race lwt130, baselevels
```

(*output omitted*)

bwt	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
race						
1. White	452.0676	157.4976	2.87	0.005	141.3453	762.7899
2. Black	0	(base)				
3. Other	209.0862	168.9987	1.24	0.218	-124.3262	542.4986
lwt130	4.659211	1.749535	2.66	0.008	1.207607	8.110815
_cons	2641.382	140.9233	18.74	0.000	2363.358	2919.405

Displaying the base level in the regression output

The `baselevels` option displays a line for the reference (base) level in the regression output. You may also specify this as a permanent setting for future regression analyses:

```
. set showbaselevels on, permanently
```

We did this, and we recommend that you do it too.

This corresponds to the parameterization

$$\texttt{bwt} = \beta_0 + \beta_1 \times (\texttt{lwt} - 130) + \beta_2 \times \texttt{white} + \beta_3 \times \texttt{other} + \textit{error}$$

where `white` and `other` are indicator variables. From the output, we get the estimated line for blacks:

$$\text{Blacks:}\quad \text{mean } \texttt{bwt} = 2641 + 4.66 \times (\texttt{lwt} - 130)$$

Compared with the line for blacks, the line for whites is parallel, but it is shifted 452 grams upward, and the line for other races is shifted 209 grams upward. Assuming the same linear association between `lwt` and `bwt` for all races, the expected difference in birthweight between a child born to a white woman versus a black woman with the same weight is 452 grams with a 95% confidence interval [141, 763] grams. This difference is significantly different from 0, as can be seen from the confidence interval and the t test ($t = 2.87$; $p = 0.005$).

The difference between whites and other races is $452.0676 - 209.0862 = 242.9814$. You can use `lincom` to calculate this and the accompanying confidence interval; to do this, you must refer to the levels by the codes:

```
. lincom 1.race - 3.race
 ( 1)  1.race - 3.race = 0
```

bwt	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
(1)	242.9814	113.8326	2.13	0.034	18.40448	467.5583

You can produce a table of all the pairwise comparisons by `pwcompare`.

```
. pwcompare i.race
Pairwise comparisons of marginal linear predictions
Margins: asbalanced
```

	Contrast	Std. err.	Unadjusted [95% conf.	interval]
race				
2. Black vs 1. White	-452.0676	157.4976	-762.7899	-141.3453
3. Other vs 1. White	-242.9814	113.8326	-467.5583	-18.40448
3. Other vs 2. Black	209.0862	168.9987	-124.3262	542.4986

The output from the `regress` and `lincom` commands will only supply you with tests comparing two races. If you want to test the hypothesis of no overall difference between the three races, you can do this with the `testparm` command:

```
. testparm i.race
 ( 1)  1.race = 0
 ( 2)  3.race = 0

       F(  2,   185) =    5.17
            Prob > F =    0.0066
```

This command shows a statistically significant difference between the three races. With this model specification, the hypothesis corresponds to the coefficients of `white` and `other` both being zero.

If we want to focus on each of the races rather than differences between races, we can use the `bn.` prefix, indicating that we want no base level. Furthermore, we must use the `noconstant` option; otherwise, Stata will choose one of the races as the base to avoid redundant parameters:

```
. regress bwt bn.race lwt130, noconstant
      Source |       SS           df       MS      Number of obs   =       189
-------------+----------------------------------   F(4, 185)       =    833.71
       Model |  1.6470e+09         4   411739397   Prob > F        =    0.0000
    Residual |  91364383.3       185  493861.531   R-squared       =    0.9474
-------------+----------------------------------   Adj R-squared   =    0.9463
       Total |  1.7383e+09       189  9197470.74   Root MSE        =    702.75

------------------------------------------------------------------------------
         bwt | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
        race |
   1. White  |   3093.449   71.81421    43.08   0.000     2951.769    3235.129
   2. Black  |   2641.382   140.9233    18.74   0.000     2363.358    2919.405
   3. Other  |   2850.468   87.60896    32.54   0.000     2677.627    3023.309
             |
      lwt130 |   4.659211   1.749535     2.66   0.008     1.207607    8.110815
------------------------------------------------------------------------------
```

This corresponds to the parameterization

$$\texttt{bwt} = \beta_1 \times (\texttt{lwt} - 130) + \beta_2 \times \texttt{white} + \beta_3 \times \texttt{black} + \beta_4 \times \texttt{other} + \textit{error}$$

and the three estimated lines are

Whites: mean $\texttt{bwt} = 4.66 \times (\texttt{lwt} - 130) + 3093$

Blacks: mean $\texttt{bwt} = 4.66 \times (\texttt{lwt} - 130) + 2641$

Others: mean $\texttt{bwt} = 4.66 \times (\texttt{lwt} - 130) + 2850$

This parameterization of the model will not change the output from `lincom` or `pwcompare`, but `testparm` works differently. After a regression with the `bn` prefix and the `noconstant` option, `testparm i.race` will test the (pretty meaningless) hypothesis that all coefficients are equal to 0:

```
. testparm i.race
 ( 1)  1bn.race = 0
 ( 2)  2.race = 0
 ( 3)  3.race = 0

       F(  3,   185) = 1109.85
            Prob > F =    0.0000
```

To get the desired test for identical lines, we must use the `equal` option:

```
. testparm i.race, equal
 ( 1)  - 1bn.race + 2.race = 0
 ( 2)  - 1bn.race + 3.race = 0

       F(  2,   185) =    5.17
            Prob > F =    0.0066
```

An alternative way to work with categorical variables

Factor variables were introduced in Stata 11, but still in Stata 17, the `exlogistic` command and many user-written commands do not understand factor notation. For such commands, we can use the `xi:` prefix. The following syntax creates the new variables `_Irace_1`, `_Irace_2`, `_Irace_3`, `_Ismoke_1`, and `_Ismoke_2`. As you can see, the output is not too pretty, but it works:

```
. xi: regress bwt i.race i.smoke
i.race            _Irace_1-3          (naturally coded; _Irace_1 omitted)
i.smoke           _Ismoke_0-1         (naturally coded; _Ismoke_0 omitted)

      Source |       SS           df       MS      Number of obs   =       189
-------------+----------------------------------   F(3, 185)       =      8.69
       Model |  12346897.6         3  4115632.54   Prob > F        =    0.0000
    Residual |  87568400.9       185  473342.708   R-squared       =    0.1236
-------------+----------------------------------   Adj R-squared   =    0.1094
       Total |  99915298.6       188  531464.354   Root MSE        =       688

------------------------------------------------------------------------------
         bwt | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
    _Irace_2 |    -450.54    153.066    -2.94   0.004    -752.5194   -148.5607
    _Irace_3 |  -454.1813    116.436    -3.90   0.000    -683.8944   -224.4683
   _Ismoke_1 |  -428.0254   109.0033    -3.93   0.000    -643.0746   -212.9761
       _cons |   3334.858   91.74301    36.35   0.000      3153.86    3515.855
------------------------------------------------------------------------------
```

If we want blacks (`race` = 2) to be the future reference category, we omit it with this strange `char` command:

```
. char race[omit] 2
```

In section 11.4, we illustrate how to work with interactions in Stata (since version 12). To create an interaction term with `xi:`, use the `*` operator, as in `i.race*i.smoke`:

```
. xi: regress bwt i.race*i.smoke
i.race            _Irace_1-3          (naturally coded; _Irace_2 omitted)
i.smoke           _Ismoke_0-1         (naturally coded; _Ismoke_0 omitted)
i.race*i.smoke    _IracXsmo_#_#       (coded as above)
```

(*output omitted*)

bwt	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
_Irace_1	574.25	199.5008	2.88	0.004	180.6326	967.8674
_Irace_3	-40.26364	194.1079	-0.21	0.836	-423.2408	342.7135
_Ismoke_1	-350.5	275.4749	-1.27	0.205	-894.0152	193.0152
_IracXsmo_1_1	-250.8654	308.9992	-0.81	0.418	-860.5245	358.7938
_IracXsmo_3_1	293.4303	351.1314	0.84	0.404	-399.3562	986.2168
_cons	2854.5	170.8423	16.71	0.000	2517.426	3191.574

The indicator variables generated by `xi:` remain in the dataset. To clean up after this command, type (this assumes that you have no important variables starting with `_I`):

```
. drop _I*
```

11.4 Interactions in regression models

Factor variables and interactions in general

As demonstrated, Stata has a flexible way of specifying factor variables. Before we show examples with interactions, we will summarize how you work with categorical variables and interactions in Stata:

- A categorical variable is specified most commonly with an `i.` or a `b#.` prefix, where `#` is an integer value indicating the base level. For example, `b2.` defines the category coded 2 as the base level.
- The `bn.` prefix combined with the `noconstant` option indicates an analysis without the base level.
- A continuous variable is specified by the `c.` prefix, for example, `c.lwt130`. This prefix is needed only when the continuous variable is part of an interaction term.
- An interaction term (without main effects) is specified by one `#`, for example, `b2.race#c.lwt130` or `i.smoke#i.race`.
- An interaction with main effects is specified by `##`, for example, `b2.race##c.lwt130` or `i.smoke##i.race`.
- You can also use `#` to specify the product of two continuous variables, for example, `c.age#c.lwt130` or `c.lwt130#c.lwt130` (`lwt130` squared).

(Pages omitted)