Data Analysis Using Stata

Fourth Edition

Ulrich Kohler
University of Potsdam, Germany

Frauke Kreuter
LMU Munich, Germany

Anna-Carolina Haensch
LMU Munich, Germany

A Stata Press Publication
StataCorp LLC
College Station, Texas

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

Print ISBN-10: 1-59718-453-5
Print ISBN-13: 978-1-59718-453-3
ePub ISBN-10: 1-59718-454-3
ePub ISBN-13: 978-1-59718-454-0

Library of Congress Control Number: 2026943079

Contents

(Pages omitted)

Preface

Data Analysis Using Stata goes beyond simply teaching Stata commands—it guides you through all the essential steps of data analysis using practical, real-world examples. We explore both public topics, like income differences between men and women and elections, and more personal issues such as rent and living conditions. By focusing on common sense rather than diving into social science theories, we make the material accessible to a wider audience. Of course, these examples are stand-ins for the scientific theory behind data analysis, but this approach helps make the learning process smoother across different disciplines.

Whether you are a biometrician, econometrician, psychometrician, or some other kind of "metrician", this book is meant for anyone with an interest in data analysis. While we do not aim to cover every Stata command or statistical technique, we focus on providing a solid understanding of the essentials. By the end, you'll be equipped to tackle a wide range of data analysis challenges—not just in Stata but also in other languages.

We recommend that both beginners and more experienced readers start with the preface and the first chapter, "The first time". These sections offer helpful guidance for navigating the book. Beginners will benefit from working through the chapters sequentially, trying out the examples on their computers as they go. More advanced users can explore the index for quick tips or even challenge themselves to write their own commands. And for those who do not have access to Stata, or prefer a different software package, the data analysis methods discussed here can easily be adapted to other platforms.

Structure

Chapter 1, "The first time", shows what a typical session of analyzing data could look like. To beginners, this chapter conveys a sense of Stata and explains some basic concepts such as variables, observations, and missing values. To advanced users who already have experience in other statistical packages, this chapter offers a quick entry into Stata. They will find many cross-references within this chapter, which can therefore be viewed as an extended table of contents.

The rest of the book is divided into three parts:

Chapters 2–6 serve as an introduction to the basic tools of Stata. Throughout the subsequent chapters, these tools are used extensively. It is not possible to portray the

basic Stata tools, however, without using some of the statistical techniques explained in the second part of the book. The techniques described in chapter 6 may not seem useful until you begin working with your own results, so you may want to skim chapter 6 now and read it more carefully when you need it.

Throughout chapters 7–10, we show examples of data analysis. In chapter 7, we present techniques for describing and comparing distributions. Chapter 8 covers statistical inference and explains whether and how one can transfer judgments made from a statistic obtained in a dataset to something more than just the dataset. Chapter 9 introduces linear regression. It explains in general terms the technique itself, how to apply it in Stata, and why we need it. Afterward, we discuss how to test the statistical assumptions of the model. We conclude the chapter with a discussion of sophisticated regression models for causal inference. Chapter 10, in which we describe regression models for categorical dependent variables, is structured in the same way as the previous chapter to emphasize the similarity between these techniques.

Chapters 11 and 12 deal with more advanced Stata topics that beginners may not need. In chapter 11, we explain how to read and write files that are not in the Stata format. At the beginning of chapter 12, we introduce some special tools to aid in writing do-files. You can use these tools to create your own Stata commands and then store them as ado-files, which are explained in the second part of the chapter.

Changes in this edition

It has been quite some time since the third edition. Stata has undergone significant transformations since then, continuously evolving to meet the demands of modern data analysis. Each new version has introduced tools and functionalities aimed at improving user experience and analytical capabilities. We aim to reflect these in this new fourth edition. We are excited to announce the addition of a new coauthor, Caro Haensch, to this edition. Her expertise and fresh perspective have enriched the content, ensuring it is comprehensive and up to date with the latest developments in data analysis.

In this edition, we discuss several new features that enhance the practical application of data analysis:

Integration of Python. We illustrate how to incorporate Python scripts into your workflow for data analysis tasks. We show the interface between Stata and Python and discuss classic cases such as the integration of advanced machine learning methods, interactive graphs, and web scraping.

Large language models for coding. We explore the changes and pitfalls associated with using large language models for coding, providing practical advice on their effective use and understanding their limitations.

Reproducibility. We have added a section on dynamic documents (`dyndoc`) to enhance the reproducibility and documentation of data analysis workflows.

Missing values and Unicode. We now offer a more systematic approach to handling missing values and have included a new section on Unicode in the chapter on reading and writing data.

Causal inference, parsimonious description, and prediction. The regression chapter is restructured and extended to differentiate between different cases, such as causal inference, parsimonious description, and prediction.

Restructured book sections. The section on additional features is now consolidated in chapter 1, covering resources and information, updating procedures, and providing additional tools like the *Stata Journal* and Statistical Software Components Archive ado-files.

General updates. We updated and improved various sections where necessary.

Using this book: Materials and hints

The only way to learn how to analyze data is to do it. To help you learn by doing, we have provided data files (available on the internet) that you can use with the commands we discuss in this book. You can access these files from within Stata or by downloading a zip archive.

Please do not hesitate to contact us if you have any trouble obtaining these data files and do-files.

- If the machine you are using to run Stata is connected to the internet, you can download the files from within Stata. To do this, type the following commands in the Stata Command window (see the beginning of chapter 1 for information about using Stata commands):

```
. mkdir c:/data/daus4
. cd c:/data/daus4
. net from https://www.stata-press.com/data/daus4/
. net install data
. net get data
```

  These commands will install the files needed for the entire book except section 11.5. Readers of this section will need another large data package, namely, an extract of the database of the German Socio-Economic Panel (GSOEP). Because there are many files, you must download them in two steps. You can do that now, or later on, with

```
. mkdir c:/data/daus4/kkhsoep
. cd c:/data/daus4/kkhsoep
. net from https://www.stata-press.com/data/daus4/
. net get kkhsoep1
. net get kkhsoep2
. cd ..
```

If you are using a Macintosh or Unix system, substitute a suitable directory name in the first two commands.

- The files are also stored as a zip archive, which you can download by pointing your browser to https://www.stata-press.com/data/daus4/daus4.zip.

  To extract the file `daus4.zip`, create a new folder, `c:/data/daus4`, and copy `daus4.zip` into this folder. Unzip the file `daus4.zip` using any program that can unzip zip archives. Most computers have such a program already installed. If not, you can get one for free using the internet. Make sure to preserve the `kkhsoep` subdirectory contained in the zip file.

Throughout the book, we assume that your current working directory (folder) is the directory where you have stored our files. This is important if you want to reproduce our examples. At the beginning of chapter 1, we will explain how you can check where your current working directory is. Make sure that you do not replace our files with a modified version of the same file; avoid using the command `save, replace` while working with our files.

We cannot say it too often: the only way to learn how to analyze data is to analyze data yourself. We strongly recommend that you reproduce our examples in Stata as you read this book. A line that is written `in this font` and begins with a period (which itself should not be typed by the user) represents a Stata command, and we encourage you to enter that command in Stata. Typing the commands and seeing the results or graphs will help you better understand the text because we sometimes omit output to save space.

As you follow along with our examples, you must type all commands shown because they build on each other within a chapter. Some commands will work only if you have entered the previous commands. If you do not have time to work through a whole chapter at once, you can type the command

```
. save mydata, replace
```

before you exit Stata. When you get back to your work later, type

```
. use mydata
```

and you will be able to continue where you left off.

The exercises at the end of each chapter use either data from our data package or data used in the Stata manuals. StataCorp provides these datasets online.[1] They can be used within Stata using the command `webuse` *filename*. However, this command assumes that your computer is connected to the internet. Otherwise, you have to download the respective files manually from a different computer.

This book contains a lot of graphs, which are almost always generated with Stata. In most cases, the Stata command that generates the graph is printed above the graph,

1. https://www.stata-press.com/data/r19/.

but the more complicated graphs were produced by a Stata do-file. We have included all these do-files in our file package so that you can study these files in case you want to produce a similar graph (the name of the do-file needed for each graph is given in a footnote under the graph).

If you do not understand our explanation of a particular Stata command or just want to learn more about it, use the Stata `help` command, which we explain in chapter 1. Or you can look in the PDF. We refer to the manuals in the same way as the Stata `help` command: [R] **summarize** refers to the entry describing the `summarize` command in the *Stata Base Reference Manual*. [U] **18 Programming Stata** refers to chapter 18 of the *Stata User's Guide*. When you see a reference to a keyword in one of the *Reference* manuals, you can use the online help (see section 1.3.15) to get information on that keyword.

Teaching with this book

We have found this book to be useful for introductory courses in data analysis, as well as for courses on regression and on the analysis of categorical data. We have used it in courses at universities in Germany and the United States. When developing your own course, you might find it helpful to use the following outline of a course of lectures of 90 minutes each, held in a computer lab.

To teach an introductory course in data analysis using Stata, we recommend that you begin with chapter 1, which is designed to be an introductory lecture of roughly 1.5 hours. You can give this first lecture interactively, asking the students substantive questions about the income difference between men and women. You can then answer them by entering Stata commands, explaining the commands as you go. Usually, the students name the independent variables used to examine the stability of the income difference between men and women. Thus, you can do a stepwise analysis as a question-and-answer game. At the end of the first session, the students should save their commands in a log file, and as a homework assignment, they should produce a commented do-file (it might be helpful to provide them with a template of a do-file).

The next two lectures should work with chapters 3–5 and can be taught a bit more conventionally than the introduction. It will be clear that your students will need to learn the *language* of a program first. These two lectures need not be taught interactively but can be delivered section by section without interruption. At the end of each section, give the students time to retype the commands and ask questions. If time is limited, you can skip over sections 3.3 and 5.7, whereas you should make time for a detailed discussion of sections 5.4.3 and 5.4.4 and the examples in them. Both sections contain concepts that will be unfamiliar to the student but are very powerful tools for users of Stata.

One additional session should suffice for an overview of the commands and some interactive practice in the graphs chapter.

Two sessions can be scheduled for chapter 7. A reasonable discussion of statistical inference will take two sessions. The material provided in chapter 8 shows the necessary elements for simulations, which allows for a hands-on discussion of sampling distributions. The section on multiple imputation can be skipped in introductory courses.

Three sessions should be scheduled for chapter 9. According to our experience, even with an introductory class, you can cover sections 9.1, 9.2, and 9.3 in one session each. We recommend that you let the students calculate the regressions of the Anscombe data (see page 324) as a homework assignment or an in-class activity before you start the session on regression diagnostics.

We recommend that toward the end of the course, you spend two sessions on chapter 11, introducing data entry, management, and the like.

In addition to using this book for a general introduction to data analysis, you can use it to develop a course on regression analysis (chapter 9) or categorical data analysis (chapter 10). As with the introductory courses, it is helpful to begin with chapter 1, which gives a good overview of working with Stata and solving problems using the online help.

(Pages omitted)

In each case, you enter text in the parentheses. If the text consists of more than one line, each line must be entered in a separate set of quotation marks, as described in section 6.3.4. Here is an example:

```
. scatter rent size, title("Rent by home size")
    subtitle("Scatterplot")
    note("Data: GSOEP" "Randomized public use file")
    caption("This graph is used to demonstrate graph titles."
    "Please note how text is divided into several lines.")
```

You can use several suboptions to change the appearance of the title; for more information, see [G-3] ***title_options*** and [G-3] ***textbox_options***.

6.4 Multiple graphs

By "multiple graphs", we mean graphs that consist of different graph parts, in particular,

- `twoway` graphs that are plotted on top of each other,
- graphs that are broken out with the `by()` option and are then displayed together, and
- varying graphs that are combined using the `graph combine` command.

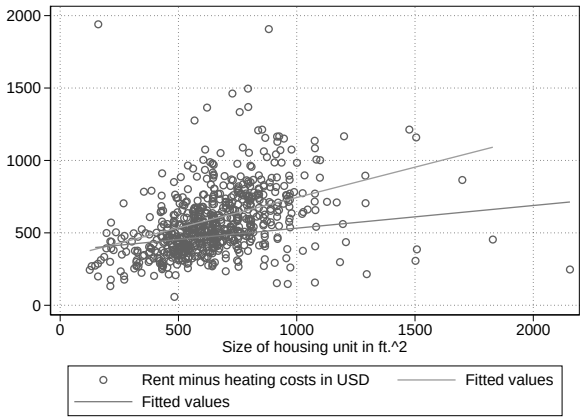
We will now quickly introduce these three types of graphs.

6.4.1 Overlaying many twoway graphs

You can overlay as many types of `twoway` graphs as you want in the same coordinate system. In the following example, three graphs are placed on top of each other: a

scatterplot; a linear fit (or regression line; see chapter 9) for the same data but restricted to the old federal states in West Germany (`rent_w`); and finally, a linear fit that is restricted to the new federal states in East Germany (`rent_e`).

```
. twoway || scatter rent size || lfit rent_w size || lfit rent_e size
```



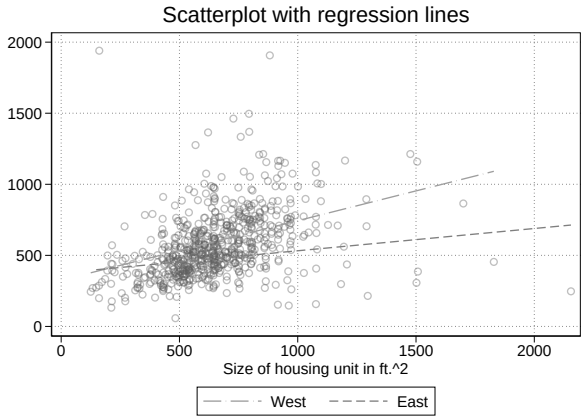
To overlay two-way graphs, you consolidate the graphs in a single `twoway` command, separated by parentheses or two vertical lines. In this book and our own work, we use two vertical lines because there are already so many parentheses in the graph syntax. The two vertical lines are particularly readable in do-files containing the individual graphs one after another with line breaks commented out (see section 2.2.2).

If you are combining several two-way graphs, you can specify options that correspond to the respective graph types, as well as two-way options that apply to all the graphs to be combined. Generally, the syntax for overlaid two-way graphs is as follows:

`twoway`
`||` `scatter` *varlist*`,` *scatter_options*
`||` `lfit` *varlist*`,` *lfit_options*
`||` *plottype varlist*`,` *plottype_options*`,`
`||` *twoway_options*

The first and the last two vertical bars are superfluous, but we use them to enhance readability. This syntax structure can be illustrated with an example (but note that commands of that length are often better typed into the Do-file Editor than into the command line):

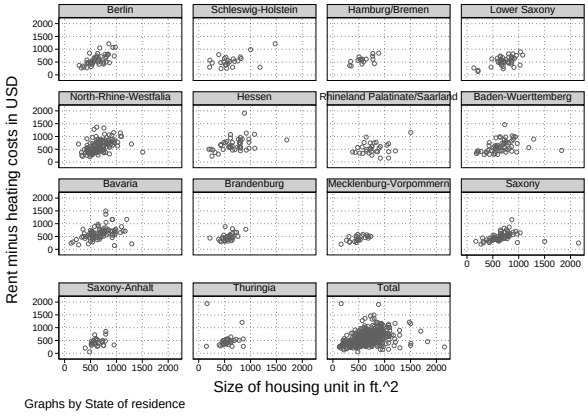
```
. twoway
    || scatter rent size, mcolor(%40)
    || lfit rent_w size, clpattern(longdash_dot)
    || lfit rent_e size, clpattern(dash)
    || ,  title("Scatterplot with regression lines")
       legend(order(2 "West" 3 "East"))
```



6.4.2 Option by()

The `by()` option displays separate graphs for each group defined by the variable in the parentheses. If more than one variable is entered in the parentheses, graphs are provided for every combination of the chosen variables. If you also specify the `total` suboption, another graph is displayed without separating it by group. Other suboptions control the positioning (for example, `rows()` and `cols()`); the display or omission of individual axes (for example, [`no`]`ixaxes`); or the appearance of the margins between the individual graphs. For the list of suboptions, see `help by_option` or [G-3] ***by__option***. One example should be enough at this point:

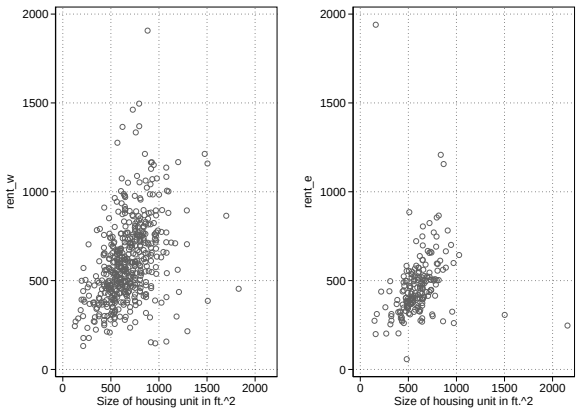
```
. scatter rent size, by(state, total)
```



6.4.3 Combining graphs

Stata allows you to combine as many graphs as you want into a joint graph. To do this, you first save the individual graphs and then combine them using `graph combine`. We will demonstrate this using a display of `rent` by `size`, separated by respondents from East and West Germany:

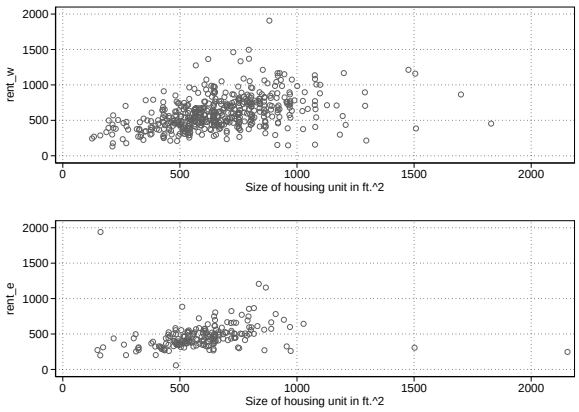
```
. scatter rent_w size, name(west, replace)
. scatter rent_e size, name(east, replace)
. graph combine west east
```



To save both graphs, we use the `name()` option, which specifies the name under which the graph will be stored in the computer's memory. The `replace` suboption tells Stata to delete any graphs already stored under this name. We then combine the two graphs using `graph combine`.

The `graph combine` command has a series of options for controlling how the combined graph is to be displayed. To begin with, it is important to set the number of rows and columns in the combined graph. The individual graphs are placed in the combined graph in rows and columns in a matrixlike fashion. The positioning of the individual graphs depends on how many rows and columns the matrix has. In the matrix above, one row and two columns were used. Here you will see what happens if we instead use two rows and one column:

```
. graph combine west east, rows(2)
```

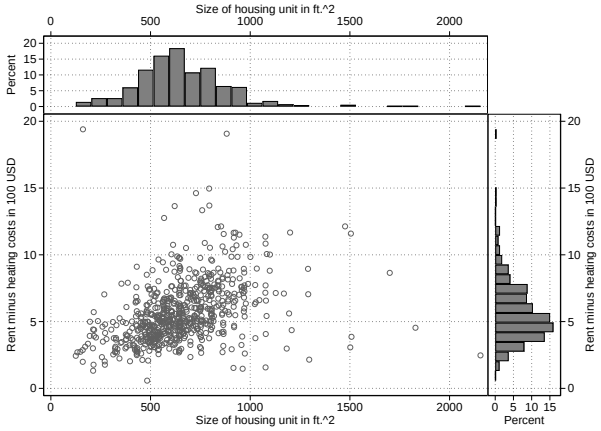


The number of individual graphs you can put in a multiple graph is limited only by printer and screen resolutions. If the main issue is the readability of the labels, you can increase their size with the `iscale()` option. The default font size decreases with every additional graph. With `iscale(1)`, you can restore the text to its original size; `iscale(*.8)` restores the text to 80% of its original size.

If you want the individual graph parts of the combined graph to have different sizes, you will have to save the graphs with different sizes before combining them. However, you cannot use the `xsize()` and `ysize()` options discussed in section 6.3.2, because these sizes are not taken into account by `graph combine`. Instead, you will have to use the forced-size options `fysize()` and `fxsize()`, which tell Stata to use only a certain percentage of the available space. For example, the `fxsize(25)` option creates a graph that uses only 25% of the width of the available space; correspondingly, a graph created using the `fysize(25)` option uses only 25% of the available height.

Here is a slightly more advanced example of `graph combine` using the forced-size options.

```
. replace rent = rent/100
variable rent was int now float
(677 real changes made)
. lab var rent "Rent minus heating costs in 100 USD"
. twoway scatter rent size, name(xy, replace)
    xlabel(, grid) ylabel(, grid gmax)
. twoway histogram size, name(hx, replace) percent
    xscale(alt) xlabel(, grid gmax) fysize(25)
. twoway histogram rent, name(hy, replace) percent horizontal
    yscale(alt) ylabel(0(5)20, grid gmax) fxsize(25)
. graph combine hx xy hy, imargin(0 0 0 0) hole(2)
```



For more details on creating such graphs, see [G-2] **graph combine**. The Graph Editor has further capabilities regarding the positioning of graph elements. The repositioning is best done with the Grid Edit Tool. A description of its functionality can be found in [G-1] **Graph Editor**.

6.5 Saving and printing graphs

To print a Stata graph, you type

```
. graph print
```

The graph displayed in Stata's Graph window is then printed directly. If you create many graphs in a do-file, you can also print them with that do-file by typing the `graph print` command after every graph command. The `graph print` command also has many options that can be used to change the printout of a graph (see `help pr_options`).

You can also print graphs that are not (or are no longer) in the Graph window, but you must first store the graph in memory or save it to a file on the hard drive. You already learned how to store graphs in memory in the previous section. There you stored numerous graphs with the `name()` option, which we then combined with the `graph combine` command. To print out these graphs, you first display them with `graph display` and then print them using `graph print`. Let us try this with the graph stored as `east` above (see page 166):

```
. graph display east
. graph print
```

Storing graphs in memory is often useful, but they are lost when you close Stata. To print graphs that you created a few days ago, you must have saved them as a file by using the `saving()` option. It works the same way as the `name()` option, except that it saves the file to the hard drive. You type the filename under which the file is to be saved in the parentheses. By default, Stata uses the `.gph` file extension. To overwrite an existing file of the same name, include the `replace` suboption in the parentheses. `replace` is commonly used when creating graphs with do-files.

Be careful when you use the Graph Editor. When you exit the Graph Editor, you will be asked if you want to save the changes you made to the graph. In most cases, you would want to answer yes. If you do so, you should make sure that you use a new filename. If you save the graph with its old name, you might be in trouble the next time you run the do-file that created the graph originally. Running the do-file will re-create the original graph and therefore overwrite the changes you made with the Graph Editor. You would need to start over.

All saved files can be, at a later point in time, printed or edited with the Graph Editor. To do so, you simply call the saved graph on the screen with the command `graph use` and start printing or editing. For example, a command such as

```
. graph combine hx xy hy, hole(2) imargin(0 0 0 0) saving(combined, replace)
```

saves the graph under the name `combined.gph` in the current working directory. If a file with the same name already exists in the working directory, it is overwritten. You can now close Stata, shut down the computer, or display another graph,

```
. graph display east
```

Regardless of what you do, you can print the graph saved in a file by typing

```
. graph use combined
. graph print
```

Finally, you will usually be exporting a Stata graph to a word processing program or presentation program rather than printing it. For this, you can use the much-loved copy-and-paste procedure, where you first copy the graph displayed in a Graph window and then paste it into the respective document. However, if you are dealing with several

graphs, we strongly recommend to save the graph in a suitable file format on the hard drive and then import to the desired program when needed.

To save a graph in a different file format, use the `graph export` command. You type the filename with the appropriate file extension after the command. Table 6.1 lists the formats that are available to you.

Table 6.1. Available file formats for graphs

Extension	File format	Restriction
Vector graphs		
`.ps`	PostScript	
`.eps`	Encapsulated PostScript	
`.svg`	Scalable Vector Graphic	
`.emf`	Enhanced Metafile	Stata for Windows only
`.pdf`	Portable Document Format	
Bitmaps		
`.png`	Portable Network Graphic	
`.tif`	Tagged Image File Format	
`.gif`	Graphics Interchange Format	Stata for Mac only
`.jpg`	Joint Photographic Experts Group	

Windows-applications use EMF when you copy-and-paste graphs from one window to another, but these files are not well transportable to other operating systems. Similarly, PostScript and Encapsulated PostScript are widely used on Unix systems but require additional software on standard Windows systems. Both PostScript and Encapsulated PostScript also remove transparency from the graphs, which limits their usability. However, the most important difference between graph formats is the difference between vector and bitmap formats. Vector graphs are scalable without loss of resolution, while bitmaps lose resolution if you magnify their size. With bitmaps, one can optimize the file size using a resolution that produces a satisfying result for a given device, while the file size for vector graphs remains the same, no matter what size of the graph is required. In situations when a graph should be inserted in a document that later might be both printed and read on screens of variable sizes, a vector format is usually preferable. Thus, we use PDF or SVG most of the time:

```
. graph export mygraph1.pdf
```

Use the other file formats as you wish.

6.6 Graphs with Python (within Stata)

Stata graphs in general, and particularly `graph twoway`, provide powerful tools for designing exactly the graph that you have in mind. However, when it comes to three-

dimensional graphs and interactive graphs, Stata's graph capabilities are less well suited. In such situations, you may want to create graphs with Python—which works well from within Stata. In the following final section of this chapter, we will be using a three-dimensional interactive scatterplot of the variables `income`, `yedu`, and `ybirth` to illustrate how this works. If you are unfamiliar with the integration of Python into Stata, see section 4.3.

We start by creating a small subset of our dataset using standard Stata commands:

```
. use data1, clear
. keep if !mi(income,yedu,ybirth) & emp==1
. keep if inrange(income,500,20000)
```

After creating the data in Stata, we load the necessary modules into Python as shown in section 4.3. In the following, we require the data class from the Stata Function Interface, as well as the class `pyplot` from the `matplotlib` module, which should be readily available in most Python installations:

```
. python
──────────────────────────────────────────────── python (type end to exit) ─────
>>> from sfi import Data
>>> import matplotlib.pyplot as plt
>>> end
──────────────────────────────────────────────────────────────────────────────
```

Note that we have aliased `pyplot` from `matplotlib` as `plt` for easier reference. This way, we can later simply type `plt()` instead of `matplotlib.pyplot()`.

Once done, the first step is to export the Stata variables to be used for the figure into Python. As recommended in section 4.3, we use the `Data.get()` method from the `sfi` module to export the variables from Stata into Python:

```
. python
──────────────────────────────────────────────── python (type end to exit) ─────
>>> income = Data.get('income')
>>> yedu = Data.get('yedu')
>>> ybirth = Data.get('ybirth')
>>> end
──────────────────────────────────────────────────────────────────────────────
```

Next, we use Python's `pyplot` method to draw the graph. For this, we first create an empty graphics window and give it a name so that we can easily fill it with contents:

```
. python
──────────────────────────────────────────────── python (type end to exit) ─────
>>> fig = plt.figure()
>>> end
──────────────────────────────────────────────────────────────────────────────
```

Next, we add a subplot into the empty graphics window. The subplot is a three-dimensional projection of three axes. As before, we give the subplot a name so that we can later fill this projection with contents:

```
. python
------------------------------------------------------ python (type end to exit) ---
>>> axes = fig.add_subplot(projection='3d')
>>> end
-------------------------------------------------------------------------------
```

Finally, we use the method `scatter()` to draw data points into the space of the axes. Having done this, we use `plt.show()` to show the graphics window (see figure 6.4):

```
. python
------------------------------------------------------ python (type end to exit) ---
>>> axes.scatter(income,yedu,ybirth,marker='o')
<mpl_toolkits.mplot3d.art3d.Path3DCollection object at 0x7f6b2f130a58>
>>> plt.show()
>>> end
-------------------------------------------------------------------------------
```

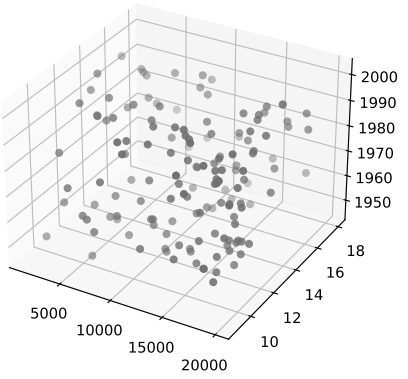


Figure 6.4. Three-dimensional scatterplot created with Python from within Stata

Note that Stata does not accept any commands as long as the graph is being shown. However, before you close the Graph window, you should realize that you can touch the figure with the mouse and turn the projection around.

We stress that depending on your operating system and Python installation, the above commands may fail on your computer. Note that lower-level languages like Python may have compatibility issues, which is why higher-level applications like Stata are often used. But those problems are solvable both on your computer and also in general. However, the solutions and generalization are outside the scope of this book; we thus refer you to the fine manual of `matplotlib` for further information: https://matplotlib.org/stable/users/index.html.

6.7 Large language models as programming help with graphs

We like to use large language models (LLMs) as support for data visualization. In our experience, they often suggest suitable options for various graph elements, such as titles, labels, and colors—all those little details that might be a little bit cumbersome to set up.

Let us try it out by creating a scatterplot. We use some of the tips and tricks we discussed in section 1.3.15.

I need help with data visualization in Stata. Specifically, I aim to create a scatterplot with ***size*** *plotted against* ***hhinc*** *from the dataset* ***data1****, and this should be combined with a line plot showing predicted values (****sizehat****) against* ***hhinc****, effectively overlaying the regression line on the scatterplot.* ***sizehat*** *has to be first created. Please add sensible titles, subtitles, and labels. The code should be concise, well-commented, easy to understand, and reusable. It is crucial that it is efficient and adheres to best practices.*

In the provided prompt, we clearly defined our goal: creating a scatterplot with a regression line in Stata (see chapter 9; for now, it is enough to know that it summarized the relationship between the two variables) and specifying the variables and the dataset. We include the relevant variables and emphasize the need for the creation of a `sizehat` variable. While we do not share existing code, we do request specific features for the code, like conciseness and good commenting practices.

In one instance, we got the following output:

begin ChatGPT-Output

```
  * Load your dataset
use data1

* Perform linear regression of size on hhinc
regress size hhinc

* Generate predicted values (sizehat) based on the regression
predict sizehat, xb

* Overlay the line plot of predicted values (sizehat) against hhinc
* on the scatterplot
twoway (scatter size hhinc) (line sizehat hhinc, sort), ///
    title("Scatterplot with Regression Line") ///
    xtitle("Household Income (hhinc)") ///
    ytitle("Size") ///
    legend(label(1 "Actual Size") label(2 "Predicted Size"))
```

end ChatGPT-Output

Running the code is producing exactly what we asked for; it makes use of the compact `twoway` command and provides us with propositions for the titles and the legends. Even if we choose not to use those exact titles and labels, it offers us a template that we can easily adapt.

We would like to remind you that obtaining favorable results like the one presented is not guaranteed and even this result is not completely flawless. For example, the option

`xb` after `predict` in line 8 is superfluous because this is already the default. Thus, we would not use it if we had written the code ourselves.

If you already have decided on a plot command, you can also ask the LLM to provide some options to tweak the plot you are creating, for example, like this:

I want to make this plot [`twoway (scatter size hhinc) (line sizehat hhinc, sort)`] more informative and visually engaging. Could you suggest some sensible optional arguments for the `twoway` command in Stata that would add meaningful titles, captions, and customized colors? I'm aiming for a clear, professional-looking plot that effectively communicates the data's story, with thoughtful use of color, labeling, and annotations to guide the viewer's understanding.

This leads in our experience to similar, if not more extensive, results as above, giving you an easy template to further improve.

6.8 Exercises

1. Load data from the National Health and Nutrition Examination Study (NHANES) using the command below:

   ```
   . webuse nhanes2.dta, clear
   ```

2. Using the NHANES data, produce a scatterplot of weight in kilograms by height in centimeters. Use hollow circles as the marker symbol.
3. Change the title of the vertical axis to "Weight (in kg)", and add a note for the data source of your graph.
4. Add reference lines to indicate the arithmetic means of weight and heights.
5. Add an axis label to explain the meaning of the reference lines.
6. Use blue marker symbols for male observations and pink marker symbols for female observations, and construct a self-explanatory legend. Remove the reference lines.
7. Plot the data for men and women separately, and produce a common figure of both plots placed on top of each other. Take care that the note on the data source does not appear twice in the figure.
8. Construct a graph similar to the previous one but this time with reference lines for the gender-specific averages of weight and height.
9. Classify the variable holding the body mass index (BMI) according to the table on page 130. Change the previous graph so that the colors of the marker symbols represent the categorized BMI.
10. Add the unique person identifier (`sampl`) to the symbols for the male and the female observations with the highest BMI.
11. Export your graphs so that they can be imported into your favorite word processing program.