Publication quality tables in Stata

User Guide for *tabout*

Version 3

Ian Watson

Numerous people have provided feedback, advice and contributions over the years and I am very grateful for their assistance. In particular I'd like to thank: Mitch Abdon, Anders Alexandersson, Ulrich Atz, JP Azevedo, Kevin Baier, Kit Baum, Megan Blaxland, Eric Booth, Ulrich Brandt, Simon Coulombe, Enzo Coviello, Nick Cox, Janelle Downing, Anwar Dudekula, Axel Engellandt, David L. Eckles, Richard Fox, Jonathan Gardner, Johannes Geyer, Bill Gould, Daniel Hoechle, Ben Jann, Stephen Jenkins, Stas Kolenikov, Floris Lazrak, Thomas Masterson, Scott Merryman, Gilbert Montcho, Nirmala Devi Naidoo, Cathy Redmond, Bill Rising, Mikko Rönkkö, Rafael Martins de Souza, Benjamin Schirge, Urvi Shah, Arjan Soede, Tim Stegmann, Herve Stolowy, Amanda Tzy-Chyi Yu, Chris Wallace, Joy Wang, Peter Young and Tony Young.

First printing, December 2016
Second printing, May 2017
Third printing, October 2017
Fourth printing, October 2018
Fifth printing, March 2019
Sixth printing, April 2019

Typeset with LaTeX

tabout version 3 beta is available from the website
http://www.tabout.net.au.

tabout version 2 is still available at
http://www.ianwatson.com.au/stata.

All enquiries to mail@ianwatson.com.au

Contents

Part I.

Publication quality tables

1. Introduction

What is **tabout**? In essence, **tabout** is a program for producing publication-quality tables of descriptive statistics. It runs inside the statistical package, *Stata*, and can be used to produce cross-tabulations of counts and percentages, as well as cross-tabulated summary tables, such as means, medians, standard deviations and so forth.

Stata is a commercial statistical package available from StataCorp (see Stata home page for more details). **tabout** is a free user-written program, called an **ado** file. Version 2 is available from inside Stata by typing: `ssc install tabout` and Version 3 beta is available from the tabout home page

Traditionally, tables produced by *Stata* have rarely been suitable for inclusion in publications and required considerable subsequent editing to make them acceptable. Over the years, many *Stata* users have written **ado** files to overcome these difficulties, with the **estout** program written by Ben Jann one of the earliest and best examples of these endeavours. Where **estout** specialises in producing tables based on estimates, such as the results of regression models, **tabout** is focussed on descriptive statistics and brings 'under one roof' a number of underlying Stata commands.

These commands include **tabulate**, **summarize**, and various **svy** commands. The output from **tabout** can also emulate the results of **table** and **tabstat**.

1.1. Output

The output from **tabout** can be exported directly to typesetting systems like LaTeX, to spreadsheets like *Excel* and to word processors like *Word*. The output can also be exported as *html*, the language of the World Wide Web, and a particularly useful format for exchanging documents across other applications, including word processors and eBook readers. For a glimpse of the kinds of tables which **tabout** produces, the reader may care to jump ahead to Chapter 3.

LaTeX home page

Other spreadsheet and word processor applications like Open Office and Libre Office will also open this output, though the ability to create dynamic documents is limited with these applications. See below for a discussion on this.

By way of a brief history, I wrote the first version of **tabout** in 2004 for Release 8 of *Stata*. In 2006 I rewrote **tabout** (as Version 2) to work with Release 9.2 of *Stata*, a release which had provided `ado` writers with a new and powerful matrix programming language, called `Mata`. Version 2 (with small refinements) circulated for 10 years without any major changes. During 2016 I rewrote **tabout** (as Version 3) to make further use of `Mata`, particularly the new features of *Stata* Release 13 which made possible direct export of tables to `docx` and `xlsx` files (that is, the native file formats of *Word* and *Excel*). Version 3 enhancements include style output options for *docx* and *xlsx*. In addition, Version 3 of **tabout** also utilised `Mata`'s flexibility to add new reshaping capabilities, such as dropping unwanted columns and rows, and 'plugging' gaps in tables. Many of the other new

If you are wondering about the font conventions here, please note that when referring to choices in a **tabout** option, I use this *font*, and when referring to file formats etc I use this `font`.

enhancements to Version 3 of **tabout** are flagged in orange text in the Options table at the end of the *User Guide*. Blue text throughout the *User Guide* indicates a link.

1.2. This guide

There are four parts to this guide:

- **Part I**: this introductory chapter and a second chapter discussing the principles of good tables and sound research methods.
- **Part II**: the *content* of the tables which shows you how to get what you want in the cells of the table. This includes the headings, title and footnotes, as well as sample counts and table statistics, in you've chosen these. These chapters also discuss layout and re-arranging tables as well as the use of *template files*. These are an important new features of Version 3, and they greatly simplify the usage of **tabout**, particularly for novices. Finally, this part of the guide also shows how you can create *dynamic documents* using **tabout**: this means that short reports or web pages can be automatically produced from inside *Stata*.
- **Part III**: the *style* of the tables: how you get your tables to look the way you want. This covers fonts, borders and lines. It also deals with the different output styles: files suitable for LaTeX, *Word*, *Excel*, and `html`.
- **Part IV**: the *key details* of using **tabout** in a useful reference format. All of these details are presented alphabetically and thematically. There are numerous cross-links in this *User Guide*: links from the text to this reference table, and links in this reference table back to the tables and code in the main parts of the *User Guide*.

I refer to *Word* and *Excel* throughout this *Guide*. This is *not* an endorsement. As a LaTeX user, I don't make use of these applications myself. But I use the terms here for ease of expression and because they have become generic. Also, most *Stata* users probably use one or both of them. I make mention below of other applications which are substitutes.

If you don't want to read the whole of the *User Guide*—and it is quite long—you can just look at the example files and read the section for the particular output style you want to use. I would, however, strongly recommend that you read the overview chapters: *Table content: overview* and *Table styles: overview*. They contain some core information which you may miss if you just dip into various parts of the *User Guide*.

Throughout this guide you will see links, which are mostly coloured blue. Another set of links are in a variety of colours and are placed above the table example code. These are links to *Stata* do files on the **tabout** website and they are available for each of the different output styles which **tabout** supports. While they closely resemble the example code reproduced in this *User Guide*, they differ slightly according to their specific requirements. You can also download these example files as a single *Stata* do file from the following links:

For *Stata* novices, the /// symbols in all the example code indicate that the command continues onto the next line.

- txt which is for tab delimited output (and can be used for opening in *Word* and *Excel* using `xls` and `doc` extensions. More on this later).
- tex which is for LaTeX output;
- docx which is for native `docx` output;
- xlsx which is for native `xlsx` output;
- htm which is for `htm` or `html` output, that is, web page output.

If you not sure about the difference between *`xls`* and *`xlsx`*, or between `doc` and *`docx`*, see the discussion on these style outputs in Chapter 10 (*Table styles: overview*). Also don't overlook the great advantages in using *`htm`* output for your *Word* documents. This is also discussed later in this *User Guide*.

I use *`htm`* and *`html`* interchangeably throughout this *Guide*, though I try to use the former for the output style and the latter for the file type. I also use *`tex`* and LaTeX interchangeably, with the former for the output style and the latter for everything else!

In the interests of security, all of the downloadable files have been given a `txt` extension, even though they are mostly `do` files. These will open in your browser, where you can inspect the contents, and from where you can copy and paste them into a `do` file. Alternatively, you can save them to your computer and give them the correct `do` extension so they will run in *Stata*.

Many of the example files use some built-in *Stata* data sets and two of these require some initial setup code. These setup do files are available from the links below:

- nlsw data set
- cancer data set

Within the margins of this guide you will see ⟹ symbols, which are links to the `alphabetical listing` for a particular option mentioned at that place in the paragraph and shown in a typewriter font in a greenish colour. Depending on your PDF viewer, you may also see a preview of the option if you hover over the link. All PDF viewers should jump to the option when you click on the link and most PDF viewers have a *back* button so you can return to where you left from. ⟹

The typeface in this *Guide* follows some common programming conventions:

- `teletype text` (in whatever colour) is used for words you can type, such as commands;
- when the words are coloured `this shade of green` they are a **tabout** option;
- when the words are also *`slanted`*, they refer to the actual arguments inside the options.

The next chapter is more philosophical than technical, outlining the ideas behind the design of **tabout**. If you are in hurry to produce some tables, you might consider coming back to that chapter later. If you are interested in producing *good* tables, and you are not too impatient, you might like to read it now. If you are also concerned with optimising your workflow, and using **tabout** for reproducible research as a sound research method, you should definitely find time to read it.

1.3. Installation

Downloading and installing tabout

When this version of **tabout** becomes the official version, it will be available from the SSC archives using the command (inside *Stata*): `ssc install tabout, replace`. At present, issuing this command will install Version 2 of **tabout**.

The Version 3 beta, which is available from the tabout home page should be installed manually, by copying the file into a suitable directory. You can either place it directly in your working directory, or in the `/ado/personal/` directory. If you already have **tabout** Version 2 installed, it will most likely be located in the `/ado/plus/` directory and, because the personal directory has a higher priority in *Stata*'s adopath, the Version 3 of

tabout will execute before Version 2. If you want to find the location of these directories on your computer, just type (inside *Stata*): `adopath`.

If you are unsure which version of **tabout** is the one being used by *Stata* you can type (inside *Stata*): `which tabout`. You should also note the version number because, each time the beta is updated, the numbering will change, and you should keep an eye on the tabout home page to make sure you have the latest version.

Multiple versions

In an ideal world, there would be just one `tabout.ado` file for **tabout** Version 3. Unfortunately, there are three versions on the **tabout** website, and which one you download depends on which version of *Stata* you are running.

- If you are running *Stata 14.2* or later, the **main version** of **tabout** is for you. If you have downloaded the correct version of **tabout**, when you type `which tabout` inside *Stata* you should see on the second line: `*! Stata 14.2 (or later) version`. With this version, all the features of the *docx* and *xlsx* styles are available to you.
- If you are running *Stata 13.1*, the **B version** of **tabout** is for you. If you have downloaded the correct version of **tabout**, when you type `which tabout` inside *Stata* you should see on the second line: `*! Stata 13.1 version`. While this version of **tabout** supports both *docx* and *xlsx* styles, not all the features are available (to do with fonts and borders, for example). The reason for this is that *Stata* introduced additional features for these Mata functions in Version 14.
- If you are running versions of *Stata* earlier than 13 (but later than 9), then the **C version** of **tabout** is for you. If you have downloaded the correct version of **tabout**, when you type `which tabout` inside *Stata* you should see on the second line: `*! Versions of Stata before Stata 13`. With this version of **tabout** you are not able to use the *docx* and *xlsx* styles. Keep in mind though that you can get very high quality *Word* documents from **tabout** using the *htm* style, and there is extensive discussion of this strategy in this *User Guide*.

Needless to say, whichever version of *Stata* you are using, you should always use the latest update for that version by typing `update all` if required. Note that error messages about 'undefined' functions usually indicates that you are running the wrong version of **tabout** for your version of *Stata*.

Running tabout

You issue the **tabout** command just like any other *Stata* command. The syntax is shown in the syntax section of Chapter 16.

2. What makes for good tables

A good table tells an interesting story. A good table is usually a combination of useful information and appropriate aesthetics. Aesthetics here is more than just 'beauty' but deals with how information is encoded and decoded. To appreciate this we need to briefly consider how graphics are used in statistics. There is a large body of literature which deals with this theme, particularly the works of William Cleveland[1] and Edward Tufte[2].

1. **Cleveland1993**; **Cleveland1994**.

2. **Tufte2001**; **Tufte1990**; **Tufte1997**; **Tufte2006**.

2.1. Encoding and decoding

As William Cleveland has shown, researchers *encode* data and readers *decode* information. How well that communication process works depends on how well the graphic (or table) is constructed. Edward Tufte has outlined a number of 'principles of graphical excellence' which apply equally to tables as much as to graphs. These include:

- the *goal* is the well-designed presentation of interesting data, which will always be a combination of *substance*, *statistics* and *design*;
- the *task* is to communicate complex ideas with *clarity*, *accuracy* and *efficiency*;
- in practice, this means giving viewer the greatest number of ideas in the shortest time with the least ink in the smallest space; and
- the table (or graph) is nearly always multivariate.

In Tufte's view, many modern graphs, particularly business-type graphs, are full of 'chart junk'. For Tufte it is important to maximise the data component and minimise the decorative junk. These sentiments apply equally to tables. In the discussion of his LaTeX package, **booktabs**,[3] Simon Fear endorses these sentiments, advocating that one should never user vertical lines nor double lines. The **booktabs** package is implemented in **tabout** and provides users who produce LaTeX output with finely-tuned settings in their tables, such as variable row spacing for headings and appropriate thicknesses in the horizontal lines (called 'rules'). Careful study of the tables in the following chapters will illuminate these subtleties.

3. **Fear2003**.

2.2. How tabout implements these principles

The main way in which **tabout** implements these principles of graphical excellence is through the use of *panels*. While simple two-way tables are

possible, **tabout** encourages users to build complex tables in which a series of two-way tables are assembled into a single vertical table.

Table A.23: Household financial stress—C10 ‡

	Household comparisons					
	Adult low paid		Other		All households	
	'000s	%	'000s	%	'000s	%
Family finances: optimists						
Poor or very poor	20	1.6	44	1.2	64	1.3
Just getting along	285	23.8	720	19.0	1,005	20.1
Reasonably comfortable	645	53.9	2,039	53.7	2,684	53.8
Prosperous or v comfort	246	20.6	991	26.1	1,237	24.8
Total	1,196	100.0	3,793	100.0	4,990	100.0
Family finances: pessimists						
Poor or very poor	46	3.8	104	2.8	150	3.0
Just getting along	401	33.5	1,054	27.8	1,454	29.1
Reasonably comfortable	645	53.9	2,097	55.3	2,742	55.0
Prosperous or v comfort	105	8.8	539	14.2	644	12.9
Total	1,196	100.0	3,793	100.0	4,990	100.0
Episodes of financial hardship						
Three or more	135	11.3	295	7.8	430	8.7
Two	115	9.7	282	7.5	397	8.0
One	160	13.4	509	13.5	668	13.5
None	781	65.6	2,691	71.3	3,472	69.9
Total	1,191	100.0	3,776	100.0	4,967	100.0
How easily raise $2000 in one week						
Could not raise it	244	20.4	481	12.7	725	14.6
Have to do something drastic	194	16.2	399	10.5	593	11.9
Raise it, but some sacrifices	321	26.8	949	25.1	1,270	25.5
Easily raise it	436	36.5	1,956	51.7	2,393	48.0
Total	1,196	100.0	3,785	100.0	4,981	100.0
Ownership of credit card						
No credit card	453	34.1	999	23.7	1,452	26.2
Owns credit card	876	65.9	3,210	76.3	4,086	73.8
Total	1,330	100.0	4,209	100.0	5,538	100.0
Sample size	1,200		3,849		5,049	

Notes: First two panels: self-perceptions of financial prosperity. Optimists and pessimists result from differing evaluations by first two members of household. Counts are lower in this table because of missing observations. Third panel: episodes of financial hardship. Since beginning of year have any of following happened (due to lack of money): not pay utility bills on time; not pay rent or mortgage on time; pawned or sold something; went without meals; unable to heat home; asked for financial help from family or friends; asked for help from welfare organisation. Fourth panel: worst situation reported by at least one person in household. Fifth panel: no credit card = no one in household had a credit or charge card or store account; credit card = at least one person had one. Weighted by cross-sectional household population weights. Definition of low pay: earning at or below $15.94 per hour.
Population: Adult = Households with at least one adult low paid employee; Other = Households with at least one employed person (excluding Adult etc); All = Households with at least one employed person. Data from Wave 5 (2005).
Source: HILDA Release 5. †Responding person survey form; ‡Responding person self-completion survey form; §Household survey form.

Figure 2.1.: Reduced view of a typical full-page **tabout** table, showing multiple panels.

The preference for a vertical layout is obvious: most books are printed in 'portrait' mode and the psychology of reading is for the eye to move down the page within a narrow visual band. Comparisons between percentages are easy when the data is adjacent; inclusion of the sample size makes it easier to appreciate the precision in the data; footnotes can inform the reader about the source of the data, the population and relevant definitions.

2.3. Reproducible research

While the aesthetic aspects of table production assist with achieving *clarity* in communication, the key to *accuracy* and *efficiency* lies in reproducible research. All good research should leave a trail, the breadcrumbs which allow yourself (12 months later), or your peers, to know how you got your results. Researchers should adopt the slogan 'copy and paste is my enemy' and should adapt the practice of 'files talking to files'. This is the basis of

Stata's own emphasis on `do` files, and the idea of building a nested `do` file system, where one `do` file calls other `do` files. For example:

- running master.do → (produces) the final tables and/or the final report
- master.do itself is made up of a set of smaller files:
  - raw.dta → clean.do → clean.dta
  - clean.dta → recode.do → final.dat
  - final.dta → tables.do → actual table files

This user guide has been written using LaTeX and the earlier version of **tabout** emphasised LaTeX output. Apart from the beauty of its typography, a major advantage of LaTeX is that it greatly facilitates reproducible research. For example, one runs a *Stata* `do` file like master.do shown above, and using programs like **tabout** and **estout** one produces a set of `*.tex` files. Then, with a simple `\input` command in the main `tex` document, the tables (as well as other material, such as graphs) are compiled at the same time as the main text in the document is compiled. The result, after a matter of a few seconds, is a complete publication-quality document. One can even include commands in the master.do file which compiles the main LaTeX file. Thus as soon as one runs the *Stata* `do` file, all editing is finished; no further tweaking of the tables is needed; the final document is ready for printing or electronic distribution.

The efficiency of this system is obvious, but the accuracy is also paramount. It is possible, by inspecting all the files which 'fed into' the final product, to track down any mistakes or other anomalies which have surfaced in the final document. It is also possible, with a system like this, to create 'dynamic documents'.

2.4. Dynamic documents

Dynamic documents are an extension of reproducible research in which the final report is highly automated. In its most developed form, dynamic documents make it possible for the code which produces the tables or graphics to be actually embedded within the document which produces the text of the report. In other words, there is no distinction between tables and commentary. This means that references to particular pieces of data in a table, for example, are automatically in the text which comments on that table. There is no need for the author of the document to type into their document any of the data results which are shown in the tables. The best known example of this approach is *Sweave* which runs inside the *R* statistics language. When it comes to using *Stata*, users also have a number of options for going down this path, and these were explained in depth in Bill Rising's presentation at the Oceania Stata User Group Meeting in 2016.

Sweave home page

Bill Rising "Dynamic documents in Stata: Many routes to the same goal". Download presentation.

Dynamic documents can be easily produced using **tabout**. This is particularly suitable for short reports or web pages which may need regular updating from data sets which change frequently. There is a chapter below dedicated to discussing how you can implement this approach using **tabout**. The underlying philosophy for dynamic documents is the idea of having 'files talking to files' and you can build a complete workflow around this concept.

For example, you might have a survey in the field, where the data arrives in stages, perhaps every few days. You could write a `master.do` file which cleans the data, recodes the variables and produces some preliminary tables, and this could be run against each new batch of data. As one gets closer to the end of the field work, you could make those tables come closer to their final form, including their final stylistic formatting. These tables could even be embedded in the report document and some commentary could be drafted. This kind of flexibility would allow researchers to move towards finality with the production of their reports, while still waiting for the stragglers to come in.

Another important consideration when the data changes might be the weighting of the data. There may be some final refinements made to the survey weights, but this need not entail problems, because by running `master.do` with the new weights, all of the tables in the report would update automatically.

The approach I've just sketched is difficult with repeated 'copy and paste' approach. Not only would this be a tedious task—particularly with dozens or scores of tables—but the formatting of the tables would always need to be left to the end, because this is often a labour-intensive task. Because **tabout** incorporates the formatting of the table in the initial production of the table, the strategy outlined above works seamlessly. In the case of LaTeX, the integration between content and formatting is complete. In the case of *Word*, dynamic documents are also possible and I discuss this later in the *User Guide*.

2.5. Single source publishing

Yet another variation on dynamic documents involves *single source publishing*, which refers to document production methods where multiple outputs might be required and only a single source of the original information is used. This has enormous importance for accuracy, because there is nothing worse than allowing multiple sources of information to proliferate (as you quickly discover if you allow your address book or contacts list to exist in multiple locations unsynchronised). The `xml` (*Extensible Markup Language*) is a standard developed to facilitate the production of multiple out-

puts (such as hardcopy newspapers and internet-based newspapers) from a single source. The efficiency made possible by this approach is also obvious, since there is no duplication of effort.

In the case of **tabout** an obvious example of this might be the need to produce tables for inclusion in a working document (suitable for word processing), as PDF files for exchanging by email or over the internet, and as `html` files for posting on web sites. By making use of **tabout**'s new `template file` option, a *Stata* user can now produce the same table in all of these different formats, with identical data in each table, but with the associated code and formatting suited to each destination. ⟹

Part II.

Table Content

3. Table content: overview

In essence, **tabout** allows a novice *Stata* user to produce multiple panels of cross-tabulations, and to lay out the data in a number of different ways. The output can be oneway or twoway tables of frequencies and/or percentages, as well as summary statistics (means medians etc). Standard errors and/or confidence intervals, based on *Stata's* **svy** commands, can also be included. Furthermore, a number of statistics (chi2, Gamma, Cramer's V, Kendall's tau) can be placed at the bottom of each panel. Finally, formatting of cell contents is simple, and allows users to choose the number of decimal places, and to insert percentage symbols and currency symbols.

Among the various enhancements in Version 3 of **tabout** are the ability to drop unwanted columns or rows, and to add new rows, once the initial table has been constructed. This avoids the need to recode variables, and deals with awkward situations where one variable may be missing data for certain categories (such as men who are pregant!). Version 3 also allows users to easily add a title and footnotes to a table. While this was possible in Version 2 of **tabout**, it is much simpler in Version 3.

3.1. tabout in practice: some short examples

To illustrate the basics of producing tables using **tabout**, several simple examples are shown below. However, it is first necessary to set up some data suitable for the kinds of tables we will produce in this *User Guide*. These data will be used throughout the *Guide*, so the following code is worth running at the outset (and you can copy it from the blue link). This code uses a built-in *Stata* dataset, and then recodes and labels the data to facilitate table output.

Copy this setup code

```
set seed 14921918
sysuse nlsw88, clear

la var union "Member of a union"
la def union 0 "Not a union member" ///
        1 "Union member", modify
la val union union

la var south "Location"
la def south 0 "Does not live in the South" ///
        1 "Lives in the South", modify
la val south south

la var race "Race"
la def race 1 "White" 2 "Black" ///
        3 "Other", modify
la val race race

la var collgrad "Education"
la def collgrad 0 "Not college graduate"  ///
        1 "College graduate", modify
la val collgrad collgrad

la var married "Marital status"
la def married 0 "Single" 1 "Married", modify
la val married married

gen wt = 10 * runiform()
gen int fwt = ceil(wt)

gen inc = 1000 * runiform()
gen income = cond(inc < 300, inc + 360, inc +200)
la var income "Income"

gen sex = cond(wt<5, 1, 2)
la var sex "Sex"
la def sex 1 "Male" 2 "Female", modify
replace sex = cond(wt<0.5, ., sex)
la val sex sex

gen pregnant = cond(wt>8.5, 1, 2)
la var pregnant "Currently pregnant"
la def pregnant 1 "Pregnant" 2 "Not pregnant" ///
        , modify
replace pregnant = cond(sex==1, ., pregnant)
la val pregnant pregnant

la var industry "Industry"
la var occupation "Occupation"
```

This basic *Stata* code is used to set up the main data needed for running all the examples in this guide. You can get a copy of the code by clicking on the link box at the top of the code. (You need an internet connection).

You will notice that the file for this setup code is invoked inside the various table do files used throughout this guide. This data has a large number of categorical variables and is particularly good for illustrating tables.

The modify options sprinkled throughout this code are there because this allows you to run this code repeatedly without *Stata* complaining.

The following table is produced using the code shown below. It illustrates the most basic **tabout** commands.

Table 1: A Simple Example

	Sex					
	Male		Female		Total	
	No.	%	No.	%	No.	%
Location						
Does not live in the South	584	58.2	651	57.8	1,235	58.0
Lives in the South	420	41.8	475	42.2	895	42.0
Total	1,004	100.0	1,126	100.0	2,130	100.0
Race						
White	737	73.4	810	71.9	1,547	72.6
Black	259	25.8	300	26.6	559	26.2
Other	8	0.8	16	1.4	24	1.1
Total	1,004	100.0	1,126	100.0	2,130	100.0
Member of a union						
Not a union member	635	74.3	706	76.1	1,341	75.2
Union member	220	25.7	222	23.9	442	24.8
Total	855	100.0	928	100.0	1,783	100.0

Source: nlsw88.dta

Copy table 1 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout south race union sex using table1.tex, replace ///
c(freq col) f(0c 1) style(tex) font(bold) twidth(11) ///
title(Table 1: A Simple Example) fn(Source: nlsw88.dta)
```

This code shows that **tabout** is similar to *Stata's* own **tabulate** command, as well as *Stata's* `table` command. There are four variables listed and the first three become what **tabout** calls 'vertical variables', in effect, the *panels* of the table. The fourth variable becomes the 'horizontal variable', the one which the other variables are cross-tabulated against.

The `using filename` part of the code is conventional *Stata* usage, and indicates where the output should go. Everything following the comma is an option, which in *Stata* is a way of fine-tuning any particular output that a user wants to produce. In the case of **tabout**, there are several options used here:

- `replace` overwrites the file if it already exists; ⟹
- `c(freq col)` is the `contents` option, which determines the cell contents: frequencies and column percentages; ⟹
- `f(0c 1)` is the `format` option and sets the cell numbers as zero ⟹

decimal points with commas for the frequencies and one decimal point for the column percentages;

- `style(tex)` indicates that the user wants LaTeX output (though one can also request 'pure' TeX output if desired. This is discussed later);
- `font(bold)` turns on bold formatting for the variable labels and the second row of the heading;
- `twidth(11)` sets the width of the table to 11 centimetres;
- `title(Table 1 etc)` is placed above the table; and
- `fn(Source etc)` is a footnote placed below the table.

Existing **tabout** users should be aware of the change in the `c` option. *Stata*'s **table** command also has a contents option (abbreviated to `c`) and earlier versions of **tabout** used the `c` abbreviation for *cells*, rather than *contents*. Now, in Version 3 **tabout** uses the *Stata* terminology. This makes sense, because one of the arguments for this option is `cell` (that is, cell percentage). Adopting the *Stata* convention of `contents( )` avoids confusion, although most users simply ignore the issue by using the much shorter abbreviation of `c( )`.

There are a large number of options in **tabout**, all of which can be quickly referenced in the Options table at the end of the *User Guide*. There are basically two forms these options take:

- a *switch*, where a single term turns on or off some feature. For example, the default setting is for a table to have borders above and below, and rules (horizontal lines) separating panels and heading rows. There is a `noborder` option which turns off the border lines, a `nohlines` option which turns off the lines in the heading rows, and a `noplines` option which turns off the panel lines. You can use all of these together, or in any combination, to reverse any of the defaults.
- an *argument*, where a term is followed by an argument in parentheses. Most of the options in the example above use arguments which 'pass' a particular setting to **tabout** to implement. In the absence of that option, **tabout** uses a default, which is usually a common setting. Unlike some areas in*Stata*, **tabout** avoids the use of sub-options and sub-sub-options. The standard pattern in **tabout** is for an option to take a single argument, and if more information is needed, a second option is used. As we shall see below, this produces 'families' of options. For example, reporting the sample size is set by an `npos` option (meaning *N position*) which can be set to `col`, `row`, `both`, `lab` or `tufte`; there is also an `nlab` option (meaning *N label*) which can take a phrase, such as 'Sample size' or 'Obs' to replace the default 'N'; and there is a `nwt` to assign a different weight to the N counts (which is suitable for producing population estimates).

Here is a second table which illustrates these principles, with the **tabout** code shown below.

Table 2: Including the sample size

	Sex						
	Male		*Female*		*Total*		*Obs*
	%	%	%	%	%	%	
Location							
Does not live in the South	58.2	47.3	57.8	52.7	58.0	100.0	1,235
Lives in the South	41.8	46.9	42.2	53.1	42.0	100.0	895
Total	100.0	47.1	100.0	52.9	100.0	100.0	2,130
Obs	1,004		1,126		2,130		
Race							
White	73.4	47.6	71.9	52.4	72.6	100.0	1,547
Black	25.8	46.3	26.6	53.7	26.2	100.0	559
Other	0.8	33.3	1.4	66.7	1.1	100.0	24
Total	100.0	47.1	100.0	52.9	100.0	100.0	2,130
Obs	1,004		1,126		2,130		
Member of a union							
Not a union member	74.3	47.4	76.1	52.6	75.2	100.0	1,341
Union member	25.7	49.8	23.9	50.2	24.8	100.0	442
Total	100.0	48.0	100.0	52.0	100.0	100.0	1,783
Obs	855		928		1,783		

Source: nlsw88.dta

Copy table 2 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout south race union sex using table2.tex, replace ///
c(col row) f(1) style(tex) font(italic) twidth(13) ///
title(Table 2: Including the sample size) ///
fn(Source: nlsw88.dta) npos(both) nlab(Obs)
```

New to this example are the `npos` and `nlab` options (just discussed) as well as a different `font` option (`italic`) and the inclusion of both row and column percentages. Note that the `f()` option only needs one argument here (1) because **tabout** will repeat this for as many columns as necessary. You can, of course, provide as many arguments as the types of categories in your table. For example, `f(1 2 0c)` would make the first column one decimal point, the second two decimal points, and the third no decimal points with commas. Any additional columns in the table would repeat this last setting (that is, no decimal points with commas).

As well as `0c` (to show the *comma*), you can also use `p` (to show the *percentage* sign), `m` (for *money* to show a currency symbol. To actually set which currency symbol is used, you can use the `money` option). Finally,

you can also change the decimal point and thousand separators to the style favoured in some European countries (namely ',' for decimal points and '.' for thousands) using the `dpcomma` option. ⟹

Table 2 illustrates a problem with combining row and column percentages in this way. While the column percentages are easy to read, comparing them requires the eye to jump across an intervening column. The same is true for the row percentages, where the obvious summing to 100 percent is broken up by the intervening columns.

Table 3 shows how one might improve the readability of this data by using **tabout**'s `layout` option. When this is set to `cb` (*column block*), this readability problem is resolved. A further refinement is to suppress the row of % symbols on the third heading row and add the '(%)' term to the title. **tabout** works with three heading rows, options `h1`, `h2` and `h3`. Thus by specifying `h3(nil)` you can suppress this row. We shall see later that you can also change the wording on this row. ⟹ ⟹ ⟹ ⟹

Table 3: Further refinements, using column block layout (%)

	Sex						
	Male	*Female*	*Total*	*Male*	*Female*	*Total*	*Obs*
Location							
Does not live in the South	58.2	57.8	58.0	47.3	52.7	100.0	1,235
Lives in the South	41.8	42.2	42.0	46.9	53.1	100.0	895
Total	100.0	100.0	100.0	47.1	52.9	100.0	2,130
Obs	1,004	1,126	2,130				
Race							
White	73.4	71.9	72.6	47.6	52.4	100.0	1,547
Black	25.8	26.6	26.2	46.3	53.7	100.0	559
Other	0.8	1.4	1.1	33.3	66.7	100.0	24
Total	100.0	100.0	100.0	47.1	52.9	100.0	2,130
Obs	1,004	1,126	2,130				
Member of a union							
Not a union member	74.3	76.1	75.2	47.4	52.6	100.0	1,341
Union member	25.7	23.9	24.8	49.8	50.2	100.0	442
Total	100.0	100.0	100.0	48.0	52.0	100.0	1,783
Obs	855	928	1,783				

Source: nlsw88.dta

Copy table 3 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout south race union sex using table3.tex, replace ///
style(tex) font(italic) twidth(13) c(col row) h3(nil) ///
f(1) npos(both) nlab(Obs) layout(cb) title(Table 3: ///
Further refinements, using column block layout (%)) ///
fn(Source: nlsw88.dta)
```

Understanding these different layouts as well as the kind of content which is permissible in **tabout** takes some time, but a useful summary can be seen in table below. Some of the options shown here will be discussed in coming chapters, so don't worry if it's a bit cryptic at this stage. Also, if you do make an error when typing the syntax of **tabout**, this table is displayed on your screen, alongside a hint as to the nature of your error. This table is particularly useful for summary tables, as it lists as the terms that you can include in your `contents` option. ⟹

Type of table	Allowable cell contents	Available layout
Basic	freq cell row col cum **any number of above, in any order** *for example: cells(freq col)*	col row cb rb
Basic with SE or CI (turn on *svy* option)	freq cell row col se ci lb ub **only one of:** freq cell row col *(must come first in the cell)* **and any number of:** se ci lb ub *for example: cells(col se lb ub)*	col row cb rb
Summary -as a oneway table (turn on *sum* option; also may need to turn on *oneway* option)	**any number of:** N mean var sd skewness kurtosis sum uwsum min max count median iqr r9010 r9050 r7525 r1050 p1 p5 p10 p25 p50 p75 p90 p95 p99 **with each followed by variable name** *for example: cells(min wage mean age)*	no options (fixed)
Summary -as a twoway table (turn on *sum* option)	**only one of:** N mean var sd skewness kurtosis sum uwsum min max count median iqr r9010 r9050 r7525 r1050 p1 p5 p10 p25 p50 p75 p90 p95 p99 **followed by one variable name** *for example: cells(sum income)*	no options (fixed)
Summary with SE or CI (turn on *sum* option and *svy* option)	mean **followed by one variable name** **and any number of:** se ci lb ub *for example: cells(mean weight se ci)*	col row cb rb

Note: cb = column block; rb = row block, SE = standard errors; CI = confidence intervals.

Some aesthetic considerations

Some of the more subtle ways in which aesthetics apply to tables concern the thickness of the lines and horizontal alignment. Ideally, table borders—that is, the top and bottom horizontal lines of the table—should be slightly thicker than the heading and panel lines. This is automatically implemen-

ted with `tex`, `xlsx` and `htm` output. At this stage `docx` output does not support this feature.

Horizontal alignment follows a few simple principles:

- labels in the first column are always left aligned;
- data cells are always right aligned;
- heading cells which span columns are always centred;

For other heading cells which are inside a single column, there is some variation between output styles. The `tex` and `htm` output styles use right alignment for these cells by default, but users can code this to suit themselves using the `topf` option. For the `docx` output style—where there is no facility for using `topf`—the default is centred. However, these cells can be made right aligned with the `hright` (*heading right*) option. ⟹ ⟹

There is some similar variation with the alignment of the table on the page. For `tex` and `docx` output styles the default is to centre align the table on the page, but the `tleft` (*table left*) option will place it flush with the margin. For both `xlsx` and `htm` output, there is no real concept of a page, so the default is left. If a `htm` user wants to centre their table on a particular display, they can code this using `div` tags. If an `xlsx` user wants to centre their table, then the cell location of the top left corner of the table can be specified, and this will allow users to shift the table elsewhere on the spreadsheet 'page'. ⟹

In summary, most of the default settings for these aesthetic considerations work well, but users can always fine tune these by either coding their preference (if using `tex` or `htm`) or by selecting various options which are available for particular styles.

3.2. Roadmap for Part II

This concludes a short overview of **tabout** in practice. The next three chapters—chapters 4, 5 and 6—provide a more systematic exposition of the types of tables which are possible using **tabout**. Each chapter deals with a particular type of table:

- chapter 4: basic tables—cross-tabulations of counts and percentages;
- chapter 5: summary tables—cross-tabulations of summary measures such as means and medians etc;
- chapter 6: survey tables—cross-tabulations of counts, percentages and summary measures, with standard errors and confidence intervals;

You might like to read these chapters systematically, or you might just want

to look through all the examples seeking a table which comes closest to the table you wish to produce, and then to modify the code to suit your specific needs. To make life easy, each block of code can be copied in the output style which you want by clicking on the link at the top of each block. (If you are using LaTeX, you might choose to just copy and paste the text inside the code block.) As mentioned earlier, all of the examples are also consolidated into a single file (for each output style) and you must make sure you rename their `.txt` extension to `.do` before you can run them in *Stata*.

The approach taken in each chapter is to begin with a simple table, and then to show how more refinements can be introduced into a table to deal with various requirements. Such refinements illustrate starkly that there is always a trade-off between *complexity* and *flexibility*, and while **tabout** tries to make this as simple as possible, the contradiction cannot be avoided. If users were happy to accept just some default settings, then the **tabout** syntax could remain very simple. It is giving users *choice* which introduces the complexity, but only with choice can many of the challenges of producing publication quality tables be surmounted. Some users will, no doubt, wish for even greater choice, but that would require yet more options, and more complexity.

Chapters 7, 8 and 9 introduce some more advanced features of **tabout**. These include:

- chapter 7: how to 'reshape' your tables, to drop columns or rows; or to plug gaps in your tables where idiosyncratic variables arise;
- chapter 8: using template files, a method of simplifying your **tabout** code and sharing **tabout** code between colleagues and with beginners;
- chapter 9: how you can build dynamic documents with **tabout**, or, at a minimum, how to implement some of the principles of reproducible research.

4. Basic tables

This chapter introduces the main features of basic tables, which is **tabout**'s term for oneway and twoway cross tabulations of categorical data. The examples make use of the `nlsw.dta` data which was used in the Overview chapter and supplements this with data from another built-in *Stata* dataset, namely cancer.dta. Again, some initial setting up is needed, mainly to create a categorical variable from a continuous one.

Copy this setup code

```
sysuse cancer, clear
la var died "Patient died"
la def ny  0 "No" 1 "Yes", modify
la val died ny
recode studytime ///
    (min/10 = 1 "10 or less months") ///
    (11/20 = 2 "11 to 20 months") ///
    (21/30 = 3 "21 to 30 months") ///
    (31/max = 4 "31 or more months") ///
    , gen(stime)
la var stime "To died or exp. end"

la var drug "Drug type"
la def drug 1 "Placebo" 2 "Trial drug 1" ///
    3 "Trial drug 2", modify
la val drug drug
```

This basic *Stata* code is used to set up some additional data which is also used for running examples in this guide. You can get a copy of the code by clicking on the link box at the top of the code. (You need an internet connection.

You will notice that the file for this setup code is invoked inside a number of table do files used in the next few chapters. (This data has a good mix of categorical and continuous variables.)

4.1. Twoway tables

The following table repeats some of the features of **tabout** discussed in the Overview chapter, and introduces some new ones. One thing to notice is that the contents option contains a cumulative percentage, a feature not readily available with *Stata*'s own **tabulate** command.

Unlike *Stata*'s **tabulate** command, **tabout** allows you to specify the order in which your results should appear, instead of the fixed sequence (frequency, row percentage and column percentage) which **tabulate** always uses.

Table 4: Example of a simple cross tabulation

	Patient died								
	No			*Yes*			*Total*		
	No.	Col %	Cum %	No.	Col %	Cum %	No.	Col %	Cum %
To died or exp. end									
10 or less months	4	23.5	23.5	15	48.4	48.4	19	39.6	39.6
11 to 20 months	6	35.3	58.8	8	25.8	74.2	14	29.2	68.8
21 to 30 months	2	11.8	70.6	7	22.6	96.8	9	18.8	87.5
31 or more months	5	29.4	100.0	1	3.2	100.0	6	12.5	100.0
Total	17	100.0		31	100.0		48	100.0	

Source: cancer.dta

Copy table 4 code as: txt tex docx xlsx htm

```
do cancer_data_setup
tabout stime died using table4.tex, replace ///
c(freq col cum) style(tex) font(italic) twidth(14) ///
f(0 1) clab(No. Col_% Cum_%) title(Table 4: Example ///
of a simple cross tabulation) fn(Source: cancer.dta)
```

This table shows how you can distinguish between column and cumulative percentages, which is done by overriding default % symbol in the third heading row. We saw earlier that you could suppress the third heading row by specifying `h3(nil)`, and another way to use that option is to specify the labelling of this row by placing your own terms inside the `h3` option. However, in this case it's unnecessary as there's a simpler method: the `clab`(*column label*) option. ⟹

In most circumstances users really only need to change the third heading row, and that's what the `clab` option used in Table 4 is intended for. It's a simple solution and lets you simply substitute an alternative set of column headings for the default ones. The only complicated bit is that you need to use underscores if any of your column headings have spaces in them. In this example, they do (eg. Col %), so an underscore is needed in your **tabout** syntax. These underscores are, of course, stripped out by **tabout** during the production of the table, so that normal spaces appear in the final output.

Using underscores for spaces is not common in *Stata* but is a well-known convention for file naming. It avoids the problems that spaces often cause. It is used for numerous options in **tabout** when spaces occur inside labels.

Panels

Table 5 uses panels, one of the key concepts in **tabout** and intrinsic to its design philosophy of visual comparisons.

Table 5: Example of cross tabulation using panels

	Education		
	Not college graduate	College graduate	Total
Location			
Does not live in the South	4,941	1,516	6,458
Lives in the South	3,661	1,095	4,756
Total	8,602	2,611	11,213
	%	%	%
Does not live in the South	76.5	23.5	100.0
Lives in the South	77.0	23.0	100.0
Total	76.7	23.3	100.0
	%	%	%
Does not live in the South	57.4	58.1	57.6
Lives in the South	42.6	41.9	42.4
Total	100.0	100.0	100.0
Race			
White	6,047	2,086	8,133
Black	2,452	482	2,934
Other	103	43	146
Total	8,602	2,611	11,213
	%	%	%
White	74.4	25.6	100.0
Black	83.6	16.4	100.0
Other	70.3	29.7	100.0
Total	76.7	23.3	100.0
	%	%	%
White	70.3	79.9	72.5
Black	28.5	18.5	26.2
Other	1.2	1.7	1.3
Total	100.0	100.0	100.0

Source: nlsw88.dta

Copy table 5 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout south race collgrad [iw=wt] using table5.tex, ///
replace style(tex) font(bold) c(freq row col) ///
f(0c 1 1) layout(rb) h3(nil) twidth(11) ///
title(Table 5: Example of cross tabulation ///
using panels) fn(Source: nlsw88.dta)
```

With a table like this, one can instantly compare educational background against two other variables, glancing both across the page and down the page, taking in multiple comparisons in a compact spacial arrangement. Table 5 also uses another of the layout options, in this case, the *row block* choice: `layout(rb)`. This places the cell contents (frequencies, ⟹ row and column percentages) in a row formation, rather than in columns.

In a table like Table 5, it doesn't make sense to have headings in the third row, so the `h3(nil)` option is used to remove this. However, because ⟹ of this unique layout, this option only affects the top part of the table, that is the heading rows. The % symbols re-appear at the top of all the intervening rows blocks. This may or may not be what suits the user. If you would prefer to remove these, and indicate that these numbers are percentages using the format option, then Table 6 on the next page is your solution.

The secret to the result shown in Table 6 is to use the `clab` option ⟹ alongside `h3(nil)` because this will influence every occurrence of a column heading, even when it's not at the top of a table. To make it an 'empty' column heading, as required here, simply place blank spaces in the labels, using an underscore for each column in the table (here there are three). Hence the syntax: `clab(_ _ _)`. You then need to add the *p* symbol in your format option to show percentage symbols alongside each number: ⟹ `f(0 1p 1p)`.

Table 6: Same example with refinements

	Education		
	Not college graduate	College graduate	Total
Location			
Does not live in the South	4,941	1,516	6,458
Lives in the South	3,661	1,095	4,756
Total	8,602	2,611	11,213
Does not live in the South	76.5%	23.5%	100.0%
Lives in the South	77.0%	23.0%	100.0%
Total	76.7%	23.3%	100.0%
Does not live in the South	57.4%	58.1%	57.6%
Lives in the South	42.6%	41.9%	42.4%
Total	100.0%	100.0%	100.0%
Race			
White	6,047	2,086	8,133
Black	2,452	482	2,934
Other	103	43	146
Total	8,602	2,611	11,213
White	74.4%	25.6%	100.0%
Black	83.6%	16.4%	100.0%
Other	70.3%	29.7%	100.0%
Total	76.7%	23.3%	100.0%
White	70.3%	79.9%	72.5%
Black	28.5%	18.5%	26.2%
Other	1.2%	1.7%	1.3%
Total	100.0%	100.0%	100.0%

Source: nlsw88.dta

Copy table 6 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout south race collgrad [iw=wt] using table6.tex, ///
replace style(tex) font(bold) c(freq row col) ///
f(0c 1p 1p) layout(rb) clab(_ _ _) h3(nil) twidth(11) ///
title(Table 6: Same example with refinements) ///
fn(Source: nlsw88.dta)
```

4.2. Twoway tables with statistics

One of the most useful features of **tabout** is the ability to incorporate sample counts and statistics inside the body of a table of descriptives, such as counts or percentages. There is also considerable flexibility in how this can be done. Version 2 of **tabout** allowed sample counts to be shown in columns

or rows and to be given custom labels, but prior to Version 3 of **tabout**, the statistics could only be shown in rows and there was no customisation of labels. Version 3 improves on this: users can place statistics in columns, can customise the labels and formats, and can also arrange how the statistic, and its p-value, are laid out. There is also more customisation of p-values: they can be shown as stars or as ranges.

Table 7 shows the default values for these options, applied to a Pearson's chi-squared statistic. Table 8 shows a number of customisation features:

- custom labels for sample size and for statistics;
- column position of statistics; and
- display of p-values as stars.

Table 7: Example of sample counts and stats

	Race			
	White	**Black**	**Other**	**Total**
	Col %	Col %	Col %	Col %
Location				
Does not live in the South	65.4	36.0	88.5	58.1
Lives in the South	34.6	64.0	11.5	41.9
Total	100.0	100.0	100.0	100.0
N	1,637	583	26	2,246
Pearson chi2(2) = 162.625 P-value = 0.000				
Education				
Not college graduate	74.3	82.3	65.4	76.3
College graduate	25.7	17.7	34.6	23.7
Total	100.0	100.0	100.0	100.0
N	1,637	583	26	2,246
Pearson chi2(2) = 16.919 P-value = 0.000				

nlsw88.dta

Copy table 7 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout south collgrad race using table7.tex, replace ///
style(tex) font(bold) c(col) f(1) npos(row) ///
twidth(12) clab(Col_%) stats(chi2) title(Table 7: ///
Example of sample counts and stats) fn(nlsw88.dta)
```

The novel aspect to Table 7, the statistics, makes use of *Stata*'s **tabulate** command. Thus **tabout**'s `stats` option uses the same measures: Pearson's chi-squared (*chi2*), Goodman and Kruskal's gamma (*gamma*), Cramér's V ⟹

(*V*), Kendall's tau-b (*taub*) and the likelihood-ratio chi-squared (*lrchi2*). Table 7 shows the default settings and labels for the *chi2* statistic.

Customisation of some of these settings is shown in Table 8. In particular:

- stpos(*col*) (*statistics position*) which is in the column, rather than the default row position; ⟹
- stlab(*Chi2*) (*statistics label*) which uses a shorter phrase to fit the column width; ⟹
- stform(*2*) (*statistics format*) is set to 2 decimal points (the default is 3). There is also an equivalent pform option for formatting the p-value. ⟹ ⟹
- plab(*Signif*) (*p-value label*) which also uses a shorter phrase to fit the column width; ⟹
- stars which indicates that statistical significance should be shown according to the stars convention, rather than an explicit p-value be shown. ⟹

It is also possible to change the position of the p-value using ppos, so that it has its own column. In addition, various combinations of including or omitting the statistic and the p-value are possible. Interested users might like to explore these combinations for themselves, though this guide will also demonstrate some variations later. ⟹

Table 8: Same example with customisation

	Race				
	White	**Black**	**Other**	**Total**	**Chi2**
	Col %	Col %	Col %	Col %	Signif
Location					
Does not live in the South	65.4	36.0	88.5	58.1	162.62
Lives in the South	34.6	64.0	11.5	41.9	***
Total	100.0	100.0	100.0	100.0	
Sample size	1,637	583	26	2,246	
Education					
Not college graduate	74.3	82.3	65.4	76.3	16.92
College graduate	25.7	17.7	34.6	23.7	***
Total	100.0	100.0	100.0	100.0	
Sample size	1,637	583	26	2,246	

nlsw88.dta

Copy table 8 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout south collgrad race using table8.tex, replace ///
style(tex) font(bold) c(col) f(1) clab(Col_%) ///
twidth(14) stats(chi2) stpos(col) stlab(Chi2) ///
```

```
stform(2) plab(Signif) stars npos(row) ///
nlab(Sample size) title(Table 8: Same example with ///
customisation) fn(nlsw88.dta)
```

4.3. Oneway tables

The assumption made by **tabout** is that the last variable in the variable list is always the 'horizontal' variable, that is, the variable against which all the other variables are cross-tabulated. However, the user may wish to produce one-way tables, and for this purpose, the `oneway` option is available. It basically says: "the last variable in the list is also a 'vertical' variable so don't cross-tabulate".

Table 9, below, provides an example of a typical oneway table. In this example there is only one heading row, because that is all that is needed. It has been customised here with the `clab` option, so that the reader can distinguish at a glance between column percentages and cumulative percentages. Additional heading rows could be added, if the user required this, with the `h1` and `h2` options.

Table 9: A oneway table

	Count	Col %	Cum %	Sample
Industry				
Ag/Forestry/Fisheries	84	0.8	0.8	17
Mining	14	0.1	0.9	4
Construction	160	1.4	2.3	29
Manufacturing	1,848	16.6	18.9	367
Transport/Comm/Utility	433	3.9	22.8	90
Wholesale/Retail Trade	1,685	15.1	37.9	333
Finance/Ins/Real Estate	970	8.7	46.7	192
Business/Repair Svc	429	3.9	50.5	86
Personal Services	472	4.2	54.8	97
Entertainment/Rec Svc	99	0.9	55.6	17
Professional Services	4,151	37.3	92.9	824
Public Administration	786	7.1	100.0	176
Total	11,129	100.0		2,232
Occupation				
Professional/technical	1,477	13.2	13.2	317
Managers/admin	1,322	11.8	25.1	264
Sales	3,626	32.5	57.5	726
Clerical/unskilled	511	4.6	62.1	102
Craftsmen	239	2.1	64.2	53
Operatives	1,305	11.7	75.9	246
Transport	136	1.2	77.1	28
Laborers	1,491	13.4	90.5	286
Farmers	8	0.1	90.5	1
Farm laborers	40	0.4	90.9	9
Service	75	0.7	91.6	16
Household workers	3	0.0	91.6	2
Other	938	8.4	100.0	187
Total	11,171	100.0		2,237

Source: nlsw88.dta

Copy table 9 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout industry occupation [iw=wt] using table9.tex, ///
replace style(tex) font(bold) oneway c(freq col cum) ///
f(0c 1) clab(Count Col_% Cum_%) twidth(11) npos(col) ///
nlab(Sample) title(Table 9: A oneway table) ///
fn(Source: nlsw88.dta)
```

Note also that the clab option only labels the *data columns*. The sample size column is labelled by using the nlab option. This is the same convention for statistics and p-values, each of which have their own labelling options. The **tabout** user needs to visualise their tables as composed of *panels* going down the page, and *blocks of data* plus information going across the page. The complete table has, at a minimum, the data produced by the contents option, but it can also contain (as here) sample counts, and (as seen earlier), various statistics. So keeping the distinction in your mind, between *data columns* and these *extra columns* is a useful way to avoid confusion, and also a way to customisation all these headings the way you wish. There may be occassions, for example, when you prefer this additional information (sample size and statistics) to be at the bottom of the table, using the positional options (npos, stpos and ppos), in which case, the only columns in your table will be *data columns*.


5. Summary tables

Summary tables are any measure which *Stata*'s **summmarize** command can produce, re-arranged into a table layout. If you require a cross-tabulation—that is, 'vertical' variables against one 'horizontal' variable—then only one summary measure can be used, which is placed in the table cell defined by the intersection of these variables. The range of measures, and what is allowable, was shown earlier as a table. If you are satisfied with a oneway table—using just 'vertical variables'—then any number of summary measures can be used (space permitting), and these are all arranged horizontally across the page. The examples below will make this clearer.

5.1. Twoway summary tables

The key term needed for all summary tables in **tabout** is the `sum` option, ⟹ which tells **tabout** that this is not a table composed of counts or percentages, but should use summary measures to fill the cells. What those measures should actually be is determined by the `contents` option, where the syntax ⟹ differs from that used in basic tables because the names of other variables are placed inside this option. Table 10 below illustrates this principle: it shows the use of the means measure, and those users familiar with *Stata*'s **table** command will recognise this form of syntax. Inside the `contents` option you simply use the summary measure (*`mean`*) and follow it with the name of the variable (*`weight`*).

Table 10: Simple twoway summary table of means

	Car type (mean weight in lbs.)		
	Domestic	*Foreign*	*Total*
Repair Record 1978			
1	3,100		3,100
2	3,354		3,354
3	3,442	2,010	3,299
4	3,532	2,208	2,870
5	1,960	2,403	2,323
Total	3,368	2,263	3,032

auto.dta

Copy table 10 code as: txt tex docx xlsx htm

```
sysuse auto, clear
tabout rep78 foreign using table10.tex, replace ///
style(tex) font(italic) c(mean weight) f(0c) sum ///
twidth(9) h1(Car type (mean weight in lbs.)) h3(nil) ///
title(Table 10: Simple twoway summary table of means) ///
fn(auto.dta)
```

Some refinements shown in this table include `h3(nil)`. This removes the repetition of 'Mean weight', which **tabout** would otherwise apply. The `h1(Car type (mean weight in lbs.))` changes the default heading to make it clear what the cells contain.

Table 11 shows another twoway summary table, in this case using median measures.

Table 11: Simple twoway summary table of medians

	Candidate voted for, 1992			
	Clinton	*Bush*	*Perot*	*Total*
	%	%	%	%
Family Income				
<$15k	8.3	3.2	2.5	3.2
$15-30k	10.8	8.4	4.8	8.4
$30-50k	12.3	11.4	6.3	11.4
$50-75k	8.0	8.4	3.6	8.0
$75k+	4.7	6.2	2.1	4.7
Total	8.3	8.4	3.6	6.3

voter.dta

Copy table 11 code as: txt tex docx xlsx htm

```
sysuse voter, clear
tabout inc candidat using table11.tex, replace ///
style(tex) font(italic) c(median pfrac) f(1) clab(%) ///
sum  twidth(9) title(Table 11: Simple twoway summary ///
table of medians) fn(voter.dta)
```

In Table 11 the cells of this table show percentages but that is simply because the variable *pfrac* itself is a percentage measure. Interpreting this is a bit tricky. This example is used to simply illustrate the **tabout** syntax. Basically, the table presents the breakdown of Presidential votes in 1992 according to the voter's family income, showing the median percentage vote for each candidate within each income bracket.

5.2. Oneway summary tables

Turning now to oneway tables, the use of multiple measures is illustrated in Table 12. Here there are two 'vertical' variables and no 'horizontal' variable, so the columns across the page are populated with the various summary measures, means for three variables, medians for two.

Table 12: Oneway summary table with multiple summary measures

	Mean			Median	
	MPG	Weight (lbs)	Length (in)	Price	Headroom (in)
Car type					
Domestic (70%)	19.8	3,317.1	196.1	$4,782.50	3.5
Foreign (29%)	24.8	2,315.9	168.5	$5,759.00	2.5
Total (100%)	21.3	3,019.5	187.9	$5,006.50	3.0
Repair Record 1978					
1 (2%)	21.0	3,100.0	189.0	$4,564.50	1.8
2 (11%)	19.1	3,353.8	199.4	$4,638.00	3.8
3 (43%)	19.4	3,299.0	194.0	$4,741.00	3.5
4 (26%)	21.7	2,870.0	184.8	$5,751.50	3.0
5 (15%)	27.4	2,322.7	170.2	$5,397.00	2.5
Total (100%)	21.3	3,032.0	188.3	$5,079.00	3.0

Source: auto.dta

Copy table 12 code as: txt tex docx xlsx htm

```
sysuse auto, clear
tabout foreign rep78 using table12.tex, replace ///
style(tex) font(bold) twidth(13) sum npos(tufte) ///
c(mean mpg mean weight mean length median price ///
median headroom) f(1c 1c 1c 2cm 1c) h2(Mean Median) ///
h2c(3 2) clab(MPG Weight_(lbs) Length_(in) Price ///
Headroom_(in)) title(Table 12: Oneway summary table ///
\\ with multiple summary measures) fn(Source: auto.dta)
```

What is notable in Table 12 is:

- the syntax for the `contents` option: where each summary measure is followed by a variable name: `c(mean mpg mean weight mean length ... )`; ⟹
- the `oneway` option is *not* needed here because **tabout** is restricted to only producing oneway tables when you request multiple statistics across the columns; ⟹

- the `npos(tufte)` option, which places the distribution of the categories within the two 'vertical' variables within their labels; ⟹
- the use of a LaTeX new line symbol (\\) in the title improves its appearance (both the *tex* and *htm* output styles allow codes in your title and footnote options);
- the use of the `h2` option in conjunction with the `h2c` option.

The tufte option is named after the approach adopted by Edward Tufte in his construction of a 'supertable', which he designed for the *New York Times* in 1980. See **Tufte2001**.

This last item is worth further discussion. Headings are highly customisable in **tabout** and this is done with the `h1`, `h2` and `h3` options. As we have seen earlier, placing *nil* in any of these options suppresses that row from the final table. Alternatively, if you would like to fully customise your table headings you can use these options and insert the words you wish to display. When it comes to placing these words into the correct columns, or spanning the right number of columns, you can write your own code (if you are using the *tex* or *htm* output styles), or you can let **tabout** do this for you (which will work for all output styles). ⟹ ⟹ ⟹ ⟹

How does **tabout** know which columns to span or which words go into which columns? For column placement **tabout** uses the same convention as for the `clab` option: spaces between words separate columns, and underscores are needed if phrases (containing spaces) are to appear in a column. Spanning is a bit more complicated, but makes use of the three 'partner' options: `h1c`, `h2c` and `h3c`, all of which stand for *heading columns*. By inserting numbers in these options, you tell **tabout** which columns to span with the corresponding words found in their 'partner' `h1`, `h2` or `h3` option. ⟹ ⟹ ⟹ ⟹

So what tells **tabout** that `h2`, for example, should be treated in this way, and not regarded as the user's own code? The presence of the 'partner' option `h2c`. If **tabout** finds that present, it parses the `h2` option and does all the work for you. If it finds it missing, then **tabout** assumes that your `h2` words are all your own coding and just inserts the words as it finds them.

In the case of Table 12, the *Mean* label was intended to span 3 columns, the *Median* label was intended to span 2 columns. Hence, the option was specified as: `h2c(3 2)`. While all of this might sound complex, in fact it simplifies matters considerably. If you want complete customisation of your headings, and you don't mind doing all the coding, you can use `h2` the way it's always functioned. On the other hand, if you find that you're regularly needing to use `h2` with your summary tables, then this simple shortcut might suit you perfectly.

This issue also illustrates how aesthetics and information decoding belong together. While `h2c` works in all output styles in the manner just described, in spanning multiple columns and centering the labels over these columns, in the case of *tex* output, it has an additional role. This option determines where the breaks in the heading lines occur. Not only do these gaps in the line look 'neater' but they also signal which columns (the first

three) belong to the 'Mean' label and which columns (the last two) belong to the 'Median' label. While sometimes this is obvious, sometimes it is not, and the use of a gap in the line confirms what belongs with what.

Quite often you may need to customise your headings when producing summary tables. **tabout** produces default headings, but often these are not suited to the particular needs of summary tables. Table 13 illustrates this problem and shows what happens when you produce a oneway table with the default column headings. **tabout** does not know how you would like the third heading row to appear, so it just repeats the variable name. **tabout** could just use the summary measure name in combination with the variable, but these are often cryptic or likely to be too cumbersome to fit neatly into a column.

Table 13: Oneway summary table with default headings

	age	age	studytime	studytime	studytime
Patient died					
No	54.2	8.0	19.0	6.0	39.0
Yes	56.8	9.0	11.0	1.0	33.0
Total	55.9	9.5	12.5	1.0	39.0
Drug type					
Placebo	56.0	8.5	8.0	1.0	23.0
Trial drug 1	56.9	12.0	14.0	6.0	32.0
Trial drug 2	54.6	8.0	26.5	6.0	39.0
Total	55.9	9.5	12.5	1.0	39.0

Source: cancer.dta

Copy table 13 code as: txt tex docx xlsx htm

```
do cancer_data_setup
tabout died drug  using table13.tex, replace ///
style(tex) font(italic) twidth(12) sum f(1) ///
c(mean age iqr age median studytime min studytime ///
max studytime) title(Table 13: Oneway summary table ///
with default headings) fn(Source: cancer.dta)
```

Table 14 shows how to solve this problem and produce true publication quality table straight from **tabout**. The first step in the solution involves using `clab` to assign headings to the columns which reflect which measures are being used. The second step uses the `h2` option to put meaningful headings above these measures: not just the variable name, but also the quantity metric (years and months).

Table 14: Same oneway summary table with customised headings

	Patient age (yrs)		*Length of study (mths)*		
	Mean	IQR	Median	Minimum	Maximum
Patient died					
No	54.2	8.0	19.0	6.0	39.0
Yes	56.8	9.0	11.0	1.0	33.0
Total	55.9	9.5	12.5	1.0	39.0
Drug type					
Placebo	56.0	8.5	8.0	1.0	23.0
Trial drug 1	56.9	12.0	14.0	6.0	32.0
Trial drug 2	54.6	8.0	26.5	6.0	39.0
Total	55.9	9.5	12.5	1.0	39.0

Source: cancer.dta. Note that IQR is interquartile range

Copy table 14 code as: txt tex docx xlsx htm

```
do cancer_data_setup
tabout died drug  using table14.tex, replace ///
style(tex) font(italic) twidth(12) sum f(1) ///
c(mean age iqr age median studytime min studytime ///
max studytime) h2c(2 3) ltrim(1) h2(Patient_age_(yrs) ///
Length_of_study_(mths)) clab(Mean IQR Median Minimum ///
Maximum) title(Table 14: Same oneway summary table ///
\\ with customised headings) fn(\emph{Source}: ///
cancer.dta. Note that IQR is interquartile range)
```

Table 14 also illustrates the use of underscores in `h2` for terms that have spaces within them—this is the same convention used by `clab`. It also shows the use of `ltrim` which determines how large the gap between the partial lines should be. This is, however, an option only available with *`tex`* output).

⟹

Advanced **tabout** users who have used the `cl1` and `cl2` options in the past, will find these missing in Version 3. This is because they are implemented automatically. The former `cltr1` `cltr2` options have been replaced by the simpler *`ltrim`* option.

As mentioned earlier all the **tabout** syntax shown in the code boxes in this guide has been for producing *`tex`* output (of course, when you click on the copy link, you get the syntax in the style you choose). The reason for this is that this user guide is written in LaTeX. It also provides me with the opportunity to highlight, from time to time, some of the advantages of LaTeX. One of these is the way the user can introduce code into labels with great ease. Thus Table 14 also shows the LaTeX newline symbol (\\) being used in the title to control where the word wrap takes place and the LaTeX italic code being used in the footnote to selectively italicise just one word.

6. Survey tables

One of the great advantages of *Stata* is its survey capabilities. Through the use of the **svy** prefix command, *Stata* users can easily analyse complex survey data. As is well known, point estimates are influenced by the survey weights, but not by other aspects of the survey design. On the other hand, this design can strongly influence the standard errors. Consequently, estimates of confidence intervals can be misleading if researchers do not take account of the survey design, particularly if their subjects are clustered.

For more discussion on this issue see the *Stata* manual entry "Remarks and Examples" for the `svy` command.

As recommended in the *Stata* manual, `if` or `in` conditions when used with the various **svy** command should make use of the subpopulation option. This approach is implemented in **tabout** behind the scenes, allowing the user to simply specify `if` or `in` as they normally would.

6.1. Basic tables with survey data

The basic tables produced by **tabout** when the `svy` option is chosen provide the same cell contents as the usual basic tables (except for the cumulative percentage). The main restriction is that only one of the set (ie. `freq`, `cell`, `row` and `col`) can be specified. On the other hand, you are not limited to what else is specified: you can use any number of the standard survey estimates, such as standard errors (`se`), confidence intervals (`ci`), lower bounds (`lb`) and upper bounds (`ub`). In other words, basic tables with survey data are intended to provide the user with one key point estimate, and a number of possible uncertainty estimates.

$\Longrightarrow$

A useful trick if you want confidence intervals (or similar) and you don't usually make use of *Stata* svy commands, is to simply set your weight variable using the `svyset` command, and then treat your data as if it were complex survey data. There is no need to specify either strata or PSU. If you have no weights, just create a weight variable equal to 1, and then `svyset` that. Then use the `svy` option in **tabout**.

One major disadvantage in using the **svy** commands in *Stata* is that the standard displays produced are not suited to publication-quality tables. For example, consider the following output, which are the default tables from the following commands:

```
use "http://www.stata-press.com/data/r14/nhanes2b", clear
svyset psuid [pweight=finalwgt], strata(stratid)
svy: tabulate race diabetes, row ci format(%7.3f)
svy: tabulate sex diabetes, row ci format(%7.3f)
```

Table 15: Standard Stata output

```
Number of strata   =        31                  Number of obs     =       10,349
Number of PSUs     =        62                  Population size   =  117,131,111
                                                Design df         =           31

------------------------------------------------------------------------------
1=white,        |
2=black,        |          diabetes, 1=yes, 0=no
3=other         |           0              1           Total
----------------+-------------------------------------------------------------
          White |       0.968          0.032           1.000
                |  [0.964,0.972]   [0.028,0.036]
                |
          Black |       0.941          0.059           1.000
                |  [0.927,0.952]   [0.048,0.073]
                |
          Other |       0.980          0.020           1.000
                |  [0.957,0.991]   [0.009,0.043]
                |
          Total |       0.966          0.034           1.000
                |  [0.962,0.969]   [0.031,0.038]
------------------------------------------------------------------------------
  Key:  row proportion
        [95% confidence interval for row proportion]

  Pearson:
    Uncorrected   chi2(2)         =   21.3483
    Design-based  F(1.52, 47.26)  =   15.0056     P = 0.0000
```

```
Number of strata   =        31                  Number of obs     =       10,349
Number of PSUs     =        62                  Population size   =  117,131,111
                                                Design df         =           31

------------------------------------------------------------------------------
1=male,         |
2=female        |          diabetes, 1=yes, 0=no
                |           0              1           Total
----------------+-------------------------------------------------------------
            Male|       0.971          0.029           1.000
                |  [0.965,0.976]   [0.024,0.035]
                |
          Female|       0.961          0.039           1.000
                |  [0.955,0.966]   [0.034,0.045]
                |
           Total|       0.966          0.034           1.000
                |  [0.962,0.969]   [0.031,0.038]
------------------------------------------------------------------------------
  Key:  row proportion
        [95% confidence interval for row proportion]

  Pearson:
    Uncorrected   chi2(1)         =    7.4897
    Design-based  F(1, 31)        =    6.2012     P = 0.0183
```

Not only is the layout problematic for easily converting into a publication-quality table, but the confidence intervals are on the row beneath the point estimates and this may not always suit your needs. As well, two separate tables are needed to conduct this particular analysis. By contrast, here is how **tabout** deals with this analysis and consolidates the results into a single table.

Table 16: Survey data example

	Diabetes					
	No		*Yes*		*Total*	*Sample size*
	Prop.	CI	Prop.	CI	Prop.	
Race						
White	0.968	[0.964-0.972]	0.032	[0.028-0.036]	1.000	9,063
Black	0.941	[0.927-0.952]	0.059	[0.048-0.073]	1.000	1,086
Other	0.980	[0.957-0.991]	0.020	[0.009-0.043]	1.000	200
Total	0.966	[0.962-0.969]	0.034	[0.031-0.038]	1.000	10,349
Pearson: Uncorrected chi2(2) = 21.348						
Design-based F(1.52, 47.26) = 15.006						
P-value = 0.000						
Sex						
Male	0.971	[0.965-0.976]	0.029	[0.024-0.035]	1.000	4,915
Female	0.961	[0.955-0.966]	0.039	[0.034-0.045]	1.000	5,434
Total	0.966	[0.962-0.969]	0.034	[0.031-0.038]	1.000	10,349
Pearson: Uncorrected chi2(1) = 7.490						
Design-based F(1.00, 31.00) = 6.201						
P-value = 0.018						

Source: nhanes2b

Copy table 16 code as: txt tex docx xlsx htm

```
use "http://www.stata-press.com/data/r14/nhanes2b", clear
svyset psuid [pweight=finalwgt], strata(stratid)

la var race "Race"
la var sex "Sex"
la var diabetes "Diabetes"
la def ny 0 "No" 1 "Yes", modify
la val diabetes ny

tabout race sex diabetes using table16.tex, replace ///
style(tex) font(italic) c(row ci) f(3) stats(chi2) ///
cisep(-) h2c(2 2 1) ltrim(2) svy npos(col) nlab(Obs) ///
twidth(16) title(Table 16: Survey data example) ///
fn(Source: nhanes2b)
```

Note the use of the `svy` option. This tells **tabout** that this table is a survey table. Keep in mind that you still need to tell *Stata* itself that this is survey data—using the `svyset` command—and what the appropriate survey settings are (such as the weight and the strata). The other new feature in this code is the `c(row ci)` option, which tells **tabout** that you want row percentages and *confidence intervals*. The `cisep(-)` option allows you to specify how the confidence intervals are *separated*, in this case with a hyphen. You can specify any symbol you like.

When it comes to brackets, the default for confidence intervals are square brackets and the default for standard errors are parentheses. The two options, `cibnone` and `sebnone` suppress each of these brackets respectively. Keep in mind that it's not necessary to have confidence intervals in one column—which can make for awkward layout—because **tabout** also allows you to specify lower bound and upper bound values in separate columns (as shown in Table 17 below).

For those using LaTeX, Table 16 illustrates the use of the `ltrim` option which makes the gaps in these lines larger than the default, which is 1 (measured in ems). Having gaps in heading lines may seem a small refinement, but they make the table more readable.

For LaTeX users, Table 16 shows an additional feature of `h2c`, the option which was introduced earlier as the 'partner' to the `h2` option. Here `h2c` is 'flying solo', instructing **tabout** where to put the gaps in the heading row. It does not need the `h2` option because it uses the default labels. Why bother using `h2c` at all? Without it, the default lines also extend to the final column, as shown in Table 17 below, where the 'Obs' are also underlined. This may or may not suit you. If it doesn't you can override the default lines by specifying your own, as shown in Table 16. In various tables which follow, I use this `h2c` option for the lines, so you should compare those tables with and without this feature to see which you prefer.

Turning now to Table 17, it repeats Table 16 but with the `contents` option using the `ub` (*upper bound*) and `lb` (*lower bound*) choices to display the confidence intervals.

Table 17: Survey data example with lb and ub options

	Diabetes							
	No			*Yes*			*Total*	*Obs*
	Prop.	LB	UB	Prop.	LB	UB	Prop.	
Race								
White	0,968	0,964	0,972	0,032	0,028	0,036	1,000	9.063
Black	0,941	0,927	0,952	0,059	0,048	0,073	1,000	1.086
Other	0,980	0,957	0,991	0,020	0,009	0,043	1,000	200
Total	0,966	0,962	0,969	0,034	0,031	0,038	1,000	10.349
Pearson: Uncorrected chi2(2) = 21,348								
Design-based F(1,52, 47,26) = 15,006								
P-value = 0,000								
Sex								
Male	0,971	0,965	0,976	0,029	0,024	0,035	1,000	4.915
Female	0,961	0,955	0,966	0,039	0,034	0,045	1,000	5.434
Total	0,966	0,962	0,969	0,034	0,031	0,038	1,000	10.349
Pearson: Uncorrected chi2(1) = 7,490								
Design-based F(1,00, 31,00) = 6,201								
P-value = 0,018								

Source: nhanes2b. Note: lb lower bound; ub upper bound.

Copy table 17 code as: txt tex docx xlsx htm

```
use "http://www.stata-press.com/data/r14/nhanes2b", clear
svyset psuid [pweight=finalwgt], strata(stratid)

la var race "Race"
la var sex "Sex"
la var diabetes "Diabetes"
la def ny 0 "No" 1 "Yes", modify
la val diabetes ny

tabout race sex diabetes using table17.tex, replace ///
style(tex) font(italic) twidth(16) c(row lb ub) svy ///
stats(chi2) f(3) dpc npos(col) nlab(Obs) ///
title(Table 17: Survey data example with ///
lb and ub options) fn(Source: nhanes2b. ///
Note: lb lower bound; ub upper bound.)
```

Table 17 also illustrates the use of the `dpc` (*decimal point comma*) option which allows European users to specify commas for decimal points and periods for thousands. The default labels (LB and UB) are used here, but you could replace them with something else using the `clab` option. In this case, the table footnote (via `fn()`) explains to the reader what the abbreviations mean.

⟹

Thanks to Arjan Soede for contributing code for the `dpcomma` option.

⟹

⟹

The next example, Table 18, shows the same data, but with further refinements. This time the point estimate is shown as a percentage. While this is the normal measure in a standard cross-tabulation, *Stata*'s **svy: tabulate** command usually presents its results as proportions. To get a percentage (as in this table) you need to use **tabout**'s `mult` (*multiplier*) option set to 100. If you wanted rates (for example, mortality rates) based on another denominator, such as 1000, you would specify `mult(1000)`.

⟹

The `mult` option replaces the `percent` option from **tabout** Version 2. Thanks to Peter Young for this contribution.

The uncertainty measure here is the standard error, and by default **tabout** presents this with parentheses. If you wish to remove these, the option `seb` will do this. (The full option name is `sebnone`, which means *SE brackets none*). Note that the cell formatting has changed, with `f(1 3)` specifying that the percentages are to be shown to 1 decimal point and and the standard errors to 3.

⟹

⟹

Table 18: Survey data example with standard errors

	Diabetes					
	No		*Yes*		*Total*	*Obs*
	%	SE	%	SE	%	
Race						
White	96.8	(0.197)	3.2	(0.197)	100.0	9,063
Black	94.1	(0.614)	5.9	(0.614)	100.0	1,086
Other	98.0	(0.764)	2.0	(0.764)	100.0	200
Total	96.6	(0.181)	3.4	(0.181)	100.0	10,349
Pearson: Uncorrected chi2(2) = 21.348						
Design-based F(1.52, 47.26) = 15.006						
P-value = 0.000						
Sex						
Male	97.1	(0.264)	2.9	(0.264)	100.0	4,915
Female	96.1	(0.267)	3.9	(0.267)	100.0	5,434
Total	96.6	(0.181)	3.4	(0.181)	100.0	10,349
Pearson: Uncorrected chi2(1) = 7.490						
Design-based F(1.00, 31.00) = 6.201						
P-value = 0.018						

Source: nhanes2b.

Copy table 18 code as: txt tex docx xlsx htm

```
use "http://www.stata-press.com/data/r14/nhanes2b", clear
svyset psuid [pweight=finalwgt], strata(stratid)

la var race "Race"
la var sex "Sex"
la var diabetes "Diabetes"
la def ny 0 "No" 1 "Yes", modify
la val diabetes ny

tabout race sex diabetes using table18.tex, replace ///
style(tex) font(italic) twidth(12) c(row se) svy ///
stats(chi2) f(1 3) mult(100) npos(col) h2c(2 2 1) ///
nlab(Obs)  title(Table 18: Survey data ///
example with standard errors) fn(Source: nhanes2b.)
```

Yet another variation to this table is the simplification of the statistics. In Table 19, instead of showing all the statistics, just the p-values are shown. They are moved to a column and shown as significance stars. The `seb` option, mentioned earlier, is also used here to suppress the parentheses around the standard errors. ⟹

Table 19: Survey data with stars for significance

	Diabetes						
	No		*Yes*		*Total*	*Obs*	*Signif*
	%	SE	%	SE	%		
Race							
White	96.8	0.197	3.2	0.197	100.0	9,063	***
Black	94.1	0.614	5.9	0.614	100.0	1,086	
Other	98.0	0.764	2.0	0.764	100.0	200	
Total	96.6	0.181	3.4	0.181	100.0	10,349	
Sex							
Male	97.1	0.264	2.9	0.264	100.0	4,915	*
Female	96.1	0.267	3.9	0.267	100.0	5,434	
Total	96.6	0.181	3.4	0.181	100.0	10,349	

Source: nhanes2b.

Copy table 19 code as: txt tex docx xlsx htm

```
use "http://www.stata-press.com/data/r14/nhanes2b", clear
svyset psuid [pweight=finalwgt], strata(stratid)

la var race "Race"
la var sex "Sex"
la var diabetes "Diabetes"
la def ny 0 "No" 1 "Yes", modify
la val diabetes ny

tabout race sex diabetes using table19.tex, replace ///
style(tex) font(italic) twidth(14) c(row se) svy seb ///
f(1 3) mult(100) npos(col) nlab(Obs) plab(Signif) ///
stats(chi2) stpos(col) ppos(only) stars ///
title(Table 19: Survey data with stars for ///
significance) fn(Source: nhanes2b.)
```

The syntax here might seem a little odd, but there is a logical basis to it. You specify a position for the statistics (`stpos(col)`) in order to change from the default position (which is the row location). But then the position for the p-value is used to drop the statistic, that is, `ppos(only)`. The permissible options (which are self-explanatory) for `ppos` are: *`none`*, *`only`*, *`below`*, *`beside`*. If *`beside`* is chosen, the p-value would be in a separate column (otherwise it sits below, which is the default). Finally, by specifying `stars` you are telling **tabout** to convert the p-value to asterisks using the conventional significance levels. If you want to change the label for this column, that is possible with the `plab` option.

Keep in mind that many of these options apply to all tables in **tabout**, not just survey tables. The basic tables can also be shown with stars and placed in columns. What makes survey tables unique is that measures of uncertainty, like standard errors and confidence intervals, can be included in the table cells.

Users should 'hack' the tabout ado file if they want to customise settings for which no option has been provided. A good example is the `stars` option. Just edit the `tabout.ado` file and change the code (a search on the phrase "***" will find it). You might prefer different cut-points, or a p-value range rather than stars. You would still type the `stars` option in the **tabout** syntax, but the tables would now reflect your own preference.

6.2. Summary tables with survey data

Summary tables with complex survey data are more limited than the summary tables presented earlier. Whereas the earlier tables allowed multiple summary measures in twoway tables, the `svy` option only allows for the mean. But this can then be followed by the uncertainty values mentioned earlier (*`se ci lb ub`*).

In Table 20 we see a similar set of options as in the earlier survey tables, but with an important new option: `sum`. As noted earlier, it is the `sum` option which tells **tabout** that you require summary measures. So when combined with the `svy` option, you get a summary table with survey data. But do keep in mind that you still need to tell *Stata* (rather than **tabout**) that your data is `svyset`, in this case, with a probability weight equal to `wt`. The other interesting feature here is the position of the sample size: in parentheses next to the labels, achieved with `npos(lab)`. This is similar to the *`tufte`* option, which was shown earlier and which placed a distributional breakdown (in percentages) next to the labels. Here the distribution is simply counts.

Notice also in Table 20 that the `clab` option did not need to specify something in every column because **tabout** knows that these headings are repeated for each of the categories in your cross-tabulation. This differs from the use of `clab` in oneway tables, where each column is unique and needs to be labelled accordingly (see the detailed discussion on labelling columns in oneway tables).

Table 20: Twoway summary table with survey data

	Education					
	Not college graduate		*College graduate*		*Total*	
	Mean wage	SE	Mean wage	SE	Mean wage	SE
Occupation						
Professional/technical (n=317)	10.01	(0.70)	12.22	(0.68)	11.04	(0.49)
Managers/admin (n=264)	10.10	(0.65)	14.40	(0.99)	11.29	(0.56)
Sales (n=726)	6.68	(0.18)	9.33	(0.98)	6.98	(0.20)
Clerical/unskilled (n=102)	8.52	(1.05)	9.05	(1.40)	8.62	(0.89)
Craftsmen (n=53)	7.03	(0.72)	10.47	(1.98)	7.48	(0.71)
Operatives (n=246)	5.65	(0.28)	4.21	(1.54)	5.62	(0.28)
Transport (n=28)	3.03	(0.31)			3.03	(0.31)
Laborers (n=286)	4.66	(0.22)	6.46	(1.12)	4.78	(0.22)
Farmers (n=1)			8.05	(0.00)	8.05	(0.00)
Farm laborers (n=9)	3.14	(0.28)	2.51	(0.00)	3.13	(0.28)
Service (n=16)	5.74	(0.46)	4.03	(0.00)	5.54	(0.44)
Household workers (n=2)	6.24	(0.08)			6.24	(0.08)
Other (n=187)	4.83	(0.68)	9.23	(0.31)	8.95	(0.31)
Total (n=2,237)	6.88	(0.15)	10.59	(0.32)	7.75	(0.14)
Location						
Does not live in the South (n=1,304)	7.58	(0.22)	10.92	(0.42)	8.37	(0.20)
Lives in the South (n=942)	5.94	(0.19)	10.12	(0.50)	6.90	(0.20)
Total (n=2,246)	6.88	(0.15)	10.59	(0.32)	7.74	(0.14)
Race						
White (n=1,637)	7.33	(0.20)	10.28	(0.32)	8.09	(0.17)
Black (n=583)	5.78	(0.17)	11.93	(1.02)	6.79	(0.25)
Other (n=26)	6.68	(0.81)	10.48	(2.50)	7.81	(0.98)
Total (n=2,246)	6.88	(0.15)	10.59	(0.32)	7.74	(0.14)

Source: nslw88.dta

Copy table 20 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
svyset [pw=wt]
tabout occupation south race collgrad using ///
table20.tex, replace style(tex) font(italic) sum svy ///
c(mean wage se) f(2 2) clab(Mean_wage SE) npos(lab) ///
twidth(15) title(Table 20: Twoway summary table with ///
survey data)  fn(Source: nslw88.dta)
```

Most of the tables shown in this *User Guide* employ the default column layout (either column or column block). The reason for this is both aesthetic and informational. Not only is the appearance more compact, with more data available in a smaller footprint, but glancing across columns when each row is the 'same' is much easier. There are times, however, when a row layout is more convenient. The default *Stata* survey tables, for ex-

ample, placed confidence intervals below the point estimates, and this can be very useful when horizontal space is limited. One can achieve this outcome quite easily with **tabout** by using the row layout option (`layout(row)`), as shown in Table 21.

There are several things worth noting about Table 21. The most important thing to realise is that when the row layout is selected, **tabout** sees this as a kind of *table rotation*, in which what were column labels become row labels. For this reason, Table 21 uses the `clab` option to insert a label for the confidence interval row, but it leaves alone the row above, where the variable category label (and the n count) are displayed. This is achieved by indicating, with the underscore, that this 'column' (ie. 'row') should be 'blank'.

tabout still views the top three rows as headings (`h1`, `h2` and `h3`) so these can be also manipulated. In this case, `h3` is used to span the label 'Average wage' across the columns. In this example, LaTeX code has been written to span the columns but this can also be done using `h3c` in the same way that `h2c` was used earlier.

Another example of this can be found in Table 29 below, where `h3c` is also used. Finally, notice that Table 21 uses the `level(90)` option, which tells **tabout** that the confidence intervals should be 90% probability levels, rather than the default 95% levels.

In the real world, one would probably leave out the customised heading on the third row, and just indicate in the footnotes or in the table title that the data contained average wages. Similarly, labelling every second row as a confidence interval is also probably unnecessary, and actually looks quite ugly. It has been done in this case to illustrate the capability, rather than to recommend the practice.

Table 21: Twoway summary table with row layout

	Education		
	Not college graduate	*College graduate*	*Total*
	Average wage		
Occupation			
Professional/technical (n=317)	10.01	12.22	11.04
(90% CI)	[8.86,11.17]	[11.10,13.33]	[10.23,11.85]
Managers/admin (n=264)	10.10	14.40	11.29
(90% CI)	[9.03,11.17]	[12.77,16.03]	[10.37,12.21]
Sales (n=726)	6.68	9.33	6.98
(90% CI)	[6.39,6.97]	[7.72,10.95]	[6.66,7.30]
Clerical/unskilled (n=102)	8.52	9.05	8.62
(90% CI)	[6.79,10.24]	[6.74,11.35]	[7.15,10.09]
Craftsmen (n=53)	7.03	10.47	7.48
(90% CI)	[5.84,8.21]	[7.21,13.72]	[6.31,8.65]
Operatives (n=246)	5.65	4.21	5.62
(90% CI)	[5.19,6.11]	[1.67,6.75]	[5.17,6.07]
Transport (n=28)	3.03		3.03
(90% CI)	[2.53,3.53]		[2.53,3.53]
Laborers (n=286)	4.66	6.46	4.78
(90% CI)	[4.30,5.03]	[4.62,8.30]	[4.41,5.14]
Farmers (n=1)		8.05	8.05
(90% CI)		[8.05,8.05]	[8.05,8.05]
Farm laborers (n=9)	3.14	2.51	3.13
(90% CI)	[2.68,3.60]	[2.51,2.51]	[2.68,3.59]
Service (n=16)	5.74	4.03	5.54
(90% CI)	[4.98,6.50]	[4.03,4.03]	[4.82,6.27]
Household workers (n=2)	6.24		6.24
(90% CI)	[6.10,6.38]		[6.10,6.38]
Other (n=187)	4.83	9.23	8.95
(90% CI)	[3.72,5.95]	[8.71,9.74]	[8.45,9.46]
Total (n=2,237)	6.88	10.59	7.75
(90% CI)	[6.63,7.13]	[10.05,11.12]	[7.52,7.99]
Location			
Does not live in the South (n=1,304)	7.58	10.92	8.37
(90% CI)	[7.22,7.94]	[10.23,11.62]	[8.04,8.70]
Lives in the South (n=942)	5.94	10.12	6.90
(90% CI)	[5.62,6.25]	[9.30,10.94]	[6.58,7.22]
Total (n=2,246)	6.88	10.59	7.74
(90% CI)	[6.63,7.13]	[10.05,11.12]	[7.51,7.98]

Source: nslw88.dta

Copy table 21 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
svyset [pw=wt]
tabout occupation south collgrad using table21.tex, ///
replace style(tex) font(italic) sum svy layout(row) ///
twidth(12) c(mean wage ci) f(2 2) clab(_ (90%_CI)) ///
level(90) npos(lab) h3(& \multicolumn{3}{c} ///
{Average wage} \\) title(Table 21: Twoway summary ///
table with row layout) fn(Source: nslw88.dta)
```

7. Reshaping tables

Version 2 of **tabout** had the capability of dropping the total rows in panels. This was useful if users wanted to avoid repetition, for example, when the numbers were the same in every totals row. All you had to do was specify the `ptotal(none)` option, and this could remove all the panel totals, or all of them except for a single row at the bottom of the table (using `ptotal(single)`). This option is no longer supported in Version 3 of **tabout**, and instead users can now remove any row they wish. The Tips and Tricks section of this User Guide shows how to achieve ptotal-type outcomes in Version 3.

⟹

At this stage the code for the ptotal option remains in tabout.ado, but the results are quite buggy. Thanks to Kevin Baier for drawing my attention to this problem.

However, it was never possible in Version 2 to remove unwanted columns, and this was a very common request from users. Now, in Version 3 of **tabout** this facility is available. Not only can users remove columns of totals, but any other column specified. At the same time, users can plug 'holes' in the tables, such as columns where a category will always be missing (such as 'pregnant men') and which may cause the columns to move out of alignment. This will be demonstrated shortly. The same dropping and plugging facility also applies to rows and this should be used instead of the former `ptotal` option.

One problem with the ptotal option was that it could be misleading because it based the table totals on the totals in the last panel. These figures may or may not have been accurate for all the panels.

7.1. Removing unwanted columns or rows

The basis of this operation is index numbers, that is, the user specifies the particular column, or columns, by their numbered location. The option is called `dropc` for columns and `dropr` for rows. Before discussing the intricacies, here is an example of the problem, and the solution offered by **tabout**.

⟹ ⟹

In Table 22 the 'Other' category is very small, yet takes up two columns. The total column, while useful to immediately indicate that these are row percentages, is also taking up a lot of space. One could recode the 'Other' category to missing, and *Stata* would then ignore it, but the total counts would then be wrong, as would the percentages.

Table 22: Unnecessary columns

	Race							
	White		Black		Other		Total	
	No.	%	No.	%	No.	%	No.	%
Location								
Does not live in the South	5,248	81	1,086	17	123	2	6,458	100
Lives in the South	2,884	61	1,848	39	23	0	4,756	100
Total	8,133	73	2,934	26	146	1	11,213	100
Sex								
Male	1,999	74	689	25	17	1	2,705	100
Female	6,108	72	2,239	26	128	2	8,475	100
Total	8,107	73	2,928	26	146	1	11,180	100
Member of a union								
Not a union member	5,195	74	1,708	24	106	2	7,010	100
Union member	1,447	64	785	35	37	2	2,269	100
Total	6,642	72	2,493	27	143	2	9,279	100

Source: nlsw88.dta

Copy table 22 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout south sex union race [iw=wt] using ///
table22.tex, replace style(tex) font(bold) f(0c) ///
c(freq row) twidth(14)  ///
title(Table 22: Unnecessary ///
columns) fn(Source: nlsw88.dta)
```

The solution can be found in Table 23, where columns 6, 7 and 9 are dropped. The footnote explains that the 'Other' column is not shown, and the column labels indicate that these are 'row %'. The outcome is a more compact table, which—if space is tight—might be required.

Because of the need to get the indexing correct when dropping columns, using `dropc(6 7 9)`, **tabout** offers an additional category in the `show` option: namely *`prepost`*. This displays on your screen the matrices before and after the column reshaping. In this way, you can check if your indexing is correct. **tabout** has always had a `show` option (where the default is *`output`*) and if you selected *`all`* previously you saw the matrices which lay behind the tables. With this new option, you see two sets of matrices (pre and post).

Table 23: Dropping columns

	Race				
	White		Black		Total
	No.	Row %	No.	Row %	No.
Location					
Does not live in the South	5,248	81	1,086	17	6,458
Lives in the South	2,884	61	1,848	39	4,756
Total	8,133	73	2,934	26	11,213
Sex					
Male	1,999	74	689	25	2,705
Female	6,108	72	2,239	26	8,475
Total	8,107	73	2,928	26	11,180
Member of a union					
Not a union member	5,195	74	1,708	24	7,010
Union member	1,447	64	785	35	2,269
Total	6,642	72	2,493	27	9,279

Source: nlsw88.dta. Note that Other category has been omitted, so totals will not be accurate.

Copy table 23 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout south sex union race [iw=wt] using ///
table23.tex, replace style(tex) font(bold) f(0c) ///
c(freq row) twidth(12) dropc(6 7 9) show(prepost) ///
clab(No. Row_%) h2c(2 2 1) title(Table 23: Dropping ///
columns) fn(Source: nlsw88.dta. Note: "Other" ///
category has been omitted. Totals will not be accurate.)
```

Table 24 shows the same information, but with variables swapped around so that you can see how rows are dropped. While it's the same operation—dropping the 'Other' category from the race variable—the implementation is slightly different. Because **tabout** operates with panels, you need to specify both the *panel number* and the *row number*, thus `dropr(3:5)`. While the 3rd panel is obvious, the 5th row is less so. This is where the `show(prepost)` is helpful because it displays the hidden rows, namely the heading rows (h2 and h3) which are 'built-in' to every panel. The reason you don't normally see them is because part of **tabout**'s housekeeping involves replacing these headings with other appropriate labels. You may also notice phrases surrounded by ! marks. These are also part of the housekeeping side of **tabout**. In summary, there are always *two rows more* than the actual data rows in a panel, so you need to take this into account when you index.

Table 24: Dropping rows

	Member of a union					
	Not a union member		Union member		Total	
	No.	%	No.	%	No.	%
Location						
Does not live in the South	3,665	52	1,589	70	5,254	57
Lives in the South	3,345	48	680	30	4,024	43
Total	7,010	100	2,269	100	9,279	100
Sex						
Male	1,701	24	569	25	2,270	25
Female	5,288	76	1,694	75	6,982	75
Total	6,989	100	2,263	100	9,252	100
Race						
White	5,195	74	1,447	64	6,642	72
Black	1,708	24	785	35	2,493	27
Total	7,010	100	2,269	100	9,279	100

Source: nlsw88.dta. Note that Other category has been omitted, so totals will not be accurate.

Copy table 24 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout south sex race union [iw=wt] using ///
table24.tex, replace style(tex) font(bold) f(0c) ///
c(freq col) twidth(14) dropr(3:5) show(prepost) ///
title(Table 24: Dropping rows) fn(Source: nlsw88.dta. ///
Note: "Other" category has been omitted. Totals ///
will not be accurate.)
```

As with the `dropc` option, if you have multiple rows to drop, use the standard convention of a *space* separating your choices, but make sure that the colon has no spaces on either side. Thus, if you wanted to drop the 'male' category from the sex variable, as well as the 'other' category from the race variable, you could specify: `dropr(3:5 2:3)` or `dropr(2:3 3:5)`. Unless a total is involved, it doesn't matter what order you specify these rows in, but you must always have the panel number first, followed by the row within the panel. (If one of the rows is for a total, then it must be specified first in your list.)

Thanks to Anders Alexandersson for drawing my attention to this.

Unfortunately, at this stage, you can only drop columns or rows in the same table, but not both at the same time. It is also worth noting that these reshaping operations only apply to the data produced by the `contents` option. If you choose to have sample counts or statistics in columns, these operations take place *after* the reshaping.

Thanks to Janelle Downing for drawing my attention to this bug. Hopefully, it will be resolved before the next release.

⟹

7.2. Plugging gaps in tables

Plugging columns

In the data setup code shown earlier in the *User Guide*, an artificial category of pregnant was created in order to demonstrate a problem which can arise when cross-tabulations involve categories which are always missing. The `nlsw88.dta` dataset was modified to artificially make some of the respondents men, thus creating an 'impossible' cross-tabulation of 'pregnant men'. How does **tabout** present such a cross-tabulation? Table 25 demonstrates the problem. The 'gaps' in themselves are not a problem, but their location is: the columns have moved out of alignment. Sometimes, you may want to just correct the alignment. At other times you may want to also fill the gap with something meaningful, such as 0 or 'NA' or a dash.

When **tabout** produces a table like this, it issues a warning:

> *Warning: not all panels have the same number of columns. Include show(all) in your syntax to view panels. Consider using the missing option for twoway tables, or the plug option for summary tables or twoway tables.*

One can fix this problem with **tabout**'s `mi` (*missing*) option, borrowed from **tabulate**, but this doesn't work for summary tables, only basic tables. Furthermore, if there are other missing values in the dataset, then extraneous columns or rows may suddenly appear if you use this missing option. ⟹

Table 25: Problem with gaps

	Sex					
	Male		Female		Total	
	No.	%	No.	%	No.	%
Location						
Does not live in the South	1,510	56	4,927	58	6,437	58
Lives in the South	1,195	44	3,548	42	4,743	42
Total	2,705	100	8,475	100	11,180	100
Currently pregnant						
Pregnant	2,972	35	2,972	35		
Not pregnant	5,503	65	5,503	65		
Total	8,475	100	8,475	100		
Member of a union						
Not a union member	1,701	75	5,288	76	6,989	76
Union member	569	25	1,694	24	2,263	24
Total	2,270	100	6,982	100	9,252	100
Race						
White	1,999	74	6,108	72	8,107	73
Black	689	25	2,239	26	2,928	26
Other	17	1	128	2	146	1
Total	2,705	100	8,475	100	11,180	100

Source: nlsw88.dta

Copy table 25 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout south pregnant union race sex [iw=wt] using ///
table25.tex, replace style(tex) font(bold)  ///
c(freq col) f(0c) twidth(14) title(Table 25: ///
Problem with gaps) fn(Source: nlsw88.dta)
```

The answer to this problem is a new feature in Version 3 of **tabout**, called *plugging*. Basically, you specify either columns or rows which have a gap, and **tabout** now fills that gap. The syntax follows the same principles as with dropping rows and columns, where you index the `plugc` or `plugr` options. Again, the `show(prepost)` option can be very useful for getting this indexing right. The default 'filler' is a blank, but you can use the `plugsym` (*plugsymbol*) option to specify something else (like 0, NA or a dash).

Table 26: Plugging columns

	Sex					
	Male		Female		Total	
	No.	%	No.	%	No.	%
Location						
Does not live in the South	1,510	56	4,927	58	6,437	58
Lives in the South	1,195	44	3,548	42	4,743	42
Total	2,705	100	8,475	100	11,180	100
Currently pregnant						
Pregnant	NA	NA	2,972	35	2,972	35
Not pregnant	NA	NA	5,503	65	5,503	65
Total	NA	NA	8,475	100	8,475	100
Member of a union						
Not a union member	1,701	75	5,288	76	6,989	76
Union member	569	25	1,694	24	2,263	24
Total	2,270	100	6,982	100	9,252	100
Race						
White	1,999	74	6,108	72	8,107	73
Black	689	25	2,239	26	2,928	26
Other	17	1	128	2	146	1
Total	2,705	100	8,475	100	11,180	100

Source: nlsw88.dta

Copy table 26 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout south pregnant union race sex [iw=wt] using ///
table26.tex, replace style(tex) font(bold) f(0c) ///
c(freq col) twidth(14) plugc(2:2 2:3) plugsym(NA) ///
show(prepost) title(Table 26: Plugging columns) ///
fn(Source: nlsw88.dta)
```

In the case of Table 26, the pregnant panel was the second panel, and therefore the column headings were straightforward: they matched the other variables. But if you wanted the pregnant panel to come first, and it had this gap in it, you would have an additional problem, as Table 27 shows. This table shows that the labels for both heading row 2 and heading row 3 are missing, and need to be filled. (By way of illustration, notice that the `plugsym` has been omitted, and the panel cells are thus left blank.) ⟹

Table 27: Plugging columns - a problem

			Sex			
			Female		Total	
			No.	%	No.	%
Currently pregnant						
Pregnant			2,972	35	2,972	35
Not pregnant			5,503	65	5,503	65
Total			8,475	100	8,475	100
Location						
Does not live in the South	1,510	56	4,927	58	6,437	58
Lives in the South	1,195	44	3,548	42	4,743	42
Total	2,705	100	8,475	100	11,180	100
Member of a union						
Not a union member	1,701	75	5,288	76	6,989	76
Union member	569	25	1,694	24	2,263	24
Total	2,270	100	6,982	100	9,252	100
Race						
White	1,999	74	6,108	72	8,107	73
Black	689	25	2,239	26	2,928	26
Other	17	1	128	2	146	1
Total	2,705	100	8,475	100	11,180	100

Source: nlsw88.dta

Copy table 27 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout pregnant south union race sex [iw=wt] using ///
table27.tex, replace style(tex) font(bold) f(0c) ///
c(freq col) twidth(14) plugc(1:2 1:3) show(prepost) ///
title(Table 27: Plugging columns - a problem) ///
fn(Source: nlsw88.dta)
```

To fix this problem you need to make use of the `h2` and `h3` options so that the missing labels can be inserted as appropriate. As mentioned earlier, if you don't want to go to the trouble of writing your own spanning code, but you do want these to be labels to be properly positioned, you also need to apply the `h2c` and `h3c` options. In this case, the `h2c` option is similar to earlier: you work out the number of spans and place them in this option: `h2c(2 2 2)`.

Table 28: Plugging columns - fixing the problem

	Sex					
	Male		Female		Total	
	No.	%	No.	%	No.	%
Currently pregnant						
Pregnant			2,972	35	2,972	35
Not pregnant			5,503	65	5,503	65
Total			8,475	100	8,475	100
Location						
Does not live in the South	1,510	56	4,927	58	6,437	58
Lives in the South	1,195	44	3,548	42	4,743	42
Total	2,705	100	8,475	100	11,180	100
Member of a union						
Not a union member	1,701	75	5,288	76	6,989	76
Union member	569	25	1,694	24	2,263	24
Total	2,270	100	6,982	100	9,252	100
Race						
White	1,999	74	6,108	72	8,107	73
Black	689	25	2,239	26	2,928	26
Other	17	1	128	2	146	1
Total	2,705	100	8,475	100	11,180	100

Source: nlsw88.dta

Copy table 28 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout pregnant south union race sex [iw=wt] using ///
table28.tex, replace style(tex) font(bold) f(0c) ///
c(freq col) twidth(14) plugc(1:2 1:3) h2(Male ///
Female Total) h2c(2 2 2) h3(No. % No. % No. %) ///
h3c(1 1 1 1 1 1) title(Table 28: Plugging columns ///
- fixing the problem) fn(Source: nlsw88.dta)
```

The h3c option is a little odd. You *do* need to use this option—because this is what tells **tabout** to work out the proper positioning of the labels—even though there is no spanning involved. Hence the repetition of 1 six times. Now all of this is fairly tedious, but the use of these options is quite rare, because plugging gaps in tables is not a common occurrence and you always have the option of placing the problematic panel lower down in the table so that the heading rows are not affected. Thus while using h2c is fairly common, as with summary tables, the use of h3c (and h1c is rarely needed).

On a more positive note, having the less commonly used option of h3c available does open up creative possibilities with your table headings. You can for instance span the columns in heading row 3 in this way without writing any code. For example, in Table 21 we saw the following code: h3(& \multicolumn{3}{c}{Average wage}\\).

As Table 29 shows, this can be replaced with the much simpler: h3(Average wage) and h3c(3) code. Not only does this make life simpler for users wanting *tex* and *htm* output, but it also makes it possible to get the same result with *docx* and *xlsx* output (where you are not able to do your own coding).

Table 29: Twoway summary table showing use of h3c

	Education		
	Not college graduate	*College graduate*	*Total*
	Average wage		
Location			
Does not live in the South (n=1,304)	7.58	10.92	8.37
	[7.22,7.94]	[10.23,11.62]	[8.04,8.70]
Lives in the South (n=942)	5.94	10.12	6.90
	[5.62,6.25]	[9.30,10.94]	[6.58,7.22]
Total (n=2,246)	6.88	10.59	7.74
	[6.63,7.13]	[10.05,11.12]	[7.51,7.98]

Source: nslw88.dta

Copy table 29 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
svyset [pw=wt]
tabout south collgrad using table29.tex, replace ///
style(tex) font(italic) twidth(12) c(mean wage ci) ///
f(2 2) clab(_ _ _) sum svy level(90) layout(row) ///
npos(lab) h3(Average_wage) h3c(3) title(Table 29: ///
Twoway summary table using h3c)  fn(Source: nslw88.dta)
```

Table 29 also illustrates the clab option being used to remove the default CI label from each second row. Note that the underscores are needed to tell **tabout** that these are empty spaces. ⟹

Finally, a word of caution with the use of all the customised headings, optionh1, h2 and h3. If you do use them, you must take account of the whole row. While you can avoid writing code to deal with spanning columns and positioning labels, you may still need to add in some additional labels which **tabout** normally takes care of for you. For example, if you've chosen to place N counts in your column, using the npos(col) option, you may need to include a label for this. Similarly, if you've got ⟹

some statistics labels in the columns you will need to take care of these as well. Why is this? If you stick with **tabout** normal headings, these features are taken care of automatically. But the customised headings are applied right towards the end of the table production process, so they over-ride the earlier settings. This is necessary to give you complete control over your headings (if you need that flexibility). This is another instance of the trade off between complexity and flexibility.

Plugging rows

This ability to plug gaps also extends to rows, though this feature is much less often required since the omission of a row does not cause alignment problems. To illustrate the problem and the solution, Tables 30 and 31 repeat the pregnancy problem from a row perspective. Note the use in Table 31 of the `pluglab` option to specify a row label, and the `plugsym` option to specify the empty cell contents in this row. The actual `plugr` option uses the same convention as for columns: a panel number, a colon, and a row number. You need to keep in mind that there are always 2 more rows than shown in the data for a panel. Hence 4:3 means panel number four, insert at the first row. If you get confused, turn on `show(prepost)` and study the matrix from which the table is built. (See the discussion of this issue earlier.)

Table 30: Plugging rows - the problem

	Currently pregnant					
	Pregnant		Not pregnant		Total	
	No.	%	No.	%	No.	%
Location						
Does not live in the South	1,913	39	3,035	61	4,948	100
Lives in the South	1,058	30	2,502	70	3,560	100
Total	2,972	35	5,536	65	8,508	100
Member of a union						
Not a union member	1,708	32	3,601	68	5,309	100
Union member	694	41	1,005	59	1,700	100
Total	2,402	34	4,607	66	7,009	100
Race						
White	2,212	36	3,921	64	6,134	100
Black	713	32	1,533	68	2,246	100
Other	47	36	82	64	129	100
Total	2,972	35	5,536	65	8,508	100
Sex						
Female	2,972	35	5,503	65	8,475	100
Total	2,972	35	5,503	65	8,475	100

Source: nlsw88.dta

Copy table 30 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout south union race sex pregnant [iw=wt] using ///
table30.tex, replace style(tex) font(bold) f(0c) ///
c(freq row) twidth(12) title(Table 30: Plugging rows ///
- the problem) fn(Source: nlsw88.dta)
```

Table 31: Plugging rows - the solution

	Currently pregnant					
	Pregnant		Not pregnant		Total	
	No.	%	No.	%	No.	%
Location						
Does not live in the South	1,913	39	3,035	61	4,948	100
Lives in the South	1,058	30	2,502	70	3,560	100
Total	2,972	35	5,536	65	8,508	100
Member of a union						
Not a union member	1,708	32	3,601	68	5,309	100
Union member	694	41	1,005	59	1,700	100
Total	2,402	34	4,607	66	7,009	100
Race						
White	2,212	36	3,921	64	6,134	100
Black	713	32	1,533	68	2,246	100
Other	47	36	82	64	129	100
Total	2,972	35	5,536	65	8,508	100
Sex						
Male*	0	0	0	0	0	0
Female	2,972	35	5,503	65	8,475	100
Total	2,972	35	5,503	65	8,475	100

Source: nlsw88.dta. Note: * No male pregnancies have been reported.

Copy table 31 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout south union race sex pregnant [iw=wt] using ///
table31.tex, replace style(tex) font(bold) f(0c) ///
c(freq row) twidth(12) plugr(4:3) pluglab(Male*) ///
plugsym(0) title(Table 31: Plugging rows - the ///
solution) fn(Source: nlsw88.dta. Note: * No male ///
pregnancies have been reported.)
```

8. Template Files

8.1. Introduction

The syntax code for **tabout** can become quite long, extending over half a dozen lines or so. This often happens when the user wants to specify a large number of options and this is, unfortunately, the price paid for the flexibility available with **tabout**. To try to make this manageable, **tabout** has many small options, with almost cryptic names, as well as allowing abbreviations (as is common in *Stata*). However, the practice of copying and pasting from the code for one table to that for another, with minor edits along the way, is not ideal. Errors can be easily made, though they are usually picked up when the final table doesn't turn out the way you wanted. Template files are a way to deal with this problem.

What is a template file? Basically, it's just a plain text file, with each **tabout** option on a new line. You can create them easily by just editing the do file which contains your **tabout** code and saving it with a different extension (`txt` is best, but you can choose anything). When you edit this you need to remove extraneous code and make sure each **tabout** option is on a new line. *The very first line of the template file must not contain any options, but is left for you to use as you see fit.* Putting a description of what the template does is a good idea, and perhaps the date of creation.

What goes into a template file? Essentially, you should divide the code for any particular table into its *generic* options and its *specific* options. Thus the *variable list* in the table would be specific, as would the *file name* and the *table title*. These specific features would go into the **tabout** code in your do file. But things like the style, the format, the layout, the labelling, and perhaps even the footnotes, would generally go into the template file. Your can then attach that template to a batch of tables using the `tp` option. ⟹ When you run your main **tabout** do file, each table will show different results, depending on the specific code for that table, but there will be a common set of characteristics (the generic features) across all the tables. The template file provides the *generic* code which **tabout** needs to make the final tables. Basically, **tabout** composes the table by mixing your *specific* code (in the do file) with the *generic* code (in the template file) to produce the table you would have produced had you done the whole process 'long hand'.

8.2. Advantages

The advantages of template files are obvious:

- the complexity in your code is reduced considerably;
- you can share template files with your colleagues, or with the wider *Stata* community;
- **tabout** novices can be given template files to make the learning curve easier;
- single source publishing is made easier: one set of template files might produce a LaTeX document for PDF distribution, another set for `html` files for the website and another set for *Excel* files for archiving;
- tables for journals or reports can change their appearance to suit the audience by switching templates;
- consistent tables for batch production are made easy, for example, appendix tables or standardised reports.

8.3. The details

Here is the code from Table 9, as it appeared earlier:

```
do nlsw_data_setup
tabout industry occupation [iw=wt] using table9.tex, ///
replace style(tex) font(bold) oneway c(freq col cum) ///
f(0c 1) clab(Count Col_% Cum_%) twidth(11) npos(col) ///
nlab(Sample) title(Table 9: A oneway table) ///
fn(Source: nlsw88.dta)
```

It is obvious that much of this code is generic, hence the following template file is created. It is saved as plain text, using the filename `oneway_basic_b.txt`. You might have noticed that the example files use `a`, `b`, `c`, `d` and `e` at the end of their names according to the output style. Most of the time you can ignore this, but it matters here with the template example, because the filename needs to be referenced. Not all templates have the same contents: they depend on the output style.

Copy template code as: txt tex docx xlsx htm

```
oneway_basic_b: In main body of report. Nov 2016.
replace
style(tex)
font(bold)
oneway
contents(freq col cum)
format(0c 1)
clab(Count Col_% Cum_%)
twidth(11)
npos(col)
nlab(Sample)
fn(Source: nlsw88.dta)
```

Notice that the first line is descriptive, but that every other line has a **tabout** option, specified in the same way as the original code (but without any line continuation (///) symbols). The `fn` option here is regarded as generic, but in some cases users might prefer to keep this with the specific code. In the final part of the *User Guide* I also demonstrate passing arguments to footnotes and titles using the older **tabout** `topf` and `botf` options, and this approach is well suited to this new template system.

The following example shows this specific code, as found in the user's do file used for creating tables. The result of running this code is Table 32, the same output as Table 9.

A couple of things worth noting. Abbreviations should be avoided in the template file. This is good practice because the file then becomes a bit more self-documenting. The `replace` option is usually placed here, unless one is building a set of tables into a single file using the `append` option. One can, of course, leave out `replace`, but if you run the code several times (as often happens in batch situations, or when first creating a table) you get annoying *Stata* messages.

Table 32: A oneway table using a template

	Count	Col %	Cum %	Sample
Industry				
Ag/Forestry/Fisheries	84	0.8	0.8	17
Mining	14	0.1	0.9	4
Construction	160	1.4	2.3	29
Manufacturing	1,848	16.6	18.9	367
Transport/Comm/Utility	433	3.9	22.8	90
Wholesale/Retail Trade	1,685	15.1	37.9	333
Finance/Ins/Real Estate	970	8.7	46.7	192
Business/Repair Svc	429	3.9	50.5	86
Personal Services	472	4.2	54.8	97
Entertainment/Rec Svc	99	0.9	55.6	17
Professional Services	4,151	37.3	92.9	824
Public Administration	786	7.1	100.0	176
Total	11,129	100.0		2,232
Occupation				
Professional/technical	1,477	13.2	13.2	317
Managers/admin	1,322	11.8	25.1	264
Sales	3,626	32.5	57.5	726
Clerical/unskilled	511	4.6	62.1	102
Craftsmen	239	2.1	64.2	53
Operatives	1,305	11.7	75.9	246
Transport	136	1.2	77.1	28
Laborers	1,491	13.4	90.5	286
Farmers	8	0.1	90.5	1
Farm laborers	40	0.4	90.9	9
Service	75	0.7	91.6	16
Household workers	3	0.0	91.6	2
Other	938	8.4	100.0	187
Total	11,171	100.0		2,237

Source: nlsw88.dta

Copy table 32 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout industry occupation [iw=wt] using table32.tex, ///
tp(oneway_basic_b.txt) ///
title(Table 32: A oneway table using a template)
```

Finally, templates files are well suited to basic files, particularly the more complex ones, like survey tables. They are less well suited to some summary tables, where the `contents` option contains different variables and the various labelling options are unique to each table. In these situations, the template files may only be a few lines long and the **tabout** code in the do file may be hardly reduced at all.

⟹

9. Dynamic documents

9.1. Titles and footnotes

As you will have noticed in most of the example files, the two new features of Version 3—`title` and `fn`— have been used extensively to surround your table with titles and footnotes. Version 2 of **tabout** also provided this functionality, but it was bit more complicated. Users needed to have already created two files, `topf` and `botf`, and these would be imported by **tabout** at the appropriate point in the table production process and incorporated into the final table.

Does this mean that `title` and `fn` make the older versions obsolete? Not entirely. In one sense, they are a replacement, but they are retained in **tabout** because they allow much greater customisation of the whole table production process, particularly the production of dynamic documents. They should be regarded as complementary to the new versions and you may find yourself wanting to use both `topf` and `botf` in conjunction with `title` and `fn`.

Both `topf` and `botf` were designed to work with two other options, `topstr` and `botstr`, which allowed users to pass arguments (eg. sentences, words or phrases) from inside the body of their **tabout** code into these two files, using the latter as a form of 'boilerplate' text, which could be customised for each particular table by passing phrases appropriate to that table. This is the key to producing dynamic documents.

9.2. Beyond titles and footnotes

How does the `topf` and `botf` system work? The first thing to note is that these options are only available to users working with plain text formats and not binary formats. That is, they are suited to the *`tex`* and *`htm`* output styles (though delimited text file users can also make use of these options, although this is unlikely). While the `title` and `fn` options are available for all output styles, the kind of dynamic documents discussed in this chapter using this methodology is not viable for binary files such as `xlsx` or `docx`. However, *Word* users can still build dynamic tables if they select the `htm` output style, and a detailed example of this is provided in Chapter 13, the *`docx`* chapter.

The first step in creating dynamic documents involves creating two files using the appropriate code for your output style, either LaTeX or `html` code. These files can be saved with a `txt` extension, even if they contain code, because they are going to be 'woven' into the final **tabout** output. (Though you can use any file extension you like). Then, at the point where you want customised text to be inserted in these files, you insert a placeholder symbol. The default is #, but you can define something else if you like using the `psymbol` option. Note that in Version 2 of **tabout** each placeholder was required to be on separate line. This no longer applies in Version 3 and you can include large blocks of text in `topf` and intersperse placeholders anywhere you like. This has important implications for the flexibility of this system, as I will demonstrate below.

Finally, in your `topstr` or `botstr` options (that is, inside your **tabout** code in your do file) you include the customised text, with each string to be inserted separated by a pipe delimiter (or another delimiter you have chosen using the `delim` option). Thus if you have three strings you want to pass to your `botf` then there must be three # symbols in that file, and two pipe delimiters in your `botstr` option (one less than the number of strings because these are separator symbols).

The following shows this approach, repeating Table 9 but using `topf` and `botf` instead of `title` and `fn`. First, here is the contents of `topf`, which is saved under the file name `title.txt`:

Copy title.txt as: txt tex docx xlsx htm

```
{\bfseries Table #: #}
```

And here is the contents of `botf`, which is saved under the filename `footnote.txt`:

Copy footnote.txt as: txt tex docx xlsx htm

```
\vspace{-1ex}
\hspace{1.2em}
{\scriptsize{Source: nlsw88.dta}}
```

Here is the **tabout** code for the table itself:

Copy table 33 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout industry occupation [iw=wt] using table33.tex, ///
replace style(tex) font(bold) oneway c(freq col cum) ///
f(0c 1) clab(Count Col_% Cum_%) twidth(11) npos(col) ///
nlab(Sample) topf(title.txt) botf(footnote.txt) ///
topstr(33|Illustrating use of topf and botf) ///
botstr(nlsw88.dta)
```

Here is the table produced by this code (identical to the earlier Table 9, except for the wording of the title):

Table 33: Illustrating use of topf and botf

	Count	Col %	Cum %	Sample
Industry				
Ag/Forestry/Fisheries	84	0.8	0.8	17
Mining	14	0.1	0.9	4
Construction	160	1.4	2.3	29
Manufacturing	1,848	16.6	18.9	367
Transport/Comm/Utility	433	3.9	22.8	90
Wholesale/Retail Trade	1,685	15.1	37.9	333
Finance/Ins/Real Estate	970	8.7	46.7	192
Business/Repair Svc	429	3.9	50.5	86
Personal Services	472	4.2	54.8	97
Entertainment/Rec Svc	99	0.9	55.6	17
Professional Services	4,151	37.3	92.9	824
Public Administration	786	7.1	100.0	176
Total	11,129	100.0		2,232
Occupation				
Professional/technical	1,477	13.2	13.2	317
Managers/admin	1,322	11.8	25.1	264
Sales	3,626	32.5	57.5	726
Clerical/unskilled	511	4.6	62.1	102
Craftsmen	239	2.1	64.2	53
Operatives	1,305	11.7	75.9	246
Transport	136	1.2	77.1	28
Laborers	1,491	13.4	90.5	286
Farmers	8	0.1	90.5	1
Farm laborers	40	0.4	90.9	9
Service	75	0.7	91.6	16
Household workers	3	0.0	91.6	2
Other	938	8.4	100.0	187
Total	11,171	100.0		2,237

Source: nlsw88.dta

9.3. A legacy system or a brave new world?

You can see how cumbersome the `topf` and `botf` system is, so why would you use it? There are two main reasons. First, if you are a LaTeX or `html` user, these allow you to craft your own code for above and below your table. Version 3 of **tabout** not only offers `title` and `fn` options, but it also now automatically places code above and below tables. However, this may not suit all users. If they prefer to just use **tabout** for the table data, and wish to write their own table code, then the `topf` and `botf` options are what they need. An important point for these users is the `ntc` (*no table code*) option which can be used to instruct **tabout** to suppress this 'automatic' generation of table code. The goal of this kind of flexibility in **tabout** is to make both

`tex` and `htm` output styles easy for novices, but without frustrating the customisation aspirations of more experienced users.

The second reason why `topf` and `botf` are valuable is because they make possible *dynamic documents*. For example, you could create a short document incorporating a number of tables, with blocks of text interspersed between the tables, and the complete document could be compiled from inside *Stata*. While I don't imagine anyone would write a long report in this way, this approach would certainly suit short documents, particularly those which are required as regular outputs from data which is regularly updated.

Such an approach would use a nesting structure for the various **tabout** options. You can think of this as kind of Russian doll, with dolls inside dolls:

```
topbody
 topf
  title
    actual data in table produced by your tabout code
  fn
 botf
botbody
```

9.4. Dynamic documents

A short example

The following short document was created entirely within **tabout**, using the `topf` and `botf` options. The actual files are not shown here, but can be downloaded and examined. The text is the same as that shown here, but you will notice # symbols where ever dynamic data (all in bold) has been placed. The Latin sentences here are simply a filler, often used by programmers for pretend text.

The title of my short report

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

And now we have an important result: **3019.46** lbs. is the average weight of all vehicles. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. And a second important result: **187.93** inches is the average length. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

And now we have the first table.

Table 34: Short report example

	Car type (mean weight in lbs.)		
	Domestic	*Foreign*	*Total*
Repair Record 1978			
1	3,100		3,100
2	3,354		3,354
3	3,442	2,010	3,299
4	3,532	2,208	2,870
5	1,960	2,403	2,323
Total	3,368	2,263	3,032

auto.dta

And these results were based on **74** observations and were completed on **15 Nov 2016**.

The code for this example is shown below as Table 34. Notice the use of *Stata* macros to hold your dynamic data, plus a global system macro for the current date. These macros are simply passed as arguments in the `topstr` and `botstr` options, and are then inserted by **tabout** into the appropriate slots in your boilerplate text, found in the `topf` and `botf` files. Notice also, that `title` and `fn` are also used, which thus frees up `topf` and `botf` for the boilerplate text. An important point for LaTeX users if you wish to run this example: add the `body` option to the Table 34 code if you want to ⟹ compile this short report rather than paste the LaTeX output into an existing document (which is what I've done here). For novices, you might like to add `comp` and `open` to your Table 34 code so that this report document ⟹ ⟹ can be viewed as soon as you finish running it. I discuss the `comp` and `open` options in detail later, in the discussion of LaTeX output so you'll need to skip ahead and read about LaTeX first. For *`htm`* output the appropriate additional code to create free-standing documents is 'built-in' and you don't need to specify `body`. There are advantages in specifying `body`, such as cascading style sheets, and they will be discussed in Chapter 15.

Copy table 34 code as: `txt` `tex` `docx` `xlsx` `htm`

```
sysuse auto, clear
sum weight
local meanwt = r(mean)
local meanwt: di %3.2f 'meanwt'
sum length
local meanlen = r(mean)
local meanlen: di %3.2f 'meanlen'
local obs = r(N)

tabout rep78 foreign using table34b.tex, replace ///
style(tex) font(italic) c(mean weight) f(0c) sum  ///
twidth(9) h1(Car type (mean weight in lbs.)) h3(nil) ///
title(Table 34: Short report example) fn(auto.dta) ///
topf(topblock.txt) topstr('meanwt'| 'meanlen') ///
botf(bottomblock.txt) botstr('obs'|$S_DATE)
```

More complex reports

As well as simple example of a single table surrounded by dynamic data, you could also construct a short report with multiple tables and lots of dynamic data. The way you'd do this is as follows. Looking back at the Russian doll example: you'd use the `topbody` option for your first table and the `botbody` option for your last table, and just keep repeating the other options, for example, formatting options, for all the other tables. The trick to making this work is to use `replace` for your first **tabout** table, and `append` for all the rest. Note that you should use the *same output file name* for all your **tabout** commands.

Next, have your blocks of boilerplate already written, with their placeholder symbols in the right place. Then run your do file, containing a batch of **tabout** table commands, plus the various local macros which hold your dynamic data. **tabout** will create the document specified in the output file name, create the appropriate table and interspersed text, then when it reads the next **tabout** command, it will create the next table, with its interspersed text, and add that to the existing output document. When **tabout** finishes, you should find a single document, with all your tables and interspersed text properly laid out.

If things don't work as you expect, check that you have the same number of placeholder symbols in your `topf` file as you have delimited strings in your `topstr` option (and the same applies to `botf` and `botstr`. Also make sure that your `topf` and `botf` options specify the correct name for an existing file which contains your boilerplate text.

For example, if you were writing a report with six tables, your **tabout** syntax, in general terms, might look like this:

table1 uses `replace`, `topbody`, `topf`, `title`, `fn`, `botf`

tables 2 through to 5, for example, use `append`, `topf`, `title`, `fn`, `botf`

table6 uses `append`, `topf`, `title`, `fn`, `botf`, `botbody`

And some of the tables might use `topstr` and `botstr` to pass arguments to the blocks of text which live inside `topf` and `botf`.

Automatic newsletters and web pages

What are some practical applications of this? The example shown above is based on LaTeX and could form the basis for a newsletter which needed updating on a regular basis. Some **tabout** users might need to maintain a web page which is based on tables that need regular updating. With this `topf` and `botf` system, users with this requirement could create the whole web page inside *Stata* and even have *Stata* run an `ftp` command to upload the web page to the internet.

The limits to this system are only set by your imagination. In the same way that you can use *Stata*'s local macros to insert dynamic data in your newsletter or web page, you could also write code in your do file to generate graphics based on the dataset in memory, save those graphics to a file, and then use a file insert command in your boilerplate text to bring the graphics in at the appropriate place. An extended example of this is shown in Chapter 15. This whole system illustrates the underlying philosophy of **tabout**: 'files should talk to files' and your research work flow should be entirely reproducible.

Part III.

Table styles

10. Table styles: overview

10.1. What you see is not what you get

Some users get a shock when they first run **tabout** and see a messy table appear in the *Stata* output window. They have not understood that **tabout** is intended for exporting tables and that the table which appears in *Stata*'s output window may bear no relationship to what it looks like inside the application to which it is destined to go. In Version 3 of **tabout** it is much easier for users to see almost instantly what their tables will look like. I will discuss this further, below.

In the early days, most users of **tabout**, and other similar *Stata* commands, exported their results as tab delimited text and then read these into their spreadsheets or word processors. Plain text with delimiters (sometimes tab separated, sometimes comma separated) has been a universal approach to moving plain text between applications for many decades.

However, plain text files still require a lot of work in the application where they surface if the user wants publication quality tables, particularly formatting, alignment, spanning of columns, fonts and so forth. While **tabout** still supports tab and comma separated text files, users increasingly prefer well formatted tables with minimal extra work. Indeed, reproducible research requires no (or minimal) extra editing of tables. For this reason, **tabout** now supports the exporting of `xlsx` and `docx` files which can be read directly into applications such as *Excel* and *Word*, with all (or most) of the formatting etc done by **tabout**.

Of course, **tabout** has always supported both `tex` and `htm` output, which provide users with publication quality tables straight from **tabout**. No further editing at all is needed. With both these formats users can build very complex documents where text and tables are interspersed and reproducible research is almost perfectly achieved.

In summary, Version 3 of **tabout** retains the traditional delimited text file options, increases the flexibility and ease of use with `tex` and `htm` output files, and introduces the native `docx` and `xlsx` output files for users who prefer to work solely in spreadsheets and/or word processors. Finally, with the use of the `open` option users can almost instantly 'see what they get'.

10.2. Options and styles

The way that **tabout** caters for all of these different outputs is that certain options are only available with some of them, some options work differently with some of them, and some options are consistent across all of them.

Here are two examples: one showing variability, the other consistency. Some styles have a `twidth` option, which sets the table width. But this doesn't apply to *`txt`* and *`xlsx`* output styles; the latter has a `cwidth` (*column width*) and a `lwidth` (*label width*) option. As for the way these various widths are measured, the default measurement for *`tex`* is centimetres, for *`htm`* it is pixels, and for *`docx`* it is percentages. (These can all be changed with the `units` option.) On the other hand, the `font` option works almost the same across all styles.

The table of options has a comments column. Where an option only applies to some output styles, those output styles are shown in [text like this]. If no red text is shown, then that option is relevant to all output styles. In addition, **tabout** users can download all, or any, of the example files from the links in this *User Guide* and then compare the differences in the **tabout** code between the different styles.

Some of the common stylistic features available in Version 3 of **tabout** which apply to all output styles (except for delimited text files) are:

- borders: the outer 'thick' lines at the top and bottom of the table. These can be turned off with the `noborder` option.
- lines:
  - the inner 'thin' lines separating the heading rows. These can be turned off with the `nohlines` option.
  - the inner 'thin' lines separating the panels. These can be turned off with the `noplines` option.
- font: whether variable labels should be bold, italic or plain using the `font` option.
- font family: the font or typeface name which is used for your table (eg. Adobe Garamond Pro, Baskerville, Arial, etc) using the `family` option.
- font size: the point size of your font set using the `fsize` option. Note that the font sizes in your output are automatically changed by a certain ratio for the following features:
  - title: increased by 1.2
  - footnote: decreased by 0.8
  - statistics (in rows): decreased by 0.9
- table width: this can be set using `twidth` for these output styles,

with the following default settings:

- `tex`: 14cm
- `htm`: no width setting is applied if no `twidth` value is specified;
- `docx`: 'Autofit to Contents' is used if no `twidth` value is specified.

◁ the `units` option let you change the unit of measurement used by the various width options. ⟹

◁ for `htm`, `docx` and `xlsx` outputs you can also set the width of the labels column (that is, the first column) and the data columns using the `lwidth` (*label width*) and `cwidth` (*column width*) options. Whatever units have been set also are also applied to these settings (eg. `cm`, `px`, `%`). ⟹ ⟹

◁ for `tex` and `docx` output there is a `landscape` option which turns the table (`tex`) or the page containing the table (`docx`) from portrait into landscape mode. ⟹

◁ for `docx` and `tex` outputs there is a `tleft` (*table left*) option. By default, **tabout** centres the table on the page, but with this option you can set it be flush with the left margin. ⟹

◁ for `tex` output only there is a `rotate` option which allows users to rotate their value labels. ⟹

◁ for `htm` output only there is a `css` option which gives users the ability to produce coloured text, lines and background (as well as many other enhancements). This requires a CSS (cascading style sheet) file. ⟹

Further details of all these features are discussed in the relevant chapters which follow. *Word* users should also seriously consider using `htm` output and are strongly encouraged to look at that chapter below. The native `docx` output is are only available to **tabout** users who have *Stata* Release 13 or later, so those on earlier Releases need to acquaint themselves with `htm` output if they want to take advantage of all the new formatting features of **tabout**.

I mentioned above that users can now see almost instantly what their final table will look like. This is done with the `open` option which opens their table file in an appropriate application. Which application depends on how their operating system has been set up to 'associate' file extensions with applications. ⟹

See the File association entry in Wikipedia for more information or do an internet search on 'changing file associations' if you need to change the defaults.

For `tex` output there are a couple of preliminary steps needed before you can use the `open` option, and you also need to specify the `body` and `compile` options. These are discussed in Chapter 12 below. ⟹ ⟹

When **tabout** creates tables for the `docx` and `xlsx` styles, the creation process can be slow, though over time *Stata* will probably improve the `Mata` functions used for these styles so speed may get better. Similarly, opening

these files may be slow, particularly the first time they open. If this annoys you, one option is to do the initial construction of your tables in `htm`—irrespective of which final output style you want. Not only is the creation process extremely fast, but viewing these in your browser is instantaneous. When the table reaches finality, you could then switch a couple of settings in your **tabout** code to send the file to the output style you actually need.

11. Delimited text file tables

11.1. Text file basics

Plain text files, also known as ascii files, are amongst the oldest and most reliable files to be found in the computer world. *Stata*'s own `do` files are ascii files, as are LaTeX and `html` files. The only difference with the latter are that they contain mark-up language. The core advantage to all ascii files is that they can be opened in any text editor and then edited. Computer users sometimes open ascii files in their word processors and then wonder why they often have problems later with those files—the reason is that word processors work with *binary* files and a user needs to be careful to save text files as plain text, not in the native format of that word processor. For this reason, novice *Stata* users are always warned to edit their `do` files in either a dedicated text editor, or in the *Stata* `do` file editor, and not in their word processor.

What is a delimited text file? It's basically an ascii file in which the columns of data, that is, the columns in your **tabout** table, are separated by a symbol. This can be a tab symbol, a comma or a semi-colon. In **tabout** you use the following arguments in the `style` option: `tab`, `csv`, or `semi` ⟹ to indicate which symbol should be used. The default option in **tabout** is always tab delimited, so if you don't specify `style`, this is what **tabout** produces.

11.2. Applications and text files

In terms of **tabout**'s output, only the native `xls` files (the older *Excel* files) are truly binary, but the native `xlsx` and `docx` files (the more modern *Excel* and *Word* files) are semi-ascii / semi-binary, in that they are based on `xml` (which is an ascii file) but are enveloped in a zip package (which is binary). In practice, it's best to regard all of the native `xls`, `xlsx` and `docx` files which **tabout** produces as binary. If you try to open any of them in a text editor, you'll see cryptic symbols rather than meaningful text. For this reason, if you use **tabout**'s `open` option—as most of the *User Guide* example ⟹ files do—you'll find that they open in a spreadsheet application like *Excel* or a word processor like *Word*.

On the other hand, it's possible—and often very useful—to open ascii files in applications like *Excel* and *Word*. For example, if you are using any of the delimited file styles and you specify the file name for your table as `table1.txt` with the `open` option also specified, then *Stata* will open your

table in your default text editor. However, if you change the file extension to `xls` or `doc`—and you leave the style unchanged, that is, the default setting of tab delimited—then the `open` option in **tabout** will cause *Stata* to open your file in *Excel* or *Word*—or whatever other spreadsheet application or word processor is associated with these file extensions on your computer—even though the contents of these files is plain text.

Using extensions in this way is a very useful trick to remember. In the case of the `xls` extension, *Excel* not only opens your file, but automatically places the data into the appropriate columns. All that you need to do next is merge the cells in the heading rows (where appropriate), apply some font formatting, and insert some line borders, and your table is largely finished. The appearance of labels may appear misaligned when you open the file, but landing on the cell will confirm that the labels are in the right place.

In the case of the `doc` extension, *Word* will open your file and you can then select the text (except the title and footnote) and use the menu option of *Table / Convert Text* and *Word* will place all your data into the appropriate table cells. As with *Excel*, everything should be aligned properly and you only need to merge cells, apply font formatting and line borders.

Using extensions in this way with what are actually tab delimited plain text files is the reason that this *User Guide* uses the terminology of *native* `xls`, `xlsx` and `docx` outputs. This distinguishes these genuine binary-type outputs from these trick (ascii) ones, that is, where the extension tricks your computer into using the appropriate application, but the contents of the file is plain old ascii text. For this reason, you can still open your 'fake' `xls` or `doc` files with a text editor to inspect them. Finally, a sound practice is for you to reserve these file extensions (`xls` and `doc`) for your 'fake' files and use the `xlsx` and `docx` extensions when you are exporting 'genuine' native `xlsx` and `docx` outputs with the `style` option.

The slowness which I mentioned earlier when it comes to producing `docx` and `xlsx` files is not an issue with `xls` and `doc` files, because these are plain text file outputs. They are produced very quickly, and will also open quite quickly if you have your application already open.

When you open a delimited text file in *Excel*, notice the following. The heading 1 row has the label is the left-most data column, so you only need to select it, and all the cells to the right, and then merge and centre them. The heading 2 row has the labels for its columns in the right-most column of that sub-group. Select it, and the cell to the left, and then merge and centre. Repeat across the heading 2 row. For the final heading, heading row 3, you will probably only need to centre the contents of each cell.

⟹

12. LaTeX tables

12.1. What is LaTeX

When users select the `style(tex)` option the output they get is a plain text file (an ascii file) with various commands embedded in the file. These are a form of mark-up code, which tells a *compiler* how to interpret the file. These days the compiler usually produces a PDF file, which is a high resolution, publication-quality document.

While the TeX system was invented by Donald Knuth in the 1980s, most users of this system employ the LaTeX macro language, invented by Leslie Lamport to make using TeX easier. (Other macro languages for TeX include ConTeXt and LuaTeX). As the LaTeX homepage defines it:

> LaTeX is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation. LaTeX is the de facto standard for the communication and publication of scientific documents. LaTeX is available as free software.

Throughout this *User Guide* I use LaTeX and `tex` interchangeably, with the latter used more often when referring to aspects of the output style and the former when I am talking more generally about these types of tables. Version 3 of **tabout** has aimed to make the LaTeX output easier for novices to use by automating much of the coding, but it remains possible for experienced users who only want 'pure' TeX output to strip out the automatic LaTeX code by specifying `ntc` (*no table code*). ⟹

Indeed, so useful is LaTeX, and so easy to use in **tabout** (after some initial setup steps), that this output option should be considered by anyone who requires high-quality stand-alone PDF files.

12.2. Newcomers to LaTeX

For *Stata users* contemplating leaving their word processors behind and writing their articles and reports with LaTeX, there are a large number of tutorials and other free materials available on the web.

Two books which have been a staple in my library for many years are *The LaTeX Companion*[1] and *A Guide to LaTeX*[2], both of which have been recently updated.

1. **Goossens1994**.
2. **Kopka1999**.

The advantages of LaTeX are numerous:

- unrivalled typesetting quality, far surpassing what word processors can produce;
- it comes as close as possible to providing a 'perfect' reproducible research work-flow—you can produce camera-ready PDF files containing multiple tables from inside *Stata* by running a master.do file;
- the *Stata* manuals, and the *Stata Journal* are both produced using LaTeX and many scientific and econometric journals accept LaTeX manuscripts;
- you can write in a text editor, which is a far more friendly and powerful environment for producing documents—it gives you access to numerous typing shortcuts, regular expression search and replace facilities, and a far more efficient interface based on a keyboard perspective.

The disadvantage of LaTeX are not to be overlooked:

- journals or other publishers often demand *Word* files;
- collaborative work can be difficult with colleagues who don't use LaTeX ;
- the error messages thrown up by the LaTeX compiler can be infuriatingly cryptic—they are often the result of a simple typo but end up wasting your time.

Dealing with these problems is not insurmountable. You can convert a LaTeX document into a `html` document, which can be imported by *Word* and other word processors with nearly all of the formatting preserved. (You can also use PDF-to-Word converters, but the results can be uneven.)

While becoming proficient in LaTeX can take some time, for users of **tabout** the new features of Version 3 make LaTeX documents an easy option. No knowledge of LaTeX is needed to produce a high resolution, publication-quality PDF file of your tables. This can be useful for emailing, placing on a website or printing. Unfortunately, *Word* does not import PDF files, so 'binding' your PDF tables into your *Word* document is not straight forward. But if you have a large number of tables, such as an appendix, you could produce these as PDF files and append them to the final version of your *Word* report (once you have exported it as a PDF file) by using one of the many PDF editors available on the internet.

12.3. Automatic compiling of LaTeX documents

Traditionally most users of **tabout** have incorporated their LaTeX tables into their main LaTeX document using the LaTeX command:
`\input{filename}`
and most will continue to do this. However, one of the new features in Version 3 is the ability to have **tabout** compile your LaTeX output into a PDF file automatically by using the `comp`(*compile*) option. This may be convenient for experienced LaTeX users: you can see instantly what your table will look like and can fine tune settings such as the table width (which tends to be the most fiddly part). For newcomers, it means you don't need to read LaTeX code and learn how to user LaTeX if you don't want to. ⟹

Even if you plan to incorporate your **tabout** tables into a main LaTeX document, the `body` option—which writes to your output file all the LaTeX code needed for a stand-alone document—may be useful. You can copy some of the code **tabout** places in the output file into the preamble of your main file, and then be sure that the tables in your final LaTeX document will match your expectations. ⟹

Here are the steps required for the compile option to work:

- for newcomers, obtain a LaTeX system, called a 'distribution':
  - for Mac and Linux users: TeXLive or MacTeX;
  - for Windows users: MikTeX or TeXLive;
- for both experienced LaTeX users and newcomers create a global macro which tells *Stata* where your LaTeX command files are to be found:
  - locate where the LaTeX command files (Mac and Unix) or LaTeX executable files (Windows) are to be found on your computer.
  - on Mac OS you can type `which pdflatex` in your terminal window to find the location; on Windows you can type `where pdflatex.exe`. (There are actually a number of LaTeX command files, but pdflatex and xelatex are the two which **tabout** uses).
  - typically, on Windows it might be `c:\texlive\2015\bin\win32` or `c:\ProgramFiles(x86)\MikTeX2.9\mixtex\bin` and on Mac it might be `/usr/local/texlive/2015/bin/x86_64-darwin`.
  - at the beginning of the *Stata* do file in which you are writing your **tabout** table code, create a global macro called *tex* thus: `global macro tex PATH_TO_LATEX` where the term `PATH_TO_LATEX` should be replaced by the location you have found. Don't add the executable name (ie. pdflatex.exe) or the final slash in the path.
  - that's all you need to do, and you only do this once. Just make

sure that the line defining your global macro is at the top of all your do files when you're creating tables.

- when you write your table code, use the `style(tex)` option, give your output file the extension `tex` and study some of the example files in the first part of this *User Guide* to see how other options are specified. Finally, add the options `body` and `comp` to your table syntax to enable **tabout** to compile your table into a stand-alone PDF file. You might also consider adding `open` and `show(comp)` to your table syntax.
- what do these options do?
  1. `body` will wrap the extra LaTeX code for your table above and below the usual LaTeX output and turn this into a complete LaTeX document. (This extra code is *not* shown in the examples in this *User Guide*);
  2. `comp` will compile the LaTeX code created in 1. into a PDF file;
  3. `open` will open the PDF file created in 2. in your computer's default PDF viewer;
  4. `show(comp)` will display the results of the compiling. This can assist with debugging your LaTeX output if it fails to compile.

12.4. Enhancements in Version 3

Ease of use

As mentioned earlier, **tabout** has always catered for LaTeX users and has new enhancements to make this approach both easier, and also more flexible. The new `title` and `fn` can be used instead of the earlier `topf` and `botf` options, which is how LaTeX users traditionally added titles and footnotes. While **tabout** always had a `body` option, to include the extra LaTeX code needed to produce a final PDF document, it now has `topbody` and `botbody` so that you can 'chain' multiple tables together. In this way, you can insert more material into your final document, including additional tables and boilerplate text. This was discussed in more detail in Chapter 9 and will be illustrated below.

⟹ ⟹
⟹ ⟹
⟹ ⟹

Another advance for novices is that much more of the LaTeX code needed for creating a table has been automatically generated when the table is produced. Previously, users needed to either write some of this code themselves, or turn on extra options to achieve these results. As mentioned earlier, making LaTeX easier to use hasn't come at the expense of experienced users losing out. If you specify `ntc` (*no table code*), **tabout** will suppress this automatic table code and you can still create your own custom code for inclusion via the `topf` and `botf` options. Experienced users will also be aware

For example, the `bt` option, the `cl1` and `cl2` options have been removed because they are now automatically included. It is now longer necessary to add code into the preamble of your document because this is now automatically included.

⟹

that many common characters or symbols are reserved in LaTeX and should not be used in labels or other text without being 'escaped'. **tabout** deals with some of the most common ones, namely `_ $ \ % & > < +` using a function called *texclean*, so you don't need to worry about escaping them or avoiding them in your labels, titles or footnotes. If you find that some additional symbols are needed, just hack the **tabout** ado file and modify this function to add your own symbols to this list (the function is near the end of the `ado` file).

Inside the automatically generated LaTeX code, as well as when the `body` (or `topbody`) option is chosen, **tabout** makes several assumptions and experienced LaTeX users of **tabout** need to be aware of these:

- the `booktabs` package is loaded and all the lines in the tables make use of the `toprule`, `midrule` and `bottomrule` commands (as well as the `cmidrule` command for partial lines). As a result, the `bt` option is no longer available.
- the `tabularx` package is loaded and all the tables are created as tabularx tables.
- when partial lines are used, such as in the heading rows, **tabout** calculates what columns need spanning. This removes the need for the previous `cl1` and `cl2` options. In addition, there is now a simple `ltrim` option which shortens the beginning of each partial line to get a better layout. This removes the need for the former `cltr1` and `cltr2` options. ⟹
- the paper size is set to *A4*, but this can be changed with the `paper` option. The acceptable choices are: *letter*, *legal* and *A4*. Note that the appropriate LaTeX terminology ("a4paper") is done automatically. ⟹
- the document type is set to *article*, but this can be changed with the `doctype` option. The usual LaTeX choices are available: *article*, *report*, *book* and *letter*. ⟹

If experienced LaTeX users are unhappy with any of these changes, they should find hacking the **tabout** ado file quite easy as all the LaTeX code is clearly laid out and easily changed. Alternatively, the `topf` option is available for complete customisation of the LaTeX document preamble. ⟹

A word of advice to both experienced and novice LaTeX users. The `tabularx` macro is an excellent method for laying out tables, but your first output might be very disappointing so don't be discouraged. The most important variable is the `twidth` (*table width*) option which is set to a default of 14cm. (This default is found in the variable *texdef* in the first part of the ado file, and you might like to change it if you find it does not suit you.) You can, of course, just set the table width for each table by specifying a different `twidth` value. Often you need to vary this, re-running the table code several times until the table looks good. As of Version 3.0.6, the `units` ⟹

option is now available for LaTeX output. This means that you can specify any of the standard LaTeX measures (such as `\columnwidth`) in the `units` option and a numeral such as 1 or .75 in the `twidth` option.

New feature available in Version 3.0.6: support for different measurement units. Thanks to Gilbert Montcho.

⟹ ⟹ ⟹

By using the `body`, `comp` and `open` options, the process can be almost real-time, and it does not take long to fine-tune your tables in this way. A word of warning: not all PDF viewers automatically refresh their contents when you re-compile your LaTeX code, so you might need to close the PDF file in between re-running your **tabout** code. For those planning to incorporate the tables into another LaTeX document using the LaTeX `\input` command, remember to rerun your do file a second time, without `body` option, or your LaTeX compiler will complain loudly (because it encounters multiple `\begin{document}` commands).

Font families

The font option in **tabout** has only ever been used to specify bold or italic for your variable labels. It assumed that users would select a font family in their own LaTeX document. In Version 3, this facility is now built-in to **tabout** and you can specify any Open Type font on your computer using the `family` option. Just enter the name as it appears in your font library (or an application's drop down box), for example, `family(Adobe Garamond Pro)`. If you choose to use this option, **tabout** uses XeLaTeX for compiling the document, since this is required for the `fontspec` package which allows for the use of these Open Type fonts. (Note that LuaLaTeX also supports both `fontspec` and micro-justification, but is much slower.)

⟹

Experienced LaTeX users may prefer to avoid the `family` option because it only works with XeLaTex, which can be slow, and which does not allow full micro-justification of text. Those users will probably just use an `input` command to insert their **tabout** tables into an existing document.

Users of **tabout** who also want `htm`, `docx` and `xlsx` can select font families as well. For LaTeX users there is an additional facility: the ability to select a second family. This is because LaTeX has always worked with two generic families: roman and sans serif. If **tabout** finds two font names in the `family` option it assigns the first one to your roman font and the second to your sans serif font. You need use to use the pipe delimiter inside the `family` option to distinguish between the two family names (since such names usually have spaces in them). You can change the pipe delimiter to another symbol if you prefer (using the `delim` option).

⟹

⟹

Why might you want two families? It might be the case that you want your tables to be in a sans serif font while you also want some boilerplate text (inserted, for example, via the `topf` option) to be a roman font. In this case, add the `ssf` (*sans serif font*) option to your table code, and **tabout** will use the first font specified for the body of the document and the second font specified for the tables in it. If you don't need to mix fonts like this, but you'd still like your tables to be in a sans serif font, just specify a single font in the `family` option and select a sans serif family name. There is no need to specify the `ssf` option: that is only used when two fonts are specified (so that **tabout** knows to use the second one for tables).

⟹

⟹

12.5. Dynamic documents with LaTeX

To draw all of this together in an example, Table 35 repeats the dynamic document shown in Table 34. Note the global macro `tex` at the top of the file and the four key options at the end of the file:

- `body` to have LaTeX document code inserted at the top and bottom;
- `comp` to have *Stata*'s shell command compile the document after **tabout** has written out the LaTeX code;
- `open` to have *Stata*'s shell command open the document in your system PDF viewer; and
- `show(comp)` which displays the results of the compiling. This can assist with debugging your LaTeX output if it fails to compile.

Notice how the font family is specified, with the pipe delimiter and with the `ssf` option to tell **tabout** that you want the table in a sans serif font. The output from this table code is shown on the next page.

This example is for a particular Mac installation. If you're on Windows, Unix (or a different Mac) the path to your LaTeX files will be different.

Copy table 35 code as: tex htm

```
global tex "/usr/local/texlive/2015/bin/x86_64-darwin"
sysuse auto, clear
sum weight
local meanwt = r(mean)
local meanwt: di %3.2f `meanwt'
sum length
local meanlen = r(mean)
local meanlen: di %3.2f `meanlen'
local obs = r(N)

tabout rep78 foreign using table35.tex, replace ///
style(tex) font(italic) c(mean weight) f(0c) sum ///
twidth(9) h1(Car type (mean weight in lbs.)) h3(nil) ///
ssf title(Table 35: Example with font families) ///
fn(auto.dta) topf(topblock.txt) botf(bottomblock.txt) ///
topstr(`meanwt'| `meanlen') botstr(`obs'|$S_DATE) ///
family(Baskerville|Arial) body comp open show(comp)
```

The title of my short report

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

And now we have an important result: **3019.46** lbs. is the average weight of all vehicles. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. And a second important result: **187.93** inches is the average length. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

And now we have the first table.

Table 35: Example with font families

	Car type (mean weight in lbs.)		
	Domestic	*Foreign*	*Total*
Repair Record 1978			
1	3,100		3,100
2	3,354		3,354
3	3,442	2,010	3,299
4	3,532	2,208	2,870
5	1,960	2,403	2,323
Total	3,368	2,263	3,032

auto.dta

And these results were based on **74** observations and were completed on **17 Nov 2016**.

Landscape mode, rotation and lines

Version 2 of **tabout** supported the rotation of value labels, and this has been retained in Version 3. Version 2 also had a `lines` option which has been replaced by two new options: `nohlines` (*no heading lines*) and `noplines` (*no panel lines*). This makes the usage consistent with `noborder` which suppresses the thicker lines at the top and bottom of the table. ⟹ ⟹ ⟹

Version 3 of **tabout** now supports landscape mode, which swaps the page layout from portrait into landscape. Both the `rotate` and `landscape` options require additional code and an additional package in the document preamble, but this is done automatically if you are using the `body` option. If you are incorporating your LaTeX code into an existing document and want these options you will need to add the following lines to your preamble: ⟹ ⟹

```
\usepackage{lscape}
\newcommand{\rot}[2]{\rule{1em}{0pt}%
\makebox[0cm][c]{\rotatebox{#1}{\ #2}}}
```

Table 36 illustrates the use of the `rotate` option, where the argument is the angle you want the labels rotated by. Sometimes the `tabularx` macro disappoints you when your value labels are very long, causing first column to be unduly wide. If there is no way you can shorten the labels, then the landscape option may solve your problem. We shall see this option used in the next example. ⟹

Table 36: Example of rotation and no panel lines

	Not college graduate	College graduate	Total	Not college graduate	College graduate	Total	Not college graduate	College graduate	Total
	No.			%					
Race									
White	1,217	420	1,637	74.3	25.7	100.0	71.0	78.9	72.9
Black	480	103	583	82.3	17.7	100.0	28.0	19.4	26.0
Other	17	9	26	65.4	34.6	100.0	1.0	1.7	1.2
Total	1,714	532	2,246	76.3	23.7	100.0	100.0	100.0	100.0
Location									
Does not live in the South	990	314	1,304	75.9	24.1	100.0	57.8	59.0	58.1
Lives in the South	724	218	942	76.9	23.1	100.0	42.2	41.0	41.9
Total	1,714	532	2,246	76.3	23.7	100.0	100.0	100.0	100.0

nlsw88.dta

Copy table 36 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout race south collgrad using table36.tex, replace ///
style(tex) font(bold) c(freq row col) f(0c 1 1) ///
layout(cb) h1(nil) h3(No. %) h3c(3 6) ltrim(-0.5) ///
noplines rotate(60) title(Table 36: Example of ///
rotation and no panel lines) fn(nlsw88.dta)
```

Several other interesting features in this **tabout** code include suppressing the h1 row heading, because it 'spoils' the rotation effect. The h3 row heading can also be suppressed for an even more pleasing effect (see Table 37 for this). In Table 36 the `h3c` option has been used to avoid the repetition of three 'No.' labels and six '%' labels. This table also illustrates the use of the `noplines` option to remove the panel lines. Because of the rotation, the line separating h1 and h2 looks a bit too short, so the `trim` option has been used. But note that it has used a negative measurement, which *increases* the length of the line. You might like to remove the `ltrim` option and compare the difference.

Table 37, on the next page, repeats Table 36 but makes the table wider to accommodate the data and uses the `land` (*landscape*) option. In LaTeX only the table is placed in landscape mode, the page remains in portrait mode. (This differs from the situation in *Word* where the page itself is placed in landscape mode.) Finally, Table 37 also changes the rotation angle to 75 degrees, so that the labels are more upright.

Copy table 37 code as: tex

```
do nlsw_data_setup
tabout race south collgrad using table37.tex, replace ///
style(tex) font(bold) c(freq row col) f(0c 1 1) ///
layout(cb) h1(nil) h3(nil) noplines twidth(18) ///
land rotate(75) title(Table 37: Example of rotation, ///
no panel lines and landscape mode) fn(nlsw88.dta)
```

Table captions

One of the great strengths of LaTeX is its extensive cross-referencing system. You can place a `\label{name}` command next to your table and then reference that label anywhere in your text, and the two will correspond and be automatically numbered. If you move that table somewhere else, the cross-referencing will automatically update when the document is compiled. For example, Table 3.1 might become Table 4.2 if moved to another chapter, but all of the references in the text will still be correct. Experienced LaTeX users will have used the `topf` option in the past to write their table code and probably include a caption facility in their document. These captions, by the way, belong inside what LaTeX calls a table environment.

Table 37: Example of rotation, no panel lines and landscape mode

	Not college graduate	*College graduate*	*Total*	*Not college graduate*	*College graduate*	*Total*	*Not college graduate*	*College graduate*	*Total*
Race									
White	1,217	420	1,637	74.3	25.7	100.0	71.0	78.9	72.9
Black	480	103	583	82.3	17.7	100.0	28.0	19.4	26.0
Other	17	9	26	65.4	34.6	100.0	1.0	1.7	1.2
Total	1,714	532	2,246	76.3	23.7	100.0	100.0	100.0	100.0
Location									
Does not live in the South	990	314	1,304	75.9	24.1	100.0	57.8	59.0	58.1
Lives in the South	724	218	942	76.9	23.1	100.0	42.2	41.0	41.9
Total	1,714	532	2,246	76.3	23.7	100.0	100.0	100.0	100.0

nlsw88.dta

In Version 3 of **tabout** the `caplab` and `cappos` options write all of the LaTeX code that is needed above and below your table. The actual caption is taken from the `title` option, so you still need to provide that. What `caplab` does is set the `\label` option inside the caption, which is how LaTeX references that table. It is usually wise to give it a meaningful name, one not related to the table numbering, because these can often change. The `cappos` option tells **tabout** whether you want the caption above or below the table. If the former, you don't need to use this option (it is the default) but if you want it below the table then specify `cappos(below)`. Table 38 (shown as Table 12.1) provides an example of this.

	Location					
	Does not live in the South		Lives in the South		Total	
	No.	%	No.	%	No.	%
Race						
White	1,071	65.4	566	34.6	1,637	100.0
Black	210	36.0	373	64.0	583	100.0
Other	23	88.5	3	11.5	26	100.0
Total	1,304	58.1	942	41.9	2,246	100.0

nlsw88.dta

Table 12.1.: Example of caption below table

Copy table 38 code as: txt tex docx xlsx htm

```
do nlsw_data_setup
tabout race south using table38.tex, replace ///
style(tex) font(bold) c(freq row) f(0c 1) ///
doctype(report) caplab(racebysouth) cappos(below) ///
title(Example of caption below table) fn(nlsw88.dta)
```

There are several important points to notice in the code for Table 38:

- the table's title was given without the word 'Table' and without a number (compare this with all the other example tables);
- LaTeX itself assigned the numbering, namely 12.1 (table 1 in the 12th chapter), and provided the word 'Table';
- the `caplab` option is given as a meaningful phrase;
- the LaTeX `doctype` is used to define this document as a *`report`* (the default is *`article`*).

In this *User Guide* the caption alignment, the font and the font size of the caption do not match the table. This is because the caption settings are taken from the preamble in this *User Guide* (look at the caption for the Figure on page 9 above to see the style used in this *Guide*). However, if you use the code shown above, and add the `body`, `comp` and `open` options, you will notice that everything about the caption matches your table.

The customisation of the caption—apart from the position—is not done by **tabout**. Some code for dealing with this is added to the preamble of your document if you select the `body` option (this includes the LaTeX packages `caption` and `float`) . If you are using `input` in your LaTeX document to bring in your tables, you will need to add these two packages to your document preamble. You might like to study the code **tabout** writes when using the `body` option and adapt it to your needs. You can personally redefine your caption in numerous ways or you can use one of several LaTeX packages which are dedicated to greater control over captions. This is particularly important for users of languages other than English who may wish to change the word 'Table' into something more appropriate, or for users who may wish to change the font, font size or alignment.

Newcomers to LaTeX might wonder if captions are really worth the effort. You might be content to just use the `title` option by itself and 'hard code' all your tables. However, you will soon find this tedious when one table in a large batch is dropped, and all the rest have to be renumbered! Furthermore, not using captions means you lose the great advantage of cross-referencing. For example, if you look at the LaTeX code found in the output file `table38.tex` you'll see the phrase: `\label{tab:racebysouth}`. Using this phrase all you need do is issue a LaTeX command such as:

```
as Table \ref{tab:racebysouth} shows ...
```

and you will get the following: 'as Table 12.1 shows'. You will also get a hyperlink (if you wish) with this approach, which can be very useful in navigating your document.

Hyperlinks require the `hyperref` package to be in your LaTeX document preamble. This is *not* done by the `body` option in **tabout**.

13. docx tables

13.1. Introduction: using tabout with Word

In the beginning **tabout** began life with LaTeX in mind but it is obvious that many *Stata* users rely on *Word* when it comes to writing articles, reports or other documents which use tables. There are at least four routes for getting those tables into *Word* which are supported by **tabout**:

1. export from **tabout** using tab delimited output, and open the file in *Word* and then use *Table - Convert - Convert Text to Table*.

   Advantages:

   - fast and largely trouble-free method

   Disadvantages:

   - tables need formatting in *Word*, that is, lines and fonts etc need setting. It is thus unsuitable for dynamic updating of table or for efficient workflows.

2. export from **tabout** using `htm` output to create a `html` file and then open this in *Word*.

   Advantages:

   - fast and preserves nearly all table formatting. Produces an actual *Word* table in the document.

   Disadvantages:

   - user needs to save document as *Word* document if opened as a `html` document, but not if inserted. (See Chapter 15 for more details);
   - getting table width correct can be fiddly.

3. export from **tabout** using `xlsx` output, open the file in *Excel* and then copy and paste into *Word*.

   Advantages:

   - produces highest quality table possible in *Word* (almost comparable to a LaTeX table) because the `xlsx` output style has more aesthetic options than the other styles.

   Disdvantagess:

   - requires *Stata* Release 13 or later;
   - requires a copy-and-paste step, but dynamic updating of tables remains possible (see discussion in Chapter 14).

4. export from **tabout** using `docx` output and open the file in *Word*.

   Advantages:

   - begins life as a `docx` file with no conversions of any kind needed. Table is an *Word* table. Dynamic updating of tables is easy.
   - 'Autofit to Contents' saves time with fiddling to get width correct.
   - placing tables in landscape mode is possible.

   Disadvantages:

   - requires *Stata* Release 13 or later.
   - can be slow.
   - aesthetically still limited compared to other output styles (but may improve over time).

This chapter deals only with the last route, the `docx` output style. If using using *Word* with any of the other routes, consult the chapter dealing with that style.

13.2. *Genuine* Word *files*

Release 13 brought genuine *Word* files to the world of *Stata*, that is, 'native' `docx` files. All of the earlier discussion on using *Word* with *Stata* in this User Guide entailed 'fooling' *Word* into treating other output styles in an appropriate way. As I showed earlier, you could use the *Word* menu to convert tab delimited files into *Word* tables, or you could open a `html` file and then save it as a *Word* document. Both of these worked fine, particularly the `html` route, but actually getting *Stata* to export genuine `docx` files is an innovation, and in coming years *Stata* may expand the capabilities of this format even further. At present, there are some limitations, which I discuss below.

Before going further it's worth noting the difference between doc files and docx files. Both are binary files, which means that opening them in text editors will just show you a garbled screen. The doc format was used by *Microsoft* from 1997 to 2004, and the docx format has been the *Microsoft* standard since then. Most other non-Microsoft word processors also allow you to open and save in this format. The Mata exporting function used by **tabout** *only* produces docx files, which contrasts with the Mata function for spreadsheets, which can produce both xls and xlsx files. I'll say more about this in the Chapter 14.

As I mentioned earlier docx contains within it an xml text file, but it is best regarded as binary.

In summary, if you want genuine *Word* files, you use a docx extension on your file name and you select **tabout**'s docx style output. The discussion on html files in Chapter 15 of this *User Guide* suggests using a doc extension with your html files, but mixing this with htm for your output style. You always had to make sure you saved your file as a *Word* document, because *Word* still regarded it as html until it was saved as a *Word* document. This trick was just a convenience to get your operating system to open the file in *Word*. The same trick was used with tab delimited files, which were given an xls extension, to open them automatically in *Excel*. It is important to keep in mind that the docx and xlsx output styles have nothing to do with these tricks, but are new features of **tabout**. However, because they rely on *Stata* Release 13 (or later), you can fall back on these tricks if you use an earlier Release of *Stata* and still require either *Word* or *Excel* tables.

13.3. Pros and cons

Which route should you go down: genuine or trick *Word* files? The choice is up to you (though my own personal preference is for html). To help you decide, here are some of the pros and cons of using the native docx format compared with using html.

Positive features:

- Landscape mode is available using the native docx output but not with html.
- The 'Autofit to Contents' width setting is available with docx, and this works well most of the time, so you don't need to spend time fine tuning the width, as can happen with html.
- *Word* knows the file is a docx file from the beginning and you don't need to go down the *File - Save As* route to ensure it becomes a docx file.

Negative features

- The docx output style can be very slow. As the *Stata* manual warns: "Creating a new document in a new session of Stata can cause some noticeable delay, usually several seconds." Fortunately, if *Word* is left open, creating further documents can be faster.

- The document is not automatically refreshed. If you forget to close a *Word* document and rerun your modified **tabout** code, it may appear nothing has changed. You need to close the open document, then run the **tabout** code. This can be tedious when you are developing a table with multiple runs.
- You cannot customise headings using format coding, though you can still customise the *content* of the headings.
- You cannot make use of **tabout**'s `topf` or `botf` options, which rules out adding additional text above and below your tables. In this respect, you cannot build the kind of dynamic documents discussed earlier in this guide.
- You cannot append multiple tables into a single document, as you can with all the other output styles.

Some of these shortcomings may be overcome in the future as *Stata* expands the capabilities of the Mata function which is used by **tabout** for creating *docx* files.

The lack of coding facilities with both *docx* and *xlsx* output styles does not mean that the `h1`, `h2` and `h3` options are not available. On the contrary, they are just as useful for these outputs as they are for the other styles. You can still customise the content of all your table headings using these options. Compared with the flexibility inherent in the *tex* and *htm* output styles, with these binary outputs you can't embed any kind of coding to control alignment or spanning of columns. This is not a problem, however, because you can make use of the 'partner' heading column options, that is, `h1c`, `h2c` and `h3c` which tell **tabout** how you want the headings to line up with columns.

You will notice that in all the example files for both the *docx* and *htm* styles that there is always a column heading option accompanying each heading option, even when no spanning of columns is involved. This can lead to strange options like: `h3c(1 1 1)` where no spanning is needed. In situations like this, you would probably use the `clab` option, but **tabout** is designed so that all three heading rows can be customised, and all three can use spanning, or single column headings, depending on your needs. For example, you may even have a table made up of a set of single column data (that is, no spanning) but where all three heading rows are needed, and this would mean that the `(1 1 1)` sequence would have to be repeated for each of them. While somewhat unusual, a situation like this could arise with summary tables. What is more common, particularly in twoway basic tables, is for the first heading row to be a grouping label, and hence spanning several columns.

13.4. Landscape, width and paper size

Landscape mode

As mentioned above, you can use the `landscape` option in **tabout** and you will get a table which opens in *Word* in landscape mode. ⟹

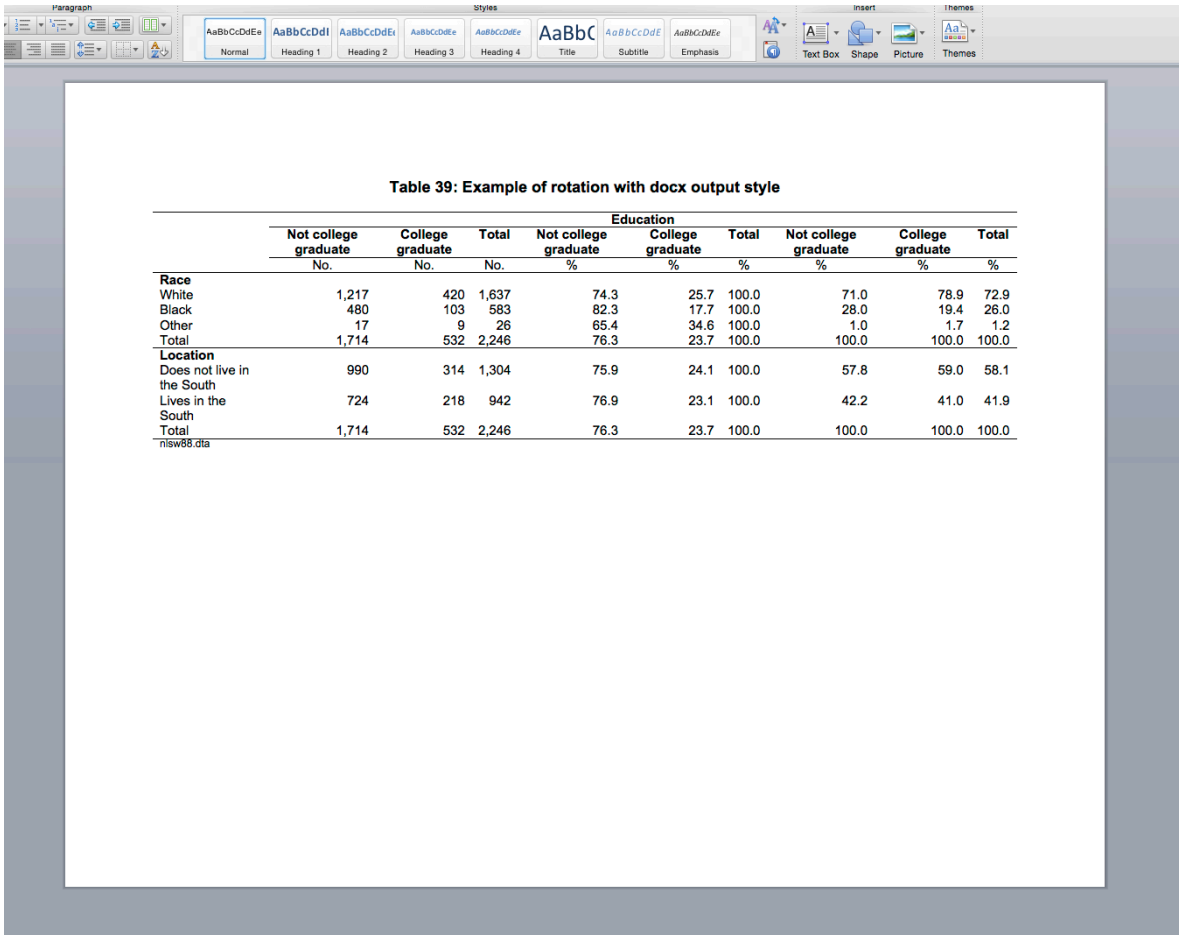


Table 39: Example of rotation with docx output style

	Education								
	Not college graduate	College graduate	Total	Not college graduate	College graduate	Total	Not college graduate	College graduate	Total
	No.	No.	No.	%	%	%	%	%	%
Race									
White	1,217	420	1,637	74.3	25.7	100.0	71.0	78.9	72.9
Black	480	103	583	82.3	17.7	100.0	28.0	19.4	26.0
Other	17	9	26	65.4	34.6	100.0	1.0	1.7	1.2
Total	1,714	532	2,246	76.3	23.7	100.0	100.0	100.0	100.0
Location									
Does not live in the South	990	314	1,304	75.9	24.1	100.0	57.8	59.0	58.1
Lives in the South	724	218	942	76.9	23.1	100.0	42.2	41.0	41.9
Total	1,714	532	2,246	76.3	23.7	100.0	100.0	100.0	100.0

nlsw88.dta

Copy table 39 code as: docx

```
do nlsw_data_setup
tabout race south collgrad using table39.docx, ///
replace style(docx) font(bold) c(freq row col) ///
f(0c 1 1) layout(cb) landscape  title(Table 39: ///
Example of rotation  with docx output style) ///
fn(nlsw88.dta) open
```

Table width

I mentioned earlier that *Word*'s 'Autofit to Contents' feature works reasonably well, which is why most of the `docx` example files don't have any `twidth` options. This option is, nevertheless, available for `docx` output styles if required (though note that the `cwdith` and `lwidth` options aren't available with the `docx` output style). ⟹

You will notice in the screenshot for Table 39 that the cell data does not line up well with the heading labels, largely because the table is now so wide in landscape mode. The top screenshot on the next page shows the result of adding the `twidth(50)` option to the the code for Table 39.

While this does improve the alignment of the *data* columns, the first column with labels is now far too narrow. You can either try to fine tune your results by varying the value of the `lwidth` option, or you can make use of **tabout**'s `hright` option. This is a rarely used feature, and has largely been designed for problems like this. It applies only to heading rows 2 and 3 and only works with the `docx` output style. If you add `hright` to the code in `table39c.do` and re-run it, you will see that it changes the default alignment of these two rows from centre to right, and largely fixes this problem. The bottom screenshot on the next page shows the result of this change. ⟹ ⟹

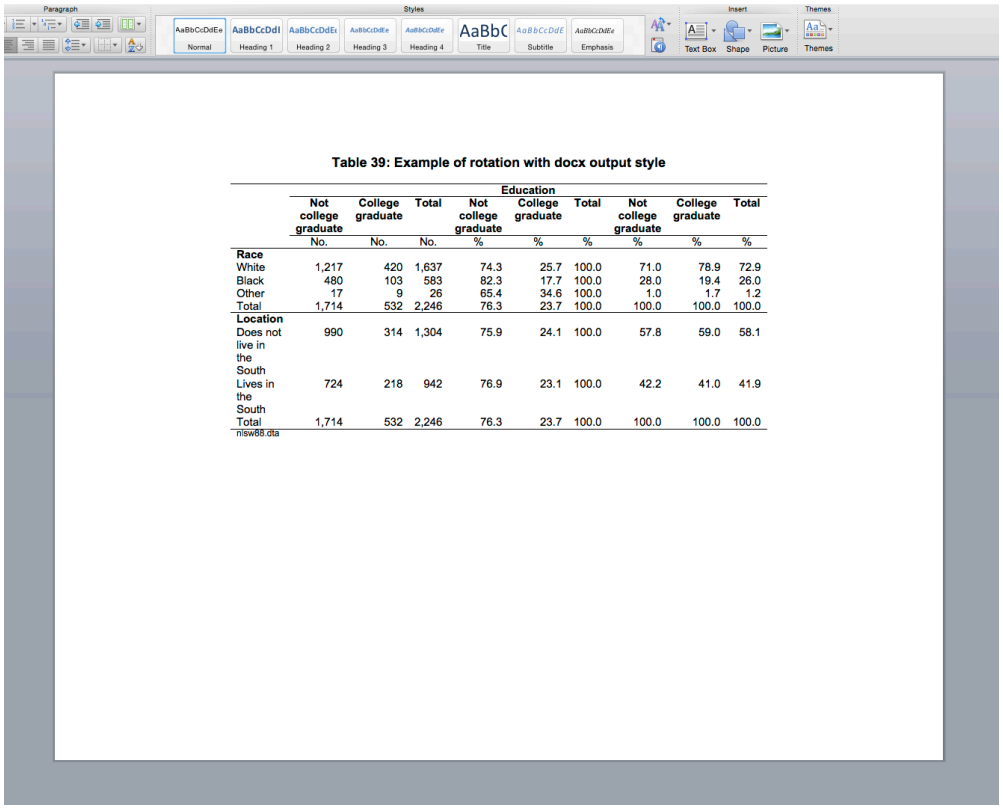


Table 39: Example of rotation with docx output style

	Education								
	Not college graduate	College graduate	Total	Not college graduate	College graduate	Total	Not college graduate	College graduate	Total
	No.	No.	No.	%	%	%	%	%	%
Race									
White	1,217	420	1,637	74.3	25.7	100.0	71.0	78.9	72.9
Black	480	103	583	82.3	17.7	100.0	28.0	19.4	26.0
Other	17	9	26	65.4	34.6	100.0	1.0	1.7	1.2
Total	1,714	532	2,246	76.3	23.7	100.0	100.0	100.0	100.0
Location									
Does not live in the South	990	314	1,304	75.9	24.1	100.0	57.8	59.0	58.1
Lives in the South	724	218	942	76.9	23.1	100.0	42.2	41.0	41.9
Total	1,714	532	2,246	76.3	23.7	100.0	100.0	100.0	100.0

nlsw88.dta

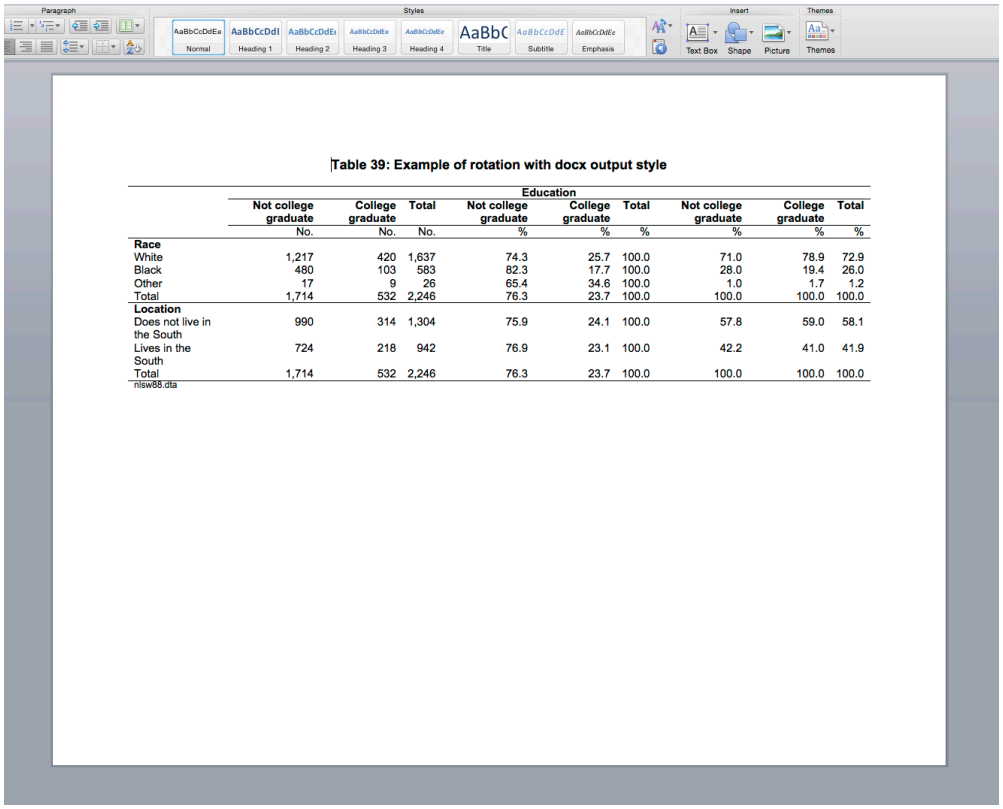


Table 39: Example of rotation with docx output style

	Education								
	Not college graduate	College graduate	Total	Not college graduate	College graduate	Total	Not college graduate	College graduate	Total
	No.	No.	No.	%	%	%	%	%	%
Race									
White	1,217	420	1,637	74.3	25.7	100.0	71.0	78.9	72.9
Black	480	103	583	82.3	17.7	100.0	28.0	19.4	26.0
Other	17	9	26	65.4	34.6	100.0	1.0	1.7	1.2
Total	1,714	532	2,246	76.3	23.7	100.0	100.0	100.0	100.0
Location									
Does not live in the South	990	314	1,304	75.9	24.1	100.0	57.8	59.0	58.1
Lives in the South	724	218	942	76.9	23.1	100.0	42.2	41.0	41.9
Total	1,714	532	2,246	76.3	23.7	100.0	100.0	100.0	100.0

nlsw88.dta

Paper size

Because **tabout**'s `docx` output style produces an actual *Word* document, paper size matters. To be even handed, the default for the `docx` style is `letter` (since the default size for the `tex` style is `A4`). Users who prefer a different choice can either specify `paper(A4)` in their **tabout** code, or hack the ado file and change the default. The following values are available for the `paper` option: `letter`, `legal`, `A3`, `A4`, and `B4JIS`. ⟹

13.5. Reproducible research with docx

I mentioned earlier that the kind of dynamic documents which were discussed in the LaTeX and `html` chapters are not possible with the `docx` style because it does not support the `topf` and `botf` options. One can, nevertheless, conduct reproducible research very successfully with *Word*. ⟹ ⟹

The following scenario is assumed: you are writing a report or article, for example, and this is done inside *Word* using all its formatting features, and it bibliography, cross-referencing, footnoting features and so forth (depending on your needs).

Each time that you need to insert a *Stata* table, you run your **tabout** code to create the table (or batch of tables) and then you select the *File - Insert* menu item to place your table or tables at an appropriate place. You make sure that you tick the *Insert File* dialogue box: *Link to File* (see screenshot on the page after next). You then keep working on your *Word* document as you see fit. Then when something changes in your data—perhaps the weighting variable, the coding of the categories or the number of observations—you go back to *Stata*, make the changes and then re-run your **tabout** code. At this point, all the tables on disk are updated, but your *Word* document is still the same. So you go into the *Edit - Links* menu item and click on the *Update Now* button inside the *Links* dialogue box. This will update all your inserted files, and you *Word* document will then reflect all your changes.

Keep in mind that this strategy relies on all of the formatting of a table being done in **tabout**. If you need to manually reformat a table inside *Word* you will loose those format changes when you update the file. Fortunately, the `docx` tables which **tabout** now produces are generally publication quality and while they will never match the standard achieved by LaTeX tables, they should not need manually formatting very often.

The following screenshots show the sequence. The first shows your *Word* document, in which part of your report is already written and you begin to insert a table. Note that the *Link to File* option is ticked. This is tedious, and is easily overlooked. It would be much better if there were a default setting which could be left marked for all your insertions (perhaps

there is, but I don't know about it.) This link feature does not appear to be available in the *File Insert* commands in other word processors like *Libre Office* and *Open Office*.

The second screenshot shows what your document looked before updating the tables. The third screenshot shows the update dialogue box (accessed through the *Edit - Links* menu) and the fourth screenshot shows the results of updating the tables. Not being a *Word* user, I rarely use these features, but there is an automatic updating option (in the *Word* preferences) which will update all links when the file is opened.

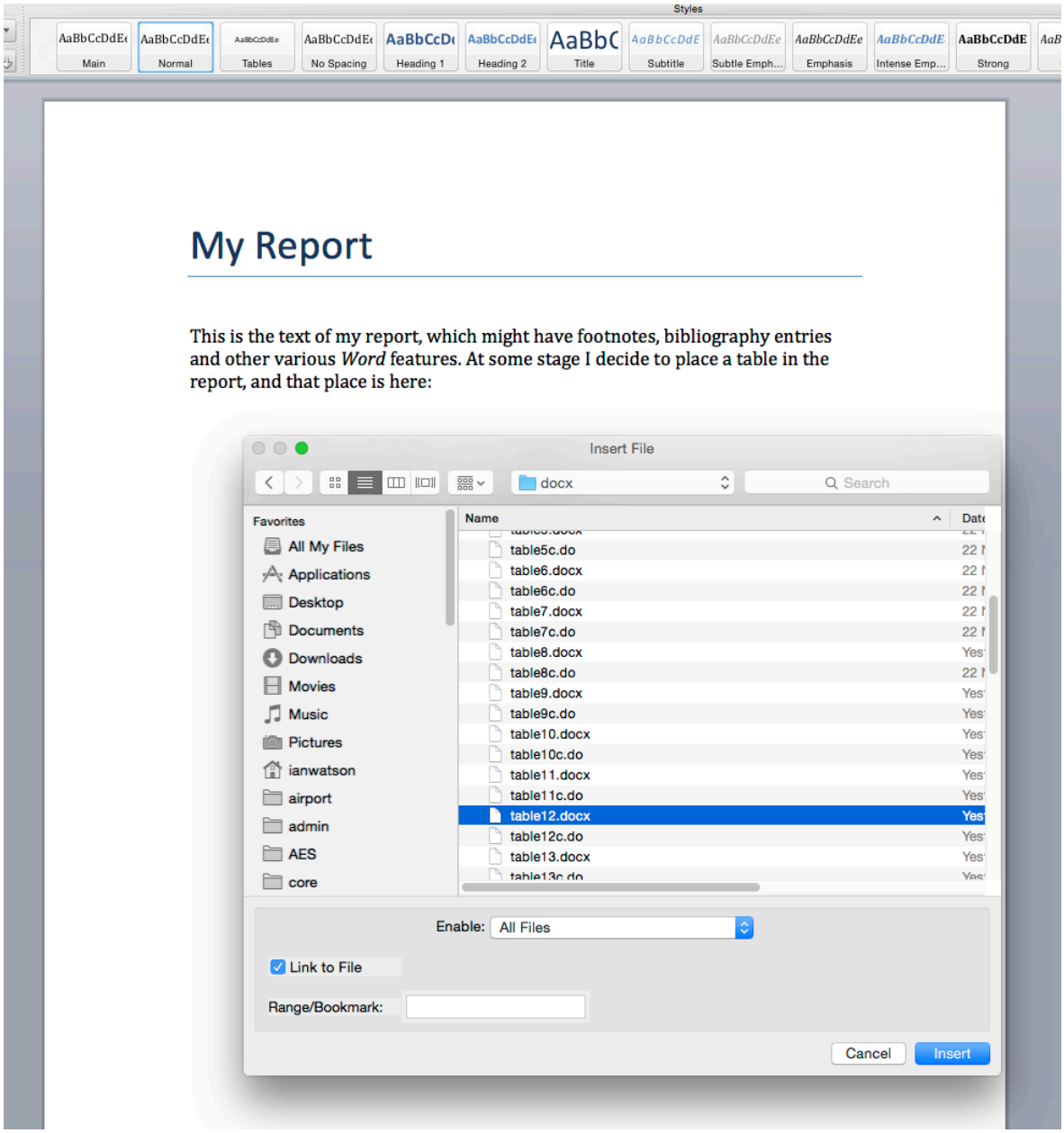
To illustrate these before and after effects I have simply changed the formatting of the data cells and expressed the proportions as percentages but in a real world situation the data itself would probably change. Notice that the width of the second table changed substantially. This was done automatically and is due to the default setting for `docx` tables ('Autofit to Contents'). If you wished the table to remain a constant width after updating its contents, you could set the `twidth` option to a fixed value. ⟹

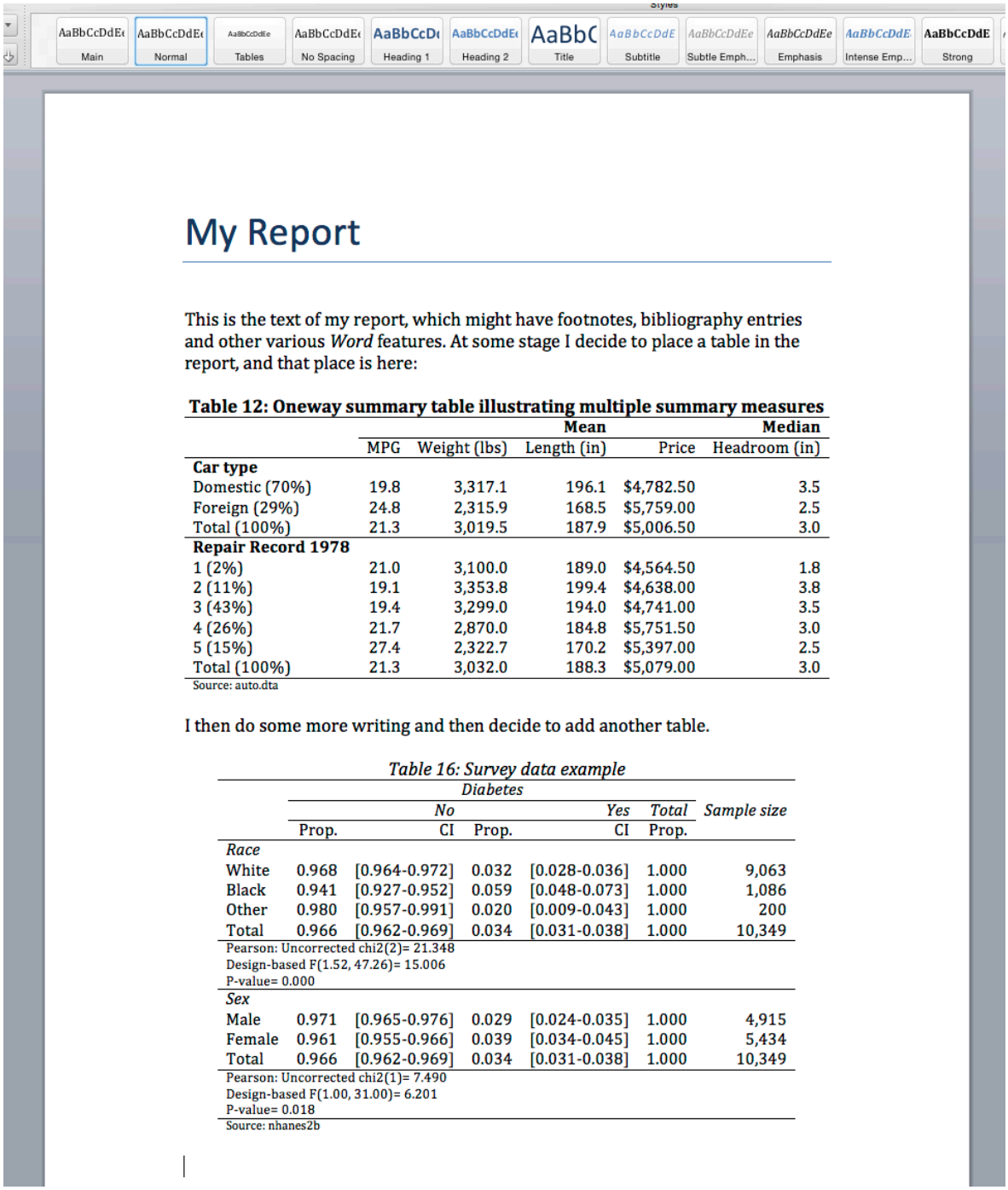
This workflow does ensure that the tables in your document are always 'up to date' but it obviously does not update any references in the the body of your text to the data in your tables. This feature is illustrated in Chapters 12 and 15 (for `tex` and `htm` output) but finding other ways to incorporate this into your *Word* document is beyond the scope of this *User Guide*. Other *Stata* resources (such as Bill Rising's presentation) may provide insights.

Bill Rising "Dynamic documents in Stata: Many routes to the same goal". Download presentation.

Note that if you don't specify a font family or font size in your **tabout** code, when you open your `docx` file inside *Word* it will take on *Word*'s default font family and font size.

Note that what *Word* calls 'font', **tabout** calls 'font family', and what *Word* calls 'font style', **tabout** calls 'font'.


Styles
Main
Normal
Tables
No Spacing
Heading 1
Heading 2
Title
Subtitle
Subtle Emph...
Emphasis
Intense Emp...
Strong
My Report
This is the text of my report, which might have footnotes, bibliography entries and other various Word features. At some stage I decide to place a table in the report, and that place is here:
Insert File
docx
Search
Favorites
All My Files
Applications
Desktop
Documents
Downloads
Movies
Music
Pictures
ianwatson
airport
admin
AES
core
Name
table5c.do
table6.docx
table6c.do
table7.docx
table7c.do
table8.docx
table8c.do
table9.docx
table9c.do
table10.docx
table10c.do
table11.docx
table11c.do
table12.docx
table12c.do
table13.docx
Enable:
All Files
Link to File
Range/Bookmark:
Cancel
Insert



My Report

This is the text of my report, which might have footnotes, bibliography entries and other various *Word* features. At some stage I decide to place a table in the report, and that place is here:

Table 12: Oneway summary table illustrating multiple summary measures

			Mean		Median
	MPG	Weight (lbs)	Length (in)	Price	Headroom (in)
Car type					
Domestic (70%)	19.8	3,317.1	196.1	$4,782.50	3.5
Foreign (29%)	24.8	2,315.9	168.5	$5,759.00	2.5
Total (100%)	21.3	3,019.5	187.9	$5,006.50	3.0
Repair Record 1978					
1 (2%)	21.0	3,100.0	189.0	$4,564.50	1.8
2 (11%)	19.1	3,353.8	199.4	$4,638.00	3.8
3 (43%)	19.4	3,299.0	194.0	$4,741.00	3.5
4 (26%)	21.7	2,870.0	184.8	$5,751.50	3.0
5 (15%)	27.4	2,322.7	170.2	$5,397.00	2.5
Total (100%)	21.3	3,032.0	188.3	$5,079.00	3.0

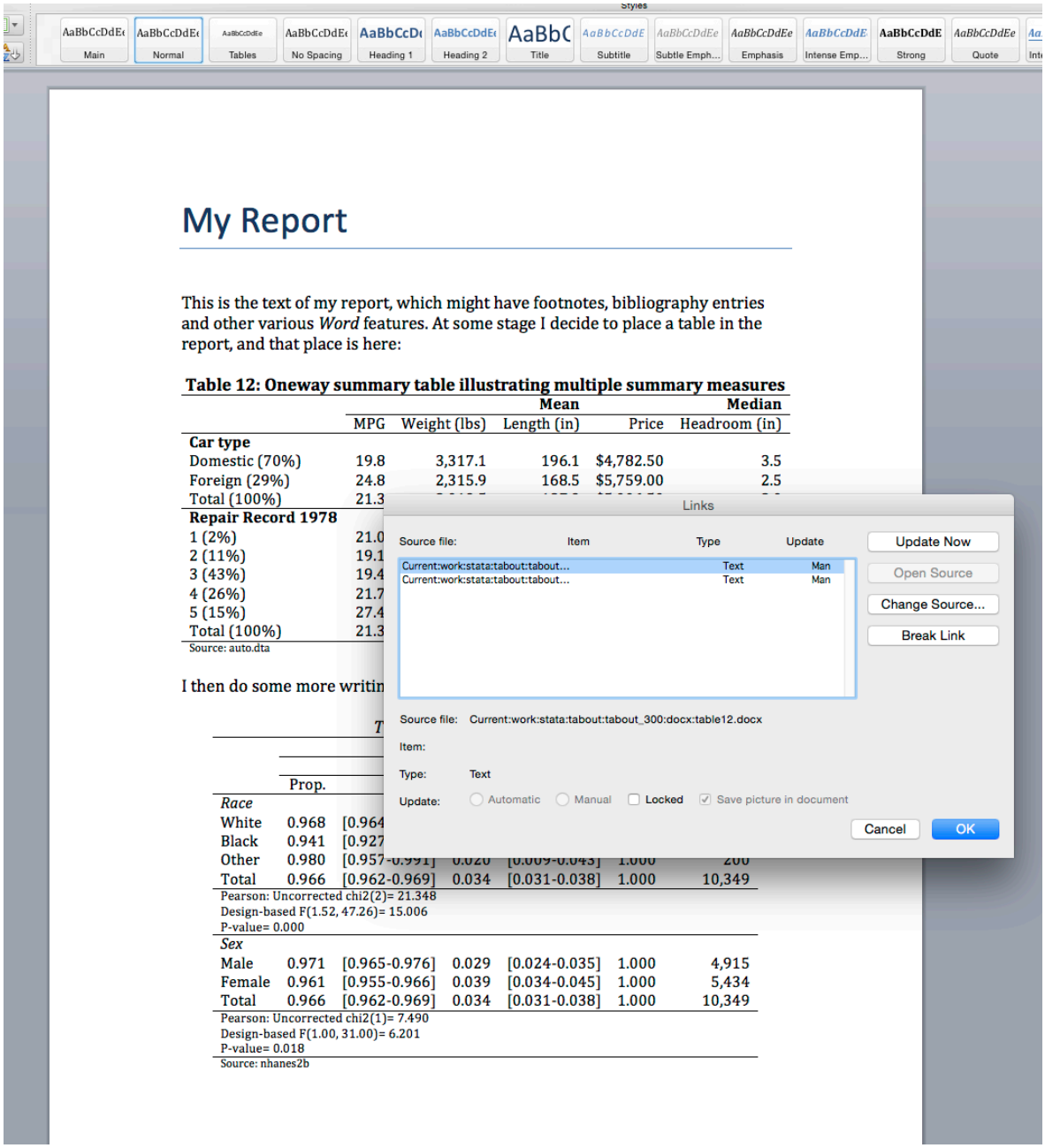
Source: auto.dta

I then do some more writing and then decide to add another table.

Table 16: Survey data example

	Diabetes					
	No		*Yes*		*Total*	*Sample size*
	Prop.	CI	Prop.	CI	Prop.	
Race						
White	0.968	[0.964-0.972]	0.032	[0.028-0.036]	1.000	9,063
Black	0.941	[0.927-0.952]	0.059	[0.048-0.073]	1.000	1,086
Other	0.980	[0.957-0.991]	0.020	[0.009-0.043]	1.000	200
Total	0.966	[0.962-0.969]	0.034	[0.031-0.038]	1.000	10,349
Pearson: Uncorrected chi2(2)= 21.348						
Design-based F(1.52, 47.26)= 15.006						
P-value= 0.000						
Sex						
Male	0.971	[0.965-0.976]	0.029	[0.024-0.035]	1.000	4,915
Female	0.961	[0.955-0.966]	0.039	[0.034-0.045]	1.000	5,434
Total	0.966	[0.962-0.969]	0.034	[0.031-0.038]	1.000	10,349
Pearson: Uncorrected chi2(1)= 7.490						
Design-based F(1.00, 31.00)= 6.201						
P-value= 0.018						

Source: nhanes2b



My Report

This is the text of my report, which might have footnotes, bibliography entries and other various *Word* features. At some stage I decide to place a table in the report, and that place is here:

Table 12: Oneway summary table illustrating multiple summary measures

	Mean			Median
	MPG	Weight (lbs)	Length (in)	Price	Headroom (in)
Car type					
Domestic (70%)	19.8	3,317.1	196.1	$4,782.50	3.5
Foreign (29%)	24.8	2,315.9	168.5	$5,759.00	2.5
Total (100%)	21.3				
Repair Record 1978					
1 (2%)	21.0				
2 (11%)	19.1				
3 (43%)	19.4				
4 (26%)	21.7				
5 (15%)	27.4				
Total (100%)	21.3				

Source: auto.dta

I then do some more writin

	Prop.					
Race						
White	0.968	[0.964				
Black	0.941	[0.927				
Other	0.980	[0.957-0.991]	0.020	[0.009-0.043]	1.000	200
Total	0.966	[0.962-0.969]	0.034	[0.031-0.038]	1.000	10,349
Pearson: Uncorrected chi2(2)= 21.348						
Design-based F(1.52, 47.26)= 15.006						
P-value= 0.000						
Sex						
Male	0.971	[0.965-0.976]	0.029	[0.024-0.035]	1.000	4,915
Female	0.961	[0.955-0.966]	0.039	[0.034-0.045]	1.000	5,434
Total	0.966	[0.962-0.969]	0.034	[0.031-0.038]	1.000	10,349
Pearson: Uncorrected chi2(1)= 7.490						
Design-based F(1.00, 31.00)= 6.201						
P-value= 0.018						

Source: nhanes2b

My Report

This is the text of my report, which might have footnotes, bibliography entries and other various *Word* features. At some stage I decide to place a table in the report, and that place is here:

Table 12: Oneway summary table illustrating multiple summary measures

	Mean				Median
	MPG	Weight (lbs)	Length (in)	Price	Headroom (in)
Car type					
Domestic (70%)	20	3317	196	4782	4
Foreign (29%)	25	2316	169	5759	2
Total (100%)	21	3019	188	5006	3
Repair Record 1978					
1 (2%)	21	3100	189	4564	2
2 (11%)	19	3354	199	4638	4
3 (43%)	19	3299	194	4741	4
4 (26%)	22	2870	185	5752	3
5 (15%)	27	2323	170	5397	2
Total (100%)	21	3032	188	5079	3

Source: auto.dta

I then do some more writing and then decide to add another table.

Table 16: Survey data example

	Diabetes					
	No		*Yes*		*Total*	*Sample size*
	%	CI	%	CI	%	
Race						
White	96.8	[96.4-97.2]	3.2	[2.8-3.6]	100.0	9,063
Black	94.1	[92.7-95.2]	5.9	[4.8-7.3]	100.0	1,086
Other	98.0	[95.7-99.1]	2.0	[0.9-4.3]	100.0	200
Total	96.6	[96.2-96.9]	3.4	[3.1-3.8]	100.0	10,349
Pearson: Uncorrected chi2(2)= 21.348						
Design-based F(1.52, 47.26)= 15.006						
P-value= 0.000						
Sex						
Male	97.1	[96.5-97.6]	2.9	[2.4-3.5]	100.0	4,915
Female	96.1	[95.5-96.6]	3.9	[3.4-4.5]	100.0	5,434
Total	96.6	[96.2-96.9]	3.4	[3.1-3.8]	100.0	10,349
Pearson: Uncorrected chi2(1)= 7.490						
Design-based F(1.00, 31.00)= 6.201						
P-value= 0.018						

Source: nhanes2b

14. xlsx tables

14.1. Introduction: genuine Excel files

As I mentioned in the last chapter, Release 13 of *Stata* made available to *Stata* users genuine, native `xlsx` files. The main differences between *Stata*'s *Word* capability and *Stata*'s *Excel* capability are:

- *Word* files are only available in the `docx` format.
- by contrast, *Excel* files are available in as both `xls` and `xlsx` files. The former are for *Excel* 1997/2003 files and the latter for *Excel* 2007/2013 files. `xlsx` is the current defacto *Excel* standard and most other non-Microsoft spreadsheet applications also allow you to open and save files in this format.
- *Word* files rely on the `Mata _docx*()` functions, which are used by **tabout** (and by other *Stata* programmers) to produce output files.
- in the case of *Excel*, there are two options: there is the `Mata` class `xl()` which is used by **tabout** (and by other programmers) and there is also a *Stata* command, `putexcel` which allows non-programming *Stata* users to export their results to *Excel* files.

Most users of *Stata* are 'programmers' if they write do files, while many users will utilise Mata functions in *Stata*'s interactive mode (or in their do files). My comments here refer to 'programmers' who produce ado files and rely on Mata to do all the heavy lifting.

All of this may sound complicated. What really matters is understanding that `xlsx` output from *Stata* is more advanced at this stage, and offers more flexibility to *Stata* users. Over time, however, it is likely that `docx` output will improve as *Stata Corporation* improves the underlying `Mata` functions.

When it comes to **tabout** the advice is simple: if you have Release 13 of *Stata* or later, and you want publication quality tables with no (or minimal) extra formatting on your part, use either the `docx` or `xlsx` output styles. If you find these too slow, or inappropriate for another reason, consider the tab delimited output style for *Excel* files and the `html` output style for *Word* files. If you follow this route, I strongly recommend that you use `xls` and `doc` as your file extensions, and this will distinguish these files from the 'genuine' `xlsx` and `docx` binary files.

If, for some reason, you do need **tabout** to produce native (binary) `xls` files (using the `Mata` `xl()` class), you can specify this with the `style(xls)` option, and **tabout** will then pass this requirement through to the `Mata` function. Make sure that you also use `xls` in your file extension to avoid confusion.

14.2. Features of xlsx files

While the docx output style had the landscape option, and used 'Autofit to Contents' for setting the width, the xlsx output style has neither of these (and landscape doesn't really make sense in a spreadsheet environment). On the other hand, the xlsx has a number of features not shared by docx:

- You have control over the label column width (the lwidth option) and the data columns width (the cwidth option). The default for these is 35 characters for the label column and 12 characters for the data columns and these defaults seem to work quite well most of the time. ⟹ ⟹
- The columns of the data rows can be indented from the right with the indent option. The default is 2 characters. This allows you to both centre your data under the headings while still retaining right alignment. ⟹
- The border lines for the table are slightly thicker than the heading lines and panel lines. This is similar to LaTeX and is something which is currently not possible with the docx tables.
- You can place your table anywhere on a sheet in your xlsx spreadsheet, specifying the columm and row coordinates of the top corner of the table.
- You can display multiple tables on the one sheet, also nominating their location on that sheet.
- You can also display multiple tables across multiple sheets in your xlsx file.

These last operations (multiple tables) can be slow but this may improve over time once *Stata* refines the underlying Mata functions. It is worth noting that the intrinsic design of **tabout** itself contributes to slowness with docx and xlsx output styles. When multiple panels are requested **tabout** re-runs the table production process for each panel, writing this out to files with each pass. When it comes to ascii (plain text) files, the file writing process is extremely fast, but with binary files, such as docx and xlsx, this repeated re-writing of the file is inherently much slower.

There is one annoying shortcoming in the Mata function for xlsx files: if you merge some cells in your table, the contents are automatically centred. While this makes sense for some situations (such as column headings), it does not make sense for other situations (such as notes below rows). **tabout** can place the sample size and the statistics in a cell beneath some data cells, and it places the table footnotes in a cell beneath the table. Ideally, one would like to have these cells merged across the table, but to do this automatically centres the contents (which looks terrible). At present

the option of 'left alignment' inside a merged cell is not possible (though hopefully this will change). Consequently, the default in **tabout** is to leave the cells unmerged. This means that if the contents of these cells is long, users may need to make manual adjustments to ensure that the final appearance of the table is satisfactory.

In some tables the default column width and indent settings are not optimal. Specifying confidence intervals, for example, can produce cell content which is too wide for the default column width. In several of the example tables (eg. `tables15d.do` and `table21d.do`) this crowding has been fixed using `cwidth(15)` and `indent(1)` to create a better result. If you run the **tabout** code with and without these settings you will notice a considerable difference.

14.3. Sheets and locations

One of the unique features of the *xlsx* output style is that it allows multiple tables on a single sheet, as well as tables spread across multiple sheets in a single *xlsx* file. **tabout**'s `location` option and `sheet` option provide you with this capability.

Location

The location feature is shown in the following screenshot, and the tabout code is shown for Table 40 and Table 41. Using the `location` option is straightforward: simply give the row and column co-ordinates (as numbers) for the top left corner of your table. The default position, if you don't

specify this option, is row 1 column 1.

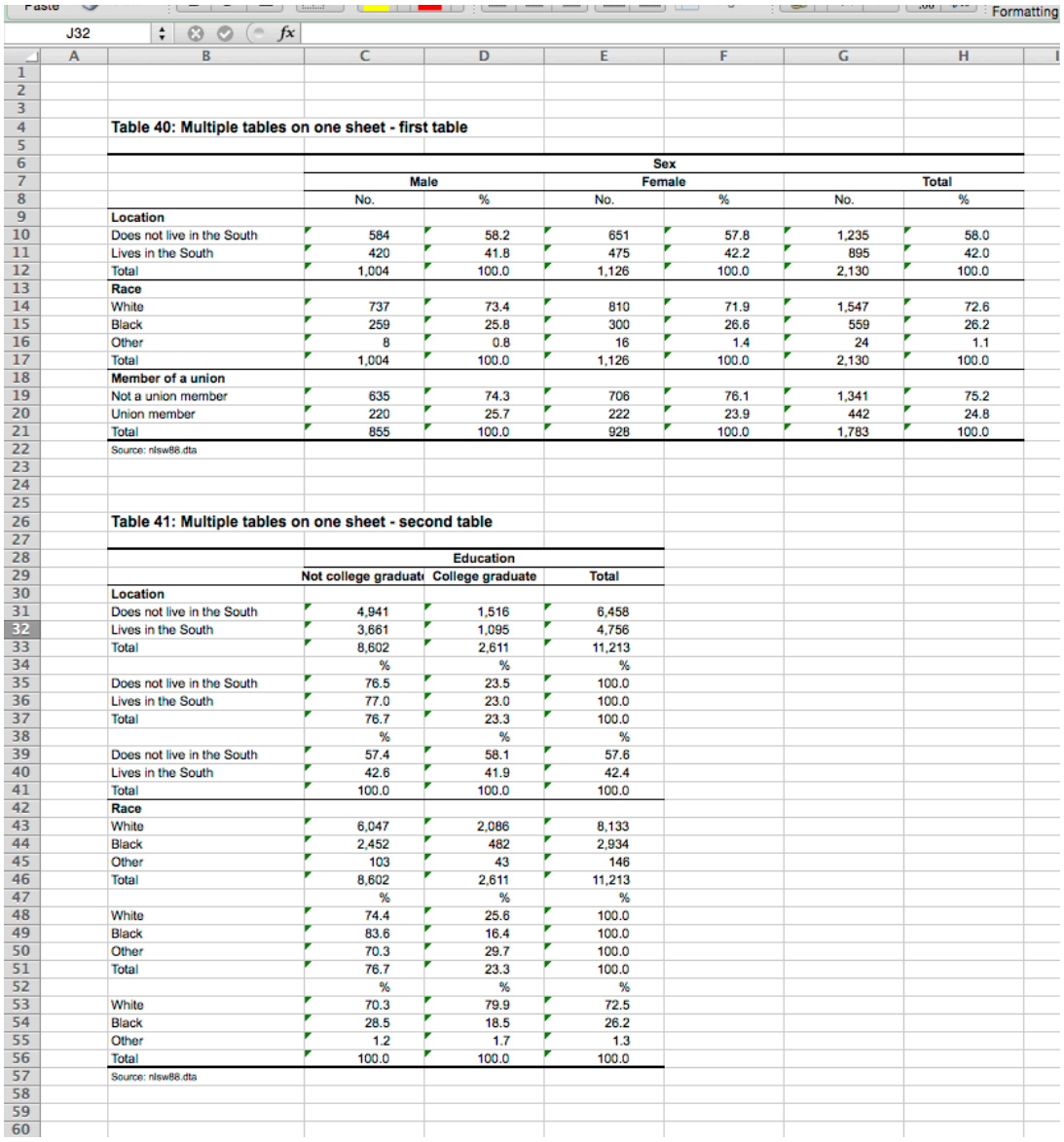


Table 40: Multiple tables on one sheet - first table

	Sex					
	Male		Female		Total	
	No.	%	No.	%	No.	%
Location						
Does not live in the South	584	58.2	651	57.8	1,235	58.0
Lives in the South	420	41.8	475	42.2	895	42.0
Total	1,004	100.0	1,126	100.0	2,130	100.0
Race						
White	737	73.4	810	71.9	1,547	72.6
Black	259	25.8	300	26.6	559	26.2
Other	8	0.8	16	1.4	24	1.1
Total	1,004	100.0	1,126	100.0	2,130	100.0
Member of a union						
Not a union member	635	74.3	706	76.1	1,341	75.2
Union member	220	25.7	222	23.9	442	24.8
Total	855	100.0	928	100.0	1,783	100.0

Source: nlsw88.dta

Table 41: Multiple tables on one sheet - second table

	Education		
	Not college graduat	College graduate	Total
Location			
Does not live in the South	4,941	1,516	6,458
Lives in the South	3,661	1,095	4,756
Total	8,602	2,611	11,213
	%	%	%
Does not live in the South	76.5	23.5	100.0
Lives in the South	77.0	23.0	100.0
Total	76.7	23.3	100.0
	%	%	%
Does not live in the South	57.4	58.1	57.6
Lives in the South	42.6	41.9	42.4
Total	100.0	100.0	100.0
Race			
White	6,047	2,086	8,133
Black	2,452	482	2,934
Other	103	43	146
Total	8,602	2,611	11,213
	%	%	%
White	74.4	25.6	100.0
Black	83.6	16.4	100.0
Other	70.3	29.7	100.0
Total	76.7	23.3	100.0
	%	%	%
White	70.3	79.9	72.5
Black	28.5	18.5	26.2
Other	1.2	1.7	1.3
Total	100.0	100.0	100.0

Source: nlsw88.dta

Copy table 40 code as: xlsx

```
do nlsw_data_setup
tabout south race union sex using table40.xlsx, ///
replace c(freq col) f(0c 1) font(bold) style(xlsx) ///
title(Table 40: Multiple tables on one sheet - ///
first table) fn(Source: nlsw88.dta) ///
location(4 2) lwidth(25) cwidth(15)
```

Copy table 41 code as: xlsx

```
do nlsw_data_setup
tabout south race collgrad [iw=wt] ///
using table40.xlsx, append style(xlsx) ///
c(freq row col) f(0c 1 1) layout(rb) h3(nil) ///
font(bold) title(Table 41: Multiple tables on ///
one sheet - second table) fn(Source: nlsw88.dta) ///
location(26 2) lwidth(25) cwidth(15)
```

In this example, both tables share the second column. The first table is on row 4, the second table on row 26. One feature of spreadsheets is that if the tables are in a vertical layout, such as this, they must share the same column widths. Thus both tables use a `lwidth` (*label width*) setting ⟹ of 25 and a `cwidth` (*column width*) setting of 15. If you omit these for the ⟹ first table, the defaults will apply, but if you then specify these settings for the second table, they will over-ride the first table settings. If it doesn't suit you to have tables sharing columns, consider placing them in a horizontal layout.

The other important point in this example is the `append` option and ⟹ the use of the same output file name. These are obviously needed for the two tables to be on the same sheet.

Sheet names

If you don't provide a name for the `sheet` option, **tabout** sets a default ⟹ of 'Sheet1', which is the first sheet in the spreadsheet. This if fine if you want multiple tables on just the first sheet. However, if you want multiple tables spread across multiple sheets you must specify sheet names (though you can use 'Sheet1', 'Sheet2', etc). If you don't specify a `location` when producing multiple sheets, **tabout** uses the default (1 1) on each of the sheets.

You can, if you wish, specify multiple locations, using the `location` ⟹ option and multiple sheets, using the `sheet` option. The following four ⟹ tables show this strategy. There is no screenshot (I simply repeat the above tables) but the full code is shown.

You will need to specify `append` for the last three tables so they all ⟹ share the same file (and all four tables must share the same file name). In this example, on the second sheet a horizontal layout has been chosen, which has allowed for different column widths for Table 44 and Table 45. The former clearly did not need a column width of 15, whereas the former needed this to accommodate the longer labels.

Copy table 42 code as: xlsx

```
do nlsw_data_setup
tabout south race union sex using table42.xlsx, ///
replace c(freq col) f(0c 1) font(bold) style(xlsx) ///
title(Table 42: Multiple tables on multiple sheets - ///
first table on first sheet) fn(Source: nlsw88.dta) ///
location(4 2) lwidth(25) cwidth(15) ///
sheet(My first sheet)
```

Copy table 43 code as: xlsx

```
do nlsw_data_setup
tabout south race collgrad [iw=wt] ///
using table42.xlsx, append style(xlsx) ///
c(freq row col) f(0c 1 1) layout(rb) h3(nil) ///
font(bold) title(Table 43: Multiple tables on ///
multiple sheets - second table on first sheet) ///
fn(Source: nlsw88.dta) location(26 2) ///
lwidth(25) cwidth(15) sheet(My first sheet)
```

Copy table 44 code as: xlsx

```
do nlsw_data_setup
tabout south race union sex using table42.xlsx, ///
append c(freq col) f(0c 1) font(bold) style(xlsx) ///
title(Table 44: Multiple tables on multiple sheets - ///
third table on second sheet) fn(Source: nlsw88.dta) ///
location(10 2) lwidth(25) cwidth(11) ///
sheet(My second sheet)
```

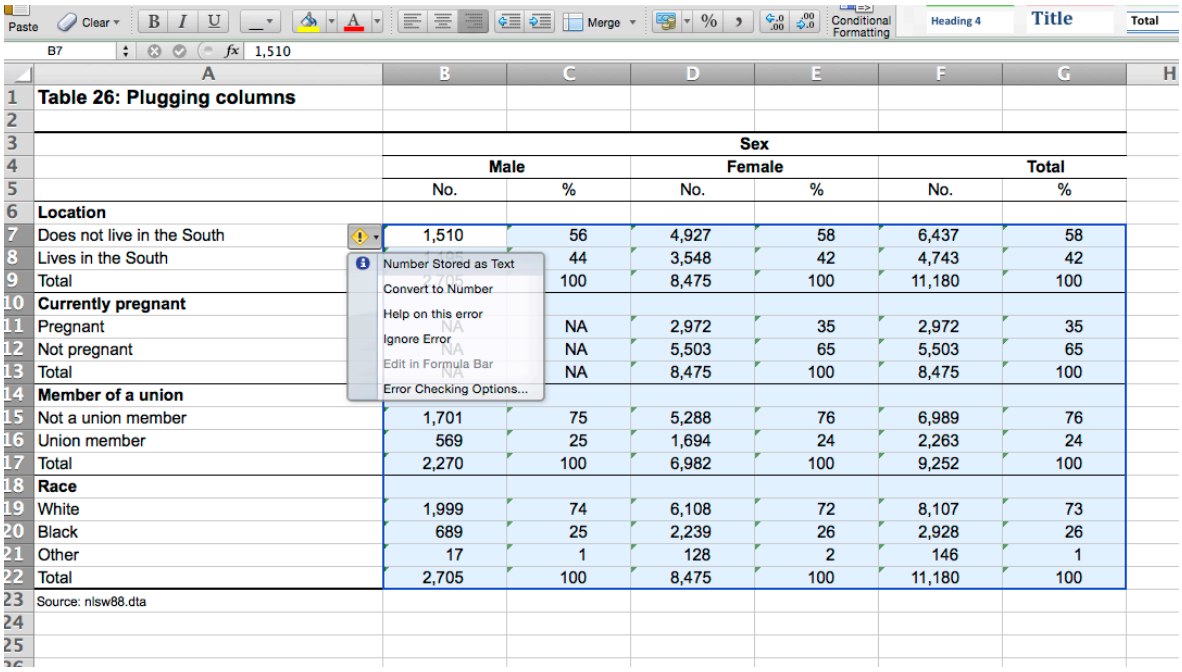
Copy table 45 code as: xlsx

```
do nlsw_data_setup
tabout south race collgrad [iw=wt] ///
using table42.xlsx, append style(xlsx) ///
c(freq row col) f(0c 1 1) layout(rb) h3(nil) ///
font(bold) title(Table 45: Multiple tables on ///
multiple sheets - fourth table on second sheet) ///
fn(Source: nlsw88.dta) location(10 13) ///
lwidth(25) cwidth(18) sheet(My second sheet) open
```

14.4. Numeric data?

When you open your `xlsx` tables in *Excel* you should see an almost perfectly formatted table. Even though you are in a spreadsheet application, the data cells will contain character data, that is, non-numeric data. *This is as it should be*, since **tabout**'s raison-de-etre is to produce publication quality tables, that is, tables which are camera-ready for publication. Part of the formatting process which **tabout** carries out entails inserting commas or full stops into strings, inserting various symbols, such as % or $, and surrounding text with various types of brackets.

However, it may be the case that you need numeric data in your table, perhaps because you intend to do further calculations on the data, or perhaps because you wish to create *Excel* graphs from your data (even though *Stata* graphs are superior!). If you are in this situation, the following screenshot is relevant.



	A	B	C	D	E	F	G
1	**Table 26: Plugging columns**						
2							
3		**Sex**					
4		**Male**		**Female**		**Total**	
5		No.	%	No.	%	No.	%
6	**Location**						
7	Does not live in the South	1,510	56	4,927	58	6,437	58
8	Lives in the South		44	3,548	42	4,743	42
9	Total		100	8,475	100	11,180	100
10	**Currently pregnant**						
11	Pregnant		NA	2,972	35	2,972	35
12	Not pregnant		NA	5,503	65	5,503	65
13	Total		NA	8,475	100	8,475	100
14	**Member of a union**						
15	Not a union member	1,701	75	5,288	76	6,989	76
16	Union member	569	25	1,694	24	2,263	24
17	Total	2,270	100	6,982	100	9,252	100
18	**Race**						
19	White	1,999	74	6,108	72	8,107	73
20	Black	689	25	2,239	26	2,928	26
21	Other	17	1	128	2	146	1
22	Total	2,705	100	8,475	100	11,180	100
23	Source: nlsw88.dta						
24							
25							

You may have noticed that the top left hand corner of your cells contains a small triangle. When you hover over this, you get a diamond symbol with an exclamation mark, and when you click your right mouse button, you see the drop down menu shown in this screenshot. The second menu item, *Convert to Number*, will convert the selected cells to numeric data, and you can then do further calculations with these data.

In this example, there are some NA cells, for missings. This conversion respects this and you can use these cells in various formulae and still get accurate results. However, you do need to be aware that parentheses can be a problem with numeric conversions, since *Excel* regards them as indicating

negative numbers (this is an accounting convention). On the other hand, square brackets appear to leave the cell untouched, and calculations fail when applied to such cells because they contain character data. If you need to get standard errors into *Excel* for conversion, consider using **tabout**'s `seb` (*standard errors brackets none*) option and if you need to get confidence intervals into *Excel* consider creating your original tables with the `c(ub lb)` (*upper bound* and *lower bound*) options. ⟹ ⟹

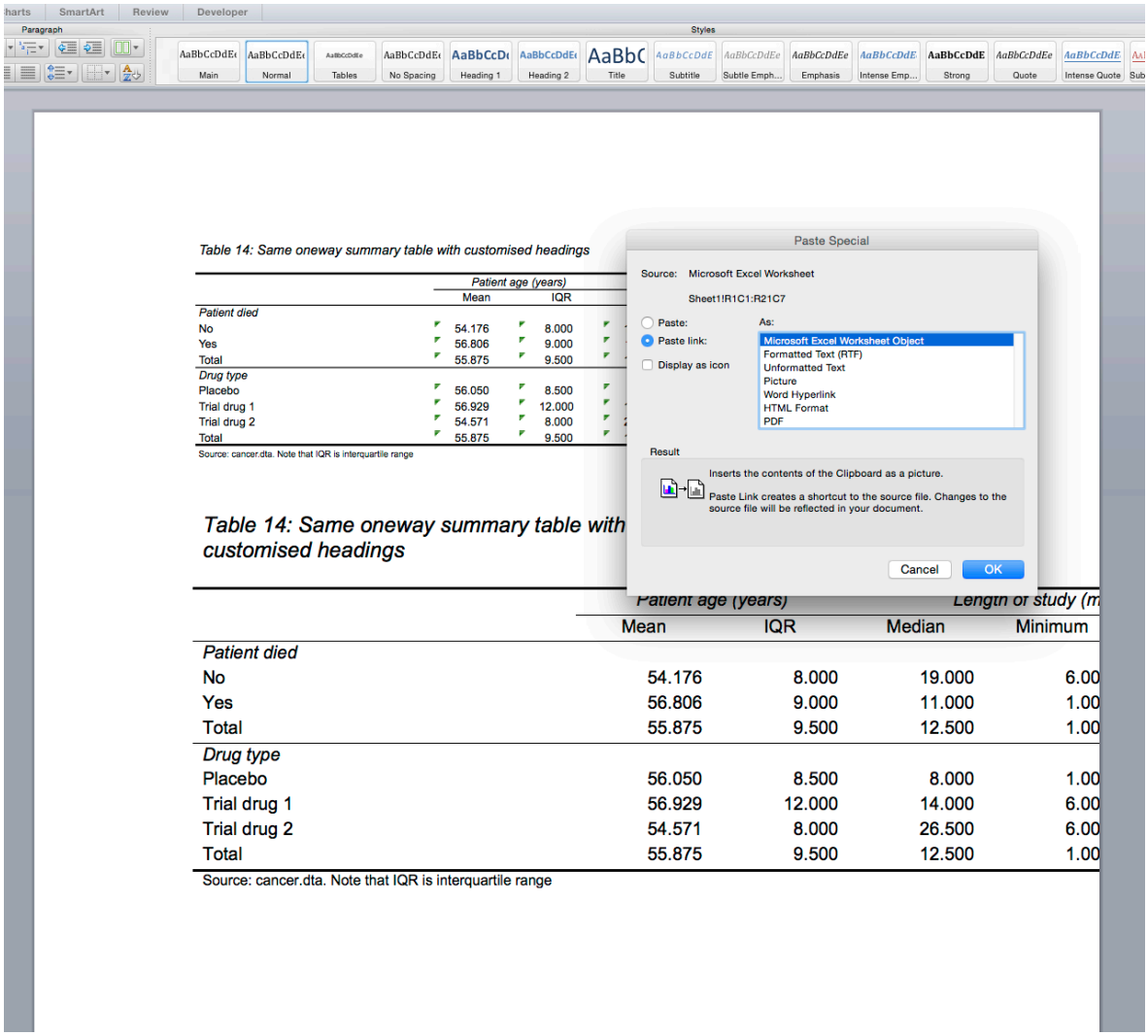
14.5. Working with xlsx files in Word

It is quite common for *Stata* users to bring their tables into *Word* via their spreadsheet application. If this is your desired method, then there are two ways to approach this. Once you have exported your table from **tabout** using the `xlsx` output style, you open it inside *Excel* and select the desired cells. You then copy these and paste them into *Word* using the *Edit - Paste Special* menu item. You have two choice regarding how you paste the table:

1. Microsoft Excel Worksheet Object; or
2. Formatted Text (RTF).

The screenshot on the following page shows both of these types of table in your *Word* document. Notice that the *Paste link* radio button is selected. This means that you can dynamically update the tables in *Word* when you rerun your **tabout** code. You do *not* need to go back into *Excel*. Once the file has been updated on your disk, *Word* can be made to update these links either automatically or using the *Edit - Links* menu item.

Several things are worth noting. If you plan on being interactive with your table—that is, you don't want full automation—then the first option might suit you. The table comes in as an image and can easily be resized on the screen by dragging the corner handles (as with all images). However, note that the corner triangle symbols are showing. If you can find a way to 'turn these off' in *Excel*, then this will solve that problem. Otherwise, you might need to convert the table to numeric data inside *Excel* before you paste it into *Word*.



In the case of the second option, you will notice that the table has extended beyond the page. This can be solved by either creating the table in **tabout** with narrower columns and a smaller font size, or by selecting the table inside *Word* and resizing it (such as the 'Autofit to Contents' option in the *Table* menu). Note that this second table is an actual *Word* table, and can be edited or formatted in any way you like inside *Word*. The first table, on the other hand, is an image and you can't manipulate the contents of the table inside *Word*, though double clicking on it will cause *Excel* to open the original `xlsx` table, where you can edit it.

15. html tables

15.1. The versatility of html

While `html` is the language of the World Wide Web, it is much more than that. It is an ideal output style for many end uses, including those who wish to import `html` files into *Word*. `html` is ideal for **tabout** users who cannot use the native `docx` output style, either because their *Stata* Release is older than Release 13 or for other reasons. It is a much better option than using tab delimited text files and then having to convert to a table and format the text.

Is it `htm` or html? Which you use is entirely up to you. The words are interchangable. In this guide I use `html` to refer to the files themselves (as they are given this extension in the examples) and `htm` to refer to the output style.

Consequently, in this chapter I will demonstrate both uses for **tabout**'s `htm` output: its suitability for web pages and for importing into *Word*. `html` is also suitable for converting documents into eBooks, but I don't offer any discussion of this. Applications such as *Calibre* will do this, but a search of the web is worthwhile if you want to go down this pathway.

Because `html` is a mark-up language and uses plain text, **tabout** produces the files very quickly. If you want to monitor how your tables look during a 'development' stage in your workflow, `html` is idea because the tables load quickly into your browser and you can view your results almost instantly. Just use the `open` option and the table will load automatically into your default browser. Each time you run **tabout** on the same table, a new copy opens in a new tab in your browser, so remember to close these tabs from time to time. ⟹

In the same way that you can 'trick' your operating system into opening tab delimited plain text files into *Excel* by giving them an 'xls' extension, you can also trick your system into opening `html` files in *Word* by giving them a `doc` extension. However, you need to remember to save your file as an *Word* file (either with `doc` or `docx` extensions) because *Word* knows that this file is a `html` file and will want to save it as such. I provide an example of this below.

While Version 2 of **tabout** provided `html` output, Version 3 provides much better `html` code, and also adds the ability to use cascading style sheets (CSS). This allows you to colour your text or your background, and to use alternate shading of your rows. Unfortunately, *Word* does not recognise everything in a CSS style sheet: it is fine with coloured text but it ignores, for example, alternate shading of rows.

15.2. Features and differences

Width settings

While LaTeX tables can be given an overall width, the width of columns can't be set. By way of contrast, when you select `htm` you can specify a `twidth` (*table width*) option, a `lwidth` (*label column width*) or a `cwidth` (*column width*). The `lwidth` refers to just the first column, the label column, and the `cwidth` to all the remaining 'data' columns. You can set all three, though this may be self-defeating. Fiddling with just two, at most, is more sensible. While the default unit of measurement for widths in LaTeX is cm, for the `htm` output style the default is `px` (pixels). The `units` option can set this to something else. In the case of `htm` it might be better to use `%` (percentage) sometimes, particularly if you are planning to open your tables in *Word*.

To see the difference between `twidth` option and the `cwidth` option look at the Table 5 code for the `htm` output style (`table5e.do`). This uses the `cwidth` option and this causes the value labels to wrap nicely inside the columns. If you compared this with the `table6e.do`, which uses the `twidth` option set at 500 pixels, you will notice when you run this, the appearance of the value labels is not as pleasing. As a general rule, setting the column widths (the `cwidth` option) rather than the table width seems to give the best results with the `htm` output style. You will notice that most of the example files use the `cwidth` option, usually set to between 80 and 100 pixels. The first column, with the variable and value labels, seems to adjust reasonably well by itself, without the need for the `lwidth` option.

Working with Word

The following code shows the `htm` code from Table 1 on page 16 and the two screenshots in Figure 15.1 show how it looks in a browser and how it looks in *Word*.

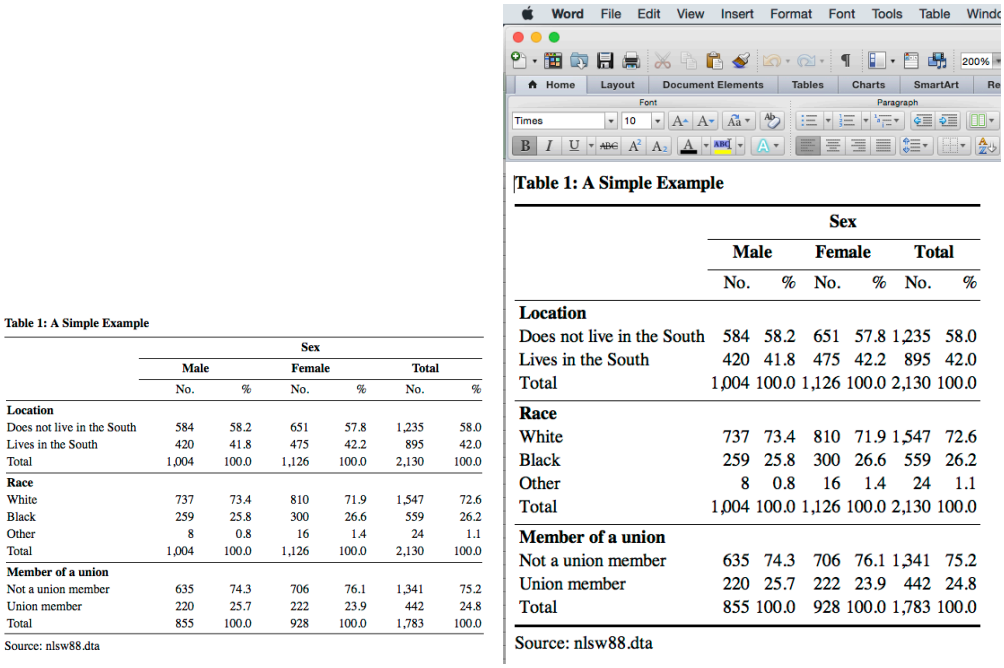


Table 1: A Simple Example

	Sex					
	Male		Female		Total	
	No.	%	No.	%	No.	%
Location						
Does not live in the South	584	58.2	651	57.8	1,235	58.0
Lives in the South	420	41.8	475	42.2	895	42.0
Total	1,004	100.0	1,126	100.0	2,130	100.0
Race						
White	737	73.4	810	71.9	1,547	72.6
Black	259	25.8	300	26.6	559	26.2
Other	8	0.8	16	1.4	24	1.1
Total	1,004	100.0	1,126	100.0	2,130	100.0
Member of a union						
Not a union member	635	74.3	706	76.1	1,341	75.2
Union member	220	25.7	222	23.9	442	24.8
Total	855	100.0	928	100.0	1,783	100.0

Source: nlsw88.dta

Figure 15.1.: How Table 1 looks in a browser and in *Word*

```
do nlsw_data_setup
tabout south race union sex using table1.html, replace ///
c(freq col) f(0c 1) style(htm) font(bold) cwidth(70) ///
title(Table 1: A Simple Example) fn(Source: nlsw88.dta)
```

You can see that *Word* preserves all the formatting of the table, and it actually places the `html` file into a proper *Word* table. It is *not* an image but is actual table text which can be edited just like a normal *Word* table.

The main problem with the table in the right panel, in *Word*, is that the columns aren't wide enough. It is generally better to use percentages for your widths if you plan to open `html` files in *Word*. I will illustrate this shortly.

You can always open `html` documents in *Word* from the file menu, or if you are working in a document and want to place the table at a particular location, you can use the *Insert - File* menu item. As mentioned above, if you just want the table to automatically open in *Word*, give it a file extension of `doc` and use the `open` option.

If you plan on working in *Word* using the `doc` extension and the `open` option, consider using percentages in the `units` option. Table 39, shown in Figure 15.2 illustrates this. The code for Table 39 is a slight modification of Table 1 but has been given a `doc` extension and the `open` option. The `twidth` option has been set to 100 and the `units` option has been set to `%`, for percentage.

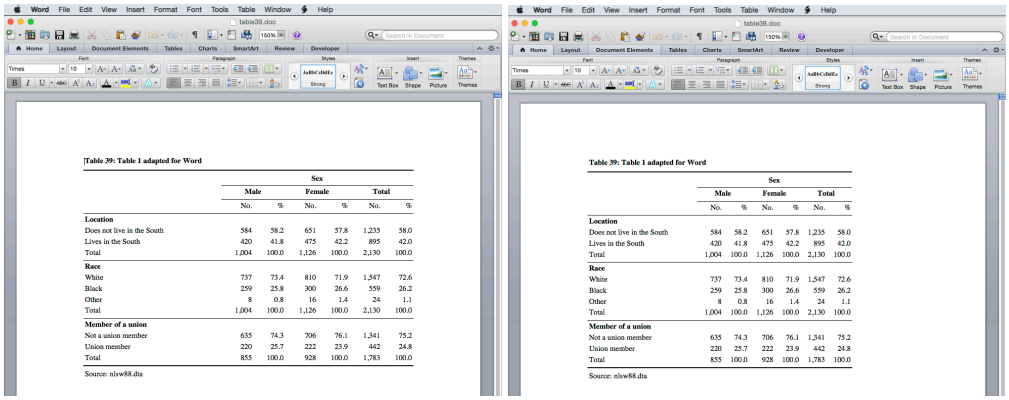


Table 39: Table 1 adapted for Word

	Sex					
	Male		Female		Total	
	No.	%	No.	%	No.	%
Location						
Does not live in the South	584	58.2	651	57.8	1,235	58.0
Lives in the South	420	41.8	475	42.2	895	42.0
Total	1,004	100.0	1,126	100.0	2,130	100.0
Race						
White	737	73.4	810	71.9	1,547	72.6
Black	259	25.8	300	26.6	559	26.2
Other	8	0.8	16	1.4	24	1.1
Total	1,004	100.0	1,126	100.0	2,130	100.0
Member of a union						
Not a union member	635	74.3	706	76.1	1,341	75.2
Union member	220	25.7	222	23.9	442	24.8
Total	855	100.0	928	100.0	1,783	100.0

Source: nlsw88.dta

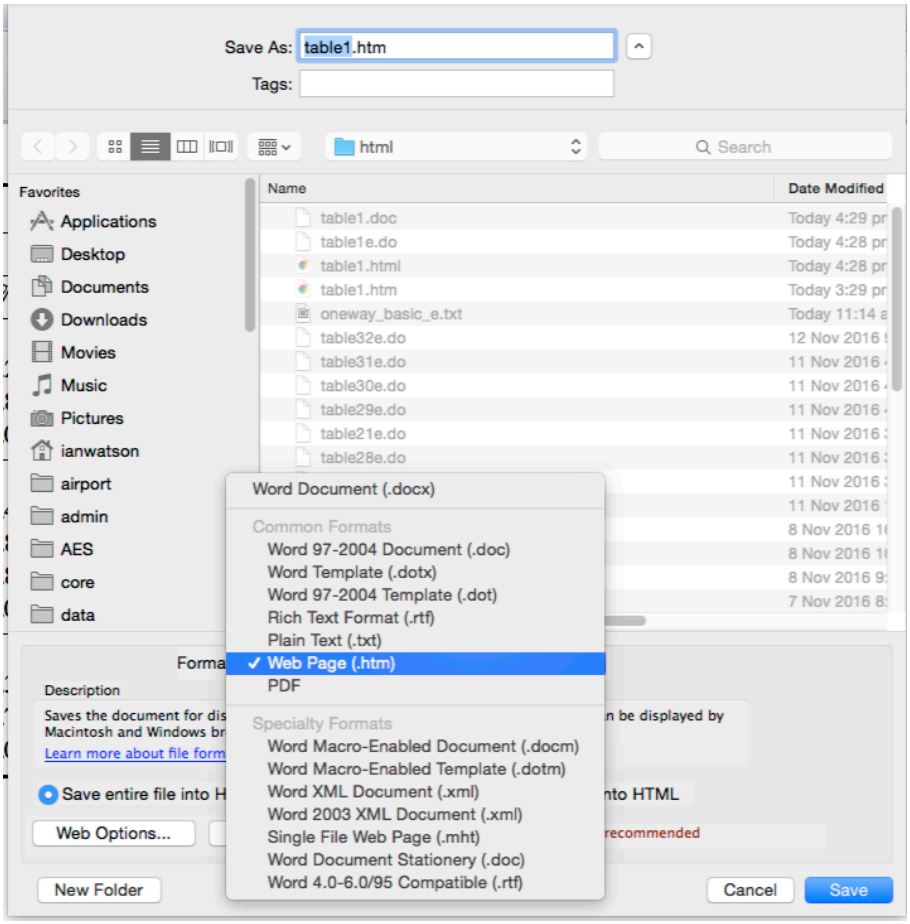
Figure 15.2.: Table 39 in *Word*: 100% and 80% width settings.

```
do nlsw_data_setup
tabout south race union sex using table39.doc, replace ///
c(freq col) f(0c 1) style(htm) font(bold) twidth(80) ///
units(%) title(Table 39: Table 1 adapted for Word) ///
fn(Source: nlsw88.dta) open
```

With a value of 100 percentage in the `twidth` option the table will use the full width of the *Word* document, as shown in the left panel. This may suit the rest of your document, but if you want to optimise the spacing in the table, try varying the percentage. The table in the right panel has been set to a percentage of 80, and has a more pleasing appearance. When fine tuning your `html` tables to place into *Word*, it is important that you change the *View* menu in *Word* from the *Web Layout* setting (which is what *Word* sets it to when it opens `html` documents) to the *Print Layout* setting if you want to get an accurate view of what your table will look like.

When it comes time to save your file in *Word*, make sure you select the format you want. The screenshot on the next page shows that when you first try to save the file, *Word* thinks it is a web page—no matter what extension you use–and you need to make sure that you select one of the Word formats from the list so that it becomes a 'normal' *Word* document.

When it comes to other aspects of your formatting, like fonts, families and font sizes, *`htm`* settings are no different to the other output styles. The `landscape` option does not apply in `html` files, because there is no concept ⟹ of a 'page' in `html`. Nor does the `tleft` *table left* option apply because ⟹ tables are never centred in *`htm`* output, but are automatically aligned to the left margin. If you would like centred tables, you can do this through customising your `html` and `css` code using **tabout**'s `topf( )` and `css` options. ⟹ ⟹ Keep in mind that the facilities for writing additional files are always available in **tabout** with `topf` and `botf`, and you can write any legitimate `html` that you like in these files.



15.2.1. Cascading style sheets: CSS

You will have noticed that you did not need to add a `body` option to get your `html` table to display in either your browser or in *Word*. This is unlike LaTeX where the `body` option is needed to compile the document. However, if you want to use the `css` option, you will need to add the `body` option to your **tabout** code.

What is CSS? Many applications, such as word processors, work with the concept of a 'style sheet', where a generic style (heading, quote etc) has a predefined look which is applied to any text whenever that style is selected. This contrasts with 'hard formatting' where the user applies the formatting features to specific bits of text. These formats need be re-applied each time the look of the text needs to change. Styles are thus a powerful way to automate the appearance of documents and to minimise tedious editing.

In the case of **tabout**, the `font` and `fsize` options are examples of 'hard formatting'. **tabout** applies these to your tables in a 'fixed' fashion

and you need to re-run your table code to change them. By contrast, the `css` option lets you specify a separate style file where various definitions of what your table should look like are to be found. If you wish to make the most of the `css` option in **tabout**, you will need some CSS skills. There are many CSS tutorials on the web as well as good books.

⟹ For a good reference see **Meyer2007**.

The example below is a simple one, which shows how the `css` option is used to alter the colour of the text, borders, lines and background. Aesthetically, this example has nothing to recommend it, but it is useful for illustrative purposes.

Table 46: Using CSS with html

	Sex						Sample size
	Male		Female		Total		
	%	%	%	%	%	%	
Location							
Does not live in the South	58.2	47.3	57.8	52.7	58.0	100.0	1235
Lives in the South	41.8	46.9	42.2	53.1	42.0	100.0	895
Total	100.0	47.1	100.0	52.9	100.0	100.0	2130
Sample size	1004		1126		2130		
Race							
White	73.4	47.6	71.9	52.4	72.6	100.0	1547
Black	25.8	46.3	26.6	53.7	26.2	100.0	559
Other	0.8	33.3	1.4	66.7	1.1	100.0	24
Total	100.0	47.1	100.0	52.9	100.0	100.0	2130
Sample size	1004		1126		2130		
Member of a union							
Not a union member	74.3	47.4	76.1	52.6	75.2	100.0	1341
Union member	25.7	49.8	23.9	50.2	24.8	100.0	442
Total	100.0	48.0	100.0	52.0	100.0	100.0	1783
Sample size	855		928		1783		

Source: nlsw88.dta

Copy table 46 code as: htm

```
do nlsw_data_setup
tabout south race union sex using table46.html, ///
replace c(col row) f(1) style(htm) font(italic) body ///
css(style.css) twidth(800) units(px) npos(both) ///
nlab(Sample size) title(Table 46: Using CSS with html) ///
fn(Source: nlsw88.dta) open
```

The content of the file which produces this effect is quite short and is shown below. One of the strengths of CSS is that it does not need to have code embedded in your text in order to 'target' parts of your document (though you can enhance this targetting by embedding `id` tags). In this example, the `<td>` tag targets all the cells in your table while the `<table>` tag targets the overall table. While the text for the document as a whole is black, the text inside the table is blue. If you have a reasonable knowledge of CSS, and you want even finer control over the elements of your table,

you could suppress the `html` code which **tabout** places around the table, and write your own `html` code to go in `topf` and `botf`. You need to specify the `ntc` (*no table code*) option to do this. In this way, you can assign CSS 'identifiers' to various parts (like the title and footnote) and gain considerable control over their appearance by simply styling these elements in certain ways in your CSS file.

```
body {
    background-color: white;
        font: 18px/24px "Lucida Grande",
    "Trebuchet MS", Arial, Helvetica,
    sans-serif;
    color: black;
}

table {
     border-style: solid solid solid solid;
     border-width: 3px;
     border-color: red;
     background: beige;
     color: blue;
}
td {
     text-align: left;
     border-color: red;
}
```

If you open Table 46 in *Word* it will look the same as in your browser because *Word* recognised the style sheet. However, not all elements in a style sheet are recognised by *Word* so you will need to experiment to find out what works for you.

15.3. Dynamic documents with html

Using `html` with Word

We saw earlier that the `docx` output style does not support the `topf` and `botf` options. The `htm` output style does support these options, but it is unlikely you would use this for dynamic documents in the way illustrated for LaTeX documents (see earlier). The reason for this is that things like footnotes and bibliographies are complicated using `html`, whereas a *Word* user can do these things relatively easy inside their word processing environment.

Instead, a more likely scenario is that you will write your main document inside *Word* and use the *File - Insert* menu item to insert at appropriate places the `html` tables produced by **tabout**. Because the `html` tables are automatically converted to *Word* tables, you have the option of fine tuning them by formatting them in the *Word* environment. There is, however, a downside to formatting tables inside *Word*. It destroys a workflow based on reproducible research. *Word* has an easily overlooked feature inside its *Insert File* dialogue box: *Link to File*. If this is ticked at the time of selecting your `html` file, *Word* keeps track of this file's location on disk. Then, if you change that file—for example, by re-running your **tabout** code and recreating your `html` tables—*Word* will know about it. All that you need to do, inside *Word*, is to periodically use the *Edit - Links* menu item and click on the *Update Now* button inside the *Links* dialogue box. Alternatively, you can set *Word* to automatically update all links when it opens files (using *Word*'s preferences settings).

This approach has many benefits. It allows *Word* users to work in an environment they are comfortable with while still gaining the benefits of reproducible research. All your **tabout** code can be re-run as often as necessary—such as when the data changes—and the *Word* document can be automatically updated. Not only is there no wastage of time in reformatting *Word* tables which have now changed, but the process is much more accurate because there is less room for error. Any task which is repetitious—such as copying and pasting numerous times, or inserting the same files numerous times—is always prone to error.

This approach could be extended dynamic documents using the same approach outlined in the LaTeX chapter. Because you can use the `topf` and `botf` options with the `htm` output style, you can use placeholders and *Stata* macros to place dynamic results into your *Word* document. Simply use the *Insert File/Link to File/Edit - Links/Update Now* sequence with your `html` file. You will find that all of the `html` document, both the table and the dynamic text, will be automatically updated inside word. If you want to use particular formatting, such as a bold font, for some of this extra text—and this is desirable as you don't want to manually reformat the imported

document—then you only need to learn a small amount of `html` or CSS code. For example, `html` works with simple 'tags', a system based on idea of an opening tag (`<tagname>`) and a closing tag (`</>`) surrounding the words. Thus a phrase like 'my key results' can be bolded by typing `<strong>my key results</strong>` inside either `topf` or `botf` files. Headings can also be inserted, using `html` headings, and graphics could also be brought in, using this method. In the next section I illustrate this approach in the context of dynamic content for web pages.

Finally, in the same way that the *tex* output style allows you to embed code in your `title` and `fn` options, so do does the *htm* output style. If you look at `table14e.do` you'll see a new line break put in the title, and italic tags used in the footnote.

Dynamic content on web pages

In the world of web site creation, most modern web sites are dynamic, often using languages such as `php` with databases like `mysql` to create 'on-the-fly' content. Such web sites often make use of statistical tables and graphics which are created instantaneously, often by internet users who land on these sites. Within this scenario, the people creating such websites often have an IT background, though they may work with researchers who have good content knowledge.

Dynamic content on web pages using **tabout** is *not* part of this scenario. Rather, the purpose of this facility in **tabout** is to enable researchers to publish their results on the web, largely without the need for any IT support. The scenario I have in mind is a researcher, or a research team, who want to provide regular updates on a research project to a wider audience. The screenshot of the web page several pages further on illustrates this (you can also visit the example web page with your browser by clicking here). This is not a dynamic *web page* as such, but is actually a 'static' web page—in the internet sense—and is not at all similar to the dynamic ones created on web servers by `php`. It is 'dynamic' in the sense that it's content can be automatically regenerated with minimal effort.

The look of this web page has been adapted from a CSS style sheet by CSS Zen Garden (with the more aesthetically pleasing parts due to the original, and the less pleasing parts due to my modifications!) The look will change according to the width of your browser window, but fixed width web pages are also common on the web. The design aspects of web pages are beyond the scope of this *User Guide*, so I don't offer any more comments on this, nor discussion of the CSS behind it. There are many good books which cover this.

CSS Zen Garden homepage.

Download the CSS file behind this web page, and remember to rename the extension from `txt` to `css`.

For a good reference see **Meyer2007**.

Instead the emphasis here is on how easy it is to create such a page. The following code creates the complete web page; all that the user has to

do afterwards is move the three files onto their web server: table47.html which is the main page, graph.png which is the graphic, and dynamic.css which is the style sheet (which will probably already be on your server if you are just updating).

Copy table 47 code as: htm

```
do cancer_data_setup
histogram studytime, by(drug)
graph export graph.png, height(400) replace

sum age
local meanage = r(mean)
local meanage: di %3.2f ‘meanage’
sum stime
local meanstime = r(mean)
local meanstime: di %3.2f ‘meanstime’
local obs = r(N)

tabout  drug died using table47.html, replace ///
style(htm) font(italic) c(freq col row) f(0c) ///
twidth(400) title(Table 47: Dynamic content web page ///
example) fn(cancer.dta) topf(topblock.html) ///
botstr(‘meanage’| ‘meanstime’|‘obs’) ///
topstr($S_DATE) botf(bottomblock.html) ///
body family(Arial) css(dynamic.css) open
```

Monthly Study Reports

Latest findings for the XYZ study

***Latest results** for the month through to **6 Dec 2016**.*

Background to the Study

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Key findings

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Detailed findings

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Table 47: Dynamic content web page example

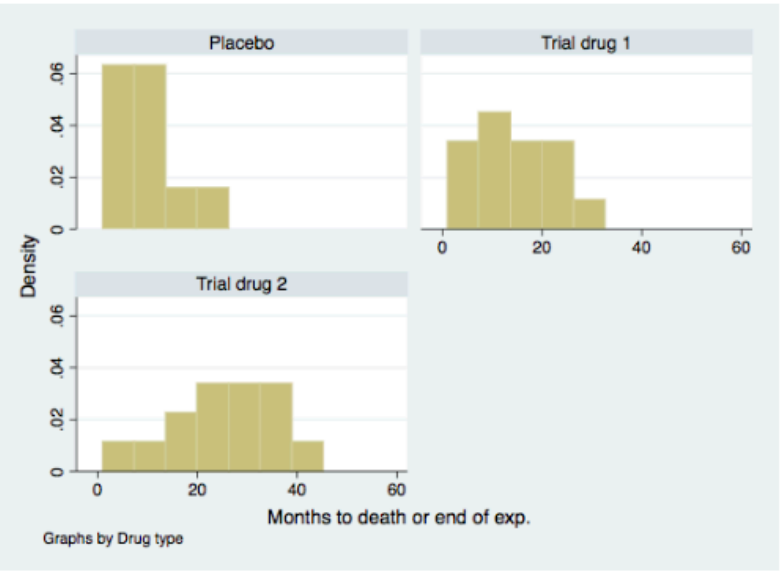
	Patient died								
	No			Yes			Total		
	No.	%	%	No.	%	%	No.	%	%
Placebo	1	6	5	19	61	95	20	42	100
Trial drug 1	8	47	57	6	19	43	14	29	100
Trial drug 2	8	47	57	6	19	43	14	29	100
Total	17	100	35	31	100	65	48	100	100

cancer.dta

Mean age of participants: 55.88 years.

Mean time to death, or until the end of the experiment: 2.04 months.

Experiment based on 48 observations.



If you have looked at the example of Table 35 in the LaTeX chapter, the code for Table 47 will be familiar. The **tabout** parts of this code create the actual table you see on the web page; the *Stata* `histogram` command creates the graph and the export command saves it to disk in a format suitable for the web. The more complicates bits are as follows:

- the `body` option creates the stand-alone code for the webpage and allows you to reference the CSS file, which is done with the `css(dynamic.css)` option;
- the `topf(topblock.html)` option brings in all the material *above* the table on the web page;
- the `botf(bottomblock.html)` option brings in all the material *below* the table on the web page;
- the `topstr` and `botstr` options pass the dynamic content (created earlier in the *Stata* code just above the **tabout** command) so that **tabout** can create the final `table47.html`.

This last point is important. As I mentioned earlier, the web page itself is not dynamic—in the internet sense of the word—but the content is dynamic in that every time you rerun your **tabout** code, the web page changes on your local computer to reflect any changes in your data or analysis. As soon as you copy `table47.html` to your web server, the world sees an updated web page.

The two files incorporated into `table47.html` by **tabout** are shown on the following pages. Ignore the Latin, it's just a text filler and ignore the cramped layout—`html` code is usually indented properly for ease of reading. The key points here are to look at the English wording and the use of the placeholder symbol #. This is where the dynamic content is placed.

In this example a single table is the 'core' of the web page, and this is made possible by combining `body`, `topf` and `botf`, with the latter two 'sandwiching' the table. If you wish to have multiple tables, then the 'nesting structure' outlined on page 70 is how you do it. That is, instead of `body` you use `topbody` and `replace` in your first **tabout** table; `append` in all the following **tabout** tables; and `botbody` in your final **tabout** table. How you use the `topf` and `botf` options throughout all your **tabout** code is up to your imagination—these facilities provide the textual and graphical material which is interspersed between your tables—and the combination you use is entirely up to you.

Copy topblock.html code as: htm

```
<div class="page-wrapper">
<section class="intro">
<h1>Monthly Study Reports</h1>
<h2>Latest findings for the XYZ study</h2>
<div class="summary">
<p><strong>Latest results</strong> for the month
through to <br> <strong>#</strong>.</p>
<p><strong>Background to the Study</strong><br>
Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in reprehenderit
in voluptate velit esse cillum dolore eu fugiat
nulla pariatur. Excepteur sint occaecat
cupidatat non proident, sunt in culpa qui officia
deserunt mollit anim id est laborum.</p>
</div>
<div class="preamble">
<h3>Key findings</h3>
<p>Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip
ex ea commodo consequat. Duis aute irure dolor
in reprehenderit in voluptate velit esse cillum
dolore eu fugiat nulla pariatur. Excepteur
sint occaecat cupidatat non proident, sunt in
culpa qui officia deserunt mollit anim id
est laborum.</p>
</div>
</section>
<div class="main supporting">
<h3>Detailed findings</h3>
<p>Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua. Ut enim ad
minim veniam, quis nostrud exercitation ullamco
laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in
voluptate velit esse cillum dolore eu fugiat
nulla pariatur. Excepteur sint occaecat cupidatat
non proident, sunt in culpa qui officia deserunt
mollit anim id est laborum.</p>
</div>
```

Copy bottomblock.html code as: htm

```
<div class="details">
<p>Mean age of participants: # years.</p>
<p>Mean time to death, or until the end
        of the experiment: # months.</p>
<p>Experiment based on # observations.</p>
</div>
<div class="graphs">
<img src="graph.png" alt="Graph">
</div>
<footer>For more information on the study
        contact ABC@def.org</footer>
</div>
```

If you find find the appearance of your web page does not look right, check for unbalanced `<div></div>` tags. Every opening tag must have a corresponding end tag somewhere in your sequence of files. For example, if you look at `topblock.html` you'll see an opening:

```
<div class = "page-wrapper">
```

but no closing tag in this file. Rather, the closing tag for this is to be found on the last line of `bottomblock.html`. If you spend a bit of time studying the matching `<div>` tags in any block of `html` code from the web, you'll soon grasp the idea of balanced tags.

Part IV.

Details

16. Key details

16.1. tabout syntax

The syntax for **tabout** follows the usual *Stata* conventions. If you are new to *Stata*, consult the manual for the meaning of *Stata*'s conventions regarding syntax. The core **tabout** syntax is:

```
tabout varlist [ if exp ] [ in range ] [ weight = exp ]
using filename [ , options ]
```

The additional information specific to **tabout** is as follows:

- the *varlist* is a list of 'vertical' (ie. row) variables, followed by one 'horizontal' (ie. column) variable, which is the last variable in the list. If the oneway option is specified, then all the variables are regarded as 'vertical'.
- *fweights aweights iweights* and *pweights* are allowed with **tabout**, depending on the underlying command; see *Stata*'s' **Manual: [U] 14.1.6 weight** and individual entries for **[R] tabulate** and **[R] summarize**. For tables of summary statistics, *iweights* are not allowed, because **tabout** uses the detail option in *Stata*'s **summarize** command (which does not allow *iweights*). The svy option requires that the data be already svyset and an error message reminds you of this if you forget.
- the using *filename* specifies the output file where your table will be sent. You need to supply an extension to the filename, and the extension will influence what applications on your computer will open the file is you specify the open option. The file will be saved in your working directory, or a directory specified with the full pathname as part of the filename.

16.2. Options: alphabetical listing

The following table is a complete listing of all the options. Note that if there is nothing in [red text] in the Comment column, then that particular option is available for *all* output styles. Otherwise, the applicable styles are shown. New features in Version 3 of **tabout** area shown in purple text and where an existing option has changed since Version 2, this is shown in orange text.

These options follow the *Stata* convention whereby abbreviations are sometimes allowed. What an option can be abbreviated to is shown in UPPER case, for example `APPend` can be abbreviated to `app` in your **tabout** syntax. If no uppercase is shown, the full term is required.

Option	Meaning	Arguments	Switch	Example	Comment
`APPend`	Add table to existing file		✓	34, 43	
`body`	Add code to create document		✓	35	[tex htm]
`botbody`	Add document code only to bottom of output file		✓		Useful for complex documents, particularly dynamic documents. New in Version 3 [tex htm]
`botf( )`	Stored code for use above table	File name		33	Requires place holders (#) for inserted text [txt tex htm]
`botstr( )`	Text to be inserted at placeholder	User text		33	Use delimiters for multiple strings (vertical bar, or user-defined via `delim( )`) [txt tex htm]
`caplab( )`	Caption label	User text		38	New in Version 3 [tex]
`cappos( )`	Caption position	*above* *below*		38	New in Version 3 [tex]
`CHKWTnone`	Don't check legality of weights		✓		*Stata* commands don't allow non-integer frequency weights. **tabout** normally checks for this but this option prevents this checking.
`ci2col`	Optimise UB and LB in two columns		✓		Requires `contents(lb ub)` and places '[' in first column and ']' in second. Achieves better alignment than `contents(ci)`.
`CIBnone`	Remove parentheses around confidence intervals		✓		
`cisep( )`	Specify separator for confidence intervals.	User text eg. `cisep(-)`		16	Default is a comma. User can choose any other separator.

Continued on next page...

Option	Meaning	Arguments	Switch	Example	Comment
clab( )	column label	User text		4	Replaces the heading 3 row with user text. Columns are separated by a space. Spaces for any text inside column must use underscores.
COMPile	compile LaTeX document		✓	35	User must set the global macro tex with location of LaTeX files. New in Version 3 [tex]
Contents( )	Specify cell contents	For options see table.		1	Note: this was previously called cell( ) but both are abbreviated to c( ). Changed in Version 3
css( )	Attach cascading style sheet (CSS) file.	File name		46	Requires body option be specified when using *htm* output style. New in Version 3 [htm]
CWidth( )	Column width	Number		40	Sets the width of columns. Measure is set by units. New in Version 3 [htm xlsx]
debug	Display underlying *Stata* commands which build basic tables				Can be useful for confirming your results. Does not show commands for summary tables.
delim( )	Specify delimiter	User text			Various options require delimiters. Default is vertical bar but this can be changed with this option. [txt tex htm]
doctype( )	Specify document type for LaTeX documents	*article* *report* *book* *letter*		38	Used by the body and topbody options, but can be changed with customised topf( ). New in Version 3 [tex]
DPComma	Use decimal point for comma		✓	17	Also uses full stop for thousands separator. Suits European *Stata* users.
dropc( )	Drop column(s)	Numbers indicating columns		23	User specifies column numbers, separated by spaces. Turn on show(*prepost*) to determine column number. New in Version 3
dropr( )	Drop row(s)	Numbers indicating panels and rows		24	User must specify panel number, followed by colon followed by row number (eg. 2:3) If multiple rows, use spaces as separators (eg 2:3 4:2). Turn on show(prepost) to determine row number. Note that the first row of data is the third row in a panel. New in Version 3

Continued on next page...

Option	Meaning	Arguments	Switch	Example	Comment
FAMily( )	Font family	User text			Also known as the typeface. Text must exactly match name used by user's operating system. Note that LaTeX users can specify two families (separated by the delimiter), the first being the roman family and the second being the sans serif family. New in Version 3 [tex htm docx xlsx]
fn( )	Place footnote below table	User text		1	Note that the font size is 0.8 of the table font size. New in Version 3
font( )	Specify font style for heading labels and variable labels	*bold* *italic* *none*		1	Also applied to table title if title is specified. [tex htm docx xlsx]
Format( )	Number of decimal points in data cells	Number followed by optional letter		1	Unlike *Stata* only a number is needed. Optional letters are: c (comma) p (percentage) m (money). The money() option specifies the currency. See also dpcomma for European formating.
fsize( )	Font size	Number			Measured in points. Default is 10pt. New in Version 3 [tex htm docx xlsx]
h1( )	Heading row 1	User text but *nil* suppresses this row		34	Separate column items must be separated with spaces. Underscores needed for spaces within column. Must use hc1 if not wanting to fully code this.
h1c( )	Heading row 1 columns	Numbers			The span across columns for each item in h1. New in Version 3
h2( )	Heading row 2	User text but *nil* suppresses this row		12	Separate column items must be separated with spaces. Underscores needed for spaces within column. Must use hc2 if not wanting to fully code this.
h2c( )	Heading row 2 columns	Numbers		12, 28	The span across columns for each item in h2. New in Version 3
h3( )	Heading row 3	User text but *nil* suppresses this row		28,29	Separate column items must be separated with spaces. Underscores needed for spaces within column. Must use hc3 if not wanting to fully code this.
h3c( )	Heading row 3 columns	Numbers		28, 29	The span across columns for each item in h3. New in Version 3
hright	Right alignment for heading rows		✓		Default is centred. Only applied to columns that are not spanned. New in Version 3 [docx]

Continued on next page...

Option	Meaning	Arguments	Switch	Example	Comment
indent( )	Indent data cells	Number			Measured in characters. Default is 2 characters. New in Version 3 [xlsx]
LANDscape	Place table in landscape mode		✓	37, 39	New in Version 3 [tex docx]
LAYout( )	Layout for columns and rows	*col* *row* *CBlock* *RBlock*		3	
Level( )	Specify level for svy estimates	Number		21	Default is 95%.
LOCation( )	Location of table on spreadsheet	Numbers		40	Row and column coordinates for top left corner. Separated by space. New in Version 3 [xlsx]
ltrim( )	Left trim of mid rules	Number		14	Replaces cltr1 and cltr2. Changed in Version 3 [tex]
LWidth( )	Label width	Number		40	Width of label (first) column. New in Version 3 [htm xlsx]
mi	Display missing values		✓		Same as mi in tabulate.
MONey( )	Specify currency	User text			Provide symbols etc for formatting when money specified in format eg. format(*2m*) money(£)
MULTiplier( )	Multiply proportion	Number		18	For use with svy option when proportions displayed. Can show as percents with mult(*100*) or rate per thousand with mult(*1000*). Note this replaces earlier percent option. Changed in Version 3
nlab( )	Label for N counts	User text		2	Label for when npos specified eg. 'Sample size'.
nnoc	N no commas		✓		Suppress display of commas in N counts.
NOBORDer	No table borders		✓		Suppress top and bottom borders of table. [tex htm docx xlsx]
NOFFset( )	Offset for N counts	Number			Control placement of N counts. Number indicates how far from first column N counts should be placed.
NOHLines	No heading lines		✓		Suppress lines between heading rows. Changed in Version 3 [tex htm docx xlsx]
NOPLines	No panel lines		✓	37	Suppress lines between panels. Changed in Version 3 [tex htm docx xlsx]

Continued on next page...

Option	Meaning	Arguments	Switch	Example	Comment
npos( )	Position of N counts	*col* *row* *lab* *tufte*		2	
ntc	No table code		✓		Suppress the code around the table which is now automatically produced. Results in 'pure' output of table contents only. Suited to users who wish to customise their tables with their own code. New in Version 3 [tex htm]
nwt( )	N weight	Variable name			Assigns a different weight to the N counts. Suitable for producing population estimates. Note that you need to use the pop option when using survey data in order to achieve weighted population estimates rather than sample counts. See discussion below.
ONEway	Oneway table		✓	9	Tells **tabout** that the last variable in the list is *not* a 'horizontal' variable.
open	Open output file in application			35	Opens the output file in the operating system's default application for that file extension. New in Version 3
PAPer( )	Specify paper size for document	*letter* *legal* *A3* *A4* *B4JIS*			Default for *docx* is *letter*, for *tex* is *A4* but note that only *A4*, *legal* and *letter* are available for *tex* (though this can also be changed in *tex* with customised topf( ). New in Version 3 [tex docx]
pform( )	p-value format	Number			Number of decimal points in p-values. Default is 3. New in Version 3
plab( )	p-value label	User text		8	New in Version 3
plugc( )	Plug columns	Numbers indicating panels and columns		26	User must specify panel number, followed by colon followed by column number (eg. 2:3) If multiple columns, use spaces as separators (eg 2:3 4:2). Turn on show(prepost) to determine column number. New in Version 3
PLUGLab( )	Label for plug	User text		31	Only applicable to missing labels when plugging rows. Use h2 or h3 if filling missing column labels. New in Version 3

Continued on next page...

Option	Meaning	Arguments	Switch	Example	Comment
plugr( )	Plug rows	Numbers indicating panels and rows		31	User must specify panel number, followed by colon followed by row number (eg. 2:3) If multiple rows, use spaces as separators (eg 2:3 4:2). Turn on show(prepost) to determine row number. Note that the first row of data is the third row in a panel. New in Version 3
PLUGSYMbol( )	Symbol to display for missing values	User text		26, 31	Filler for the blank cells created by plugging. Default is a blank, but 0 or NA or - etc can be chosen. New in Version 3
pop	Show population estimates, not sample counts		✓		Uses weight variable specified by nwt. See discussion below.
ppos( )	p-value position	*none* *only* *below* *beside*			These positions are relative to the statistic. Whether the p-value is in a row or column depends on the stpos option. New in Version 3
PSymbol( )	Placeholder symbol	User text			For use in topf and botf to indicate where strings are inserted. Default is #. Note that in **tabout** Version 2 these needed to be on a newline. In Version3 they can be interspersed throughout the file. [tex htm txt]
PTOTal( )	Display panel totals	*none* *single* *all*			Default is all. NOTE THAT THIS OPTION IS NO LONGER SUPPORTED IN VERSION 3. THE CODE REMAINS, BUT IS BUGGY.
REPlace	Replace existing output file		✓		You will get a warning message if the file exists and you haven't specified replace.
ROTate( )	Rotate value labels	Number		37	Rotates the value labels for the 'horizontal' variable. Number specifies the angle, in degrees. [tex]
SEBnone	Suppress standard error parentheses		✓	19	The default setting places ( ) around standard errors and this removes these.
sheet( )	Sheet name in spreadsheet	User text		43	Any text (including spaces) is allowed. Default is 'Sheet1'. Required when sending tables to multiple sheets. New in Version 3 [xlsx]

Continued on next page...

Option	Meaning	Arguments	Switch	Example	Comment
show( )	What should display in *Stata*	*none* *all* *output* *prepost* *comp*		23, 35	Determines what user sees in *Stata*'s Results window. Default is *output*. *all* shows the matrices which contain the raw data. *prepost* shows matrices before and after reshaping tables. *comp* shows the compile results for LaTeX users. [tex htm text]
sort	Sort table values		✓		The values are sorted in descending order of frequency. It only applies in oneway tables.
ssf	Use sans serif font		✓		Specifies that the second family font be used for the table. New in Version 3 [tex]
stars	Use stars for p-values		✓	8	Stars are used to display significance of p-value. New in Version 3
stats( )	Display various table statistics	*chi2* *gamma* *V* *taub* *lrchi2*		8	Note that only *chi2* is available for svy tables. Note that the font size for the statistics (and their labels) will be 0.9 times the table font size.
stform( )	Format for statistics	Number		8	Number of decimal points in table statistics. Default is 3. New in Version 3
stlab( )	Statistics label	User text		8	New in Version 3
stpos( )	Statistics position	*col* *row*		8	Default is row. Note that this setting also determines absolute position of p-value but relative position of latter is set by ppos. New in Version 3
style( )	Output style	*tex* *htm* *xlsx* *xls* *docx* *tab* *csv* *semi*		1	Note that the last three styles are all text delimited files, with csv using commas and semi using semi-colons as their delimiters. tab uses tabs for delimiters and is the default value of style.
sum	Produce summary table		✓	12	This is mandatory if you want summary tables.
svy	Produce survey data table		✓	16	This is mandatory if you want your table based on survey data. You still need to use *Stata*'s svyset command to indicate weights etc.
title( )	Title for table	User text		1	Note that the title will be 1.2 times the table font size. New in Version 3

Continued on next page...

Option	Meaning	Arguments	Switch	Example	Comment
`tleft`	Place table against left margin		✓		Default is to place table in the centre. New in Version 3 [tex docx]
`topbody`	Add document code only to top of output file		✓		Useful for complex documents, particularly dynamic documents. New in Version 3 [tex htm]
`topf( )`	Stored code for use below table	File name		33	Requires place holders (#) for inserted text [txt tex htm]
`topstr( )`	Text to be inserted at placeholder	User text		33	Use delimiters for multiple strings (vertical bar, or user-defined via `delim( )`) [txt tex htm]
`TOTal( )`	Label for totals	User text			Use first label for 'horizontal' variable, second label for 'vertical' variables. If spaces in labels use underscores eg. `total(All_persons Total)`.
`tp( )`	Template file	File name		32	Name of plain text file which holds **tabout** options. Avoids need to type all options into **tabout** command. Note that each option must be on a new line, and the first line in the file is a description of what the template does. New in Version 3
`TWidth( )`	Table width	Number		1	If not specified, *Word* uses 'Autofit to Contents'. Default for `tex` is 14cm. New in Version 3 [tex htm docx]
`units( )`	Units of measurement	User text			Use meaningful units for your output. Default for `tex` is `cm` but any legitimate LaTeXmeasurement can also be used (eg. `\columnwidth`); default for `htm` is `px`; default for other styles: `%`. New in Version 3 [tex htm docx xlsx]
`wide( )`	Width of Mata matrices	Number			Used in conjunction with `show(all)` to set size of Mata matrices displayed.

16.3. Options: thematic listing

Option	Meaning	Arguments	Switch	Example	Comment
CORE					
APPend	Add table to existing file		✓	34, 43	
Contents( )	Specify cell contents	For options see table.		1	Note: this was previously called cell( ) but both are abbreviated to c( ). Changed in Version 3
ONEway	Oneway table		✓	9	Tells **tabout** that the last variable in the list is *not* a 'horizontal' variable.
style( )	Output style	*tex* *htm* *xlsx* *xls* *docx* *tab* *csv* *semi*		1	Note that the last three styles are all text delimited files, with csv using commas and semi using semi-colons as their delimiters. tab uses tabs for delimiters and is the default value of style.
sum	Produce summary table		✓	12	This is mandatory if you want summary tables.
svy	Produce survey data table		✓	16	This is mandatory if you want your table based on survey data. You still need to use *Stata*'s svyset command to indicate weights etc.
REPlace	Replace existing output file		✓		You will get a warning message if the file exists and you haven't specified replace.
show( )	What should display in *Stata*	*none* *all* *output* *prepost* *comp*		23, 35	Determines what user sees in *Stata*'s Results window. Default is *output*. *all* shows the matrices which contain the raw data. *prepost* shows matrices before and after reshaping tables. *comp* shows the compile results for LaTeX users. [tex htm text]
sort	Sort table values		✓		The values are sorted in descending order of frequency. It only applies in oneway tables.
FILES					
body	Add code to create document		✓	35	[tex htm]
botbody	Add document code only to bottom of output file		✓		Useful for complex documents, particularly dynamic documents. New in Version 3 [tex htm]
botf( )	Stored code for use above table	File name		33	Requires place holders (#) for inserted text [txt tex htm]

Continued on next page...

Option	Meaning	Arguments	Switch	Example	Comment
COMPile	compile LaTeX document		✓	35	User must set the global macro tex with location of LaTeX files. New in Version 3 [tex]
css( )	Attach cascading style sheet (CSS) file.	File name		46	Requires body option be specified when using *htm* output style. New in Version 3 [htm]
open	Open output file in application			35	Opens the output file in the operating system's default application for that file extension. New in Version 3
topbody	Add document code only to top of output file		✓		Useful for complex documents, particularly dynamic documents. New in Version 3 [tex htm]
topf( )	Stored code for use below table	File name		33	Requires place holders (#) for inserted text [txt tex htm]
tp( )	Template file	File name		32	Name of plain text file which holds **tabout** options. Avoids need to type all options into **tabout** command. Note that each option must be on a new line, and the first line in the file is a description of what the template does. New in Version 3
FORMATTING					
DPComma	Use decimal point for comma		✓	17	Also uses full stop for thousands separator. Suits European *Stata* users.
FAMily( )	Font family	User text			Also known as the typeface. Text must exactly match name used by user's operating system. Note that LaTeX users can specify two families (separated by the delimiter), the first being the roman family and the second being the sans serif family. New in Version 3 [tex htm docx xlsx]
font( )	Specify font style for heading labels and variable labels	*bold* *italic* *none*		1	Also applied to table title if title is specified. [tex htm docx xlsx]
Format( )	Number of decimal points in data cells	Number followed by optional letter		1	Unlike *Stata* only a number is needed. Optional letters are: c (comma) p (percentage) m (money). The money() option specifies the currency. See also dpcomma for European formating.

Continued on next page...

Option	Meaning	Arguments	Switch	Example	Comment
`fsize( )`	Font size	Number			Measured in points. Default is 10pt. New in Version 3 [tex htm docx xlsx]
`hright`	Right alignment for heading rows		✓		Default is centred. Only applied to columns that are not spanned. New in Version 3 [docx]
`indent( )`	Indent data cells	Number			Measured in characters. Default is 2 characters. New in Version 3 [xlsx]
`LANDscape`	Place table in landscape mode		✓	37, 39	New in Version 3 [tex docx]
`LAYout( )`	Layout for columns and rows	*col* *row* *CBlock* *RBlock*		3	
`MONey( )`	Specify currency	User text			Provide symbols etc for formatting when money specified in `format` eg. `format(2m) money(£)`
SURROUNDS					
`botstr( )`	Text to be inserted at placeholder	User text		33	Use delimiters for multiple strings (vertical bar, or user-defined via `delim( )`) [txt tex htm]
`caplab( )`	Caption label	User text		38	New in Version 3 [tex]
`cappos( )`	Caption position	*above* *below*		38	New in Version 3 [tex]
`fn( )`	Place footnote below table	User text		1	Note that the font size is 0.8 of the table font size. New in Version 3
`title( )`	Title for table	User text		1	Note that the title will be 1.2 times the table font size. New in Version 3
`topstr( )`	Text to be inserted at placeholder	User text		33	Use delimiters for multiple strings (vertical bar, or user-defined via `delim( )`) [txt tex htm]

Continued on next page...

Option	Meaning	Arguments	Switch	Example	Comment
HEADINGS					
`clab( )`	column label	User text		4	Replaces the heading 3 row with user text. Columns are separated by a space. Spaces for any text inside column must use underscores.
`h1( )`	Heading row 1	User text but *nil* suppresses this row		34	Separate column items must be separated with spaces. Underscores needed for spaces within column. Must use `hc1` if not wanting to fully code this.
`h1c( )`	Heading row 1 columns	Numbers			The span across columns for each item in `h1`. New in Version 3
`h2( )`	Heading row 2	User text but *nil* suppresses this row		12	Separate column items must be separated with spaces. Underscores needed for spaces within column. Must use `hc2` if not wanting to fully code this.
`h2c( )`	Heading row 2 columns	Numbers		12, 28	The span across columns for each item in `h2`. New in Version 3
`h3( )`	Heading row 3	User text but *nil* suppresses this row		28,29	Separate column items must be separated with spaces. Underscores needed for spaces within column. Must use `hc3` if not wanting to fully code this.
`h3c( )`	Heading row 3 columns	Numbers		28, 29	The span across columns for each item in `h3`. New in Version 3

Continued on next page...

Option	Meaning	Arguments	Switch	Example	Comment
WIDTH/LINES/TOTALS					
`CWidth( )`	Column width	Number		40	Sets the width of columns. Measure is set by units. New in Version 3 [htm xlsx]
`LWidth( )`	Label width	Number		40	Width of label (first) column. New in Version 3 [htm xlsx]
`NOBORDer`	No table borders		✓		Suppress top and bottom borders of table. [tex htm docx xlsx]
`NOHLines`	No heading lines		✓		Suppress lines between heading rows. Changed in Version 3 [tex htm docx xlsx]
`NOPLines`	No panel lines		✓	37	Suppress lines between panels. Changed in Version 3 [tex htm docx xlsx]
`TWidth( )`	Table width	Number		1	If not specified, *Word* uses 'Autofit to Contents'. Default for `tex` is 14cm. New in Version 3 [tex htm docx]
`PTOTal( )`	Display panel totals	*none* *single* *all*			Default is all. NOTE THAT THIS OPTION IS NO LONGER SUPPORTED IN VERSION 3. THE CODE REMAINS, BUT IS BUGGY.
`TOTal( )`	Label for totals	User text			Use first label for 'horizontal' variable, second label for 'vertical' variables. If spaces in labels use underscores eg. `total(All_persons Total)`.

Continued on next page...

Option	Meaning	Arguments	Switch	Example	Comment
RESHAPING					
dropc( )	Drop column(s)	Numbers indicating columns		23	User specifies column numbers, separated by spaces. Turn on show(prepost) to determine column number. New in Version 3
dropr( )	Drop row(s)	Numbers indicating panels and rows		24	User must specify panel number, followed by colon followed by row number (eg. 2:3) If multiple rows, use spaces as separators (eg 2:3 4:2). Turn on show(prepost) to determine row number. Note that the first row of data is the third row in a panel. New in Version 3
plugc( )	Plug columns	Numbers indicating panels and columns		26	User must specify panel number, followed by colon followed by column number (eg. 2:3) If multiple columns, use spaces as separators (eg 2:3 4:2). Turn on show(prepost) to determine column number. New in Version 3
PLUGLab( )	Label for plug	User text		31	Only applicable to missing labels when plugging rows. Use h2 or h3 if filling missing column labels. New in Version 3
plugr( )	Plug rows	Numbers indicating panels and rows		31	User must specify panel number, followed by colon followed by row number (eg. 2:3) If multiple rows, use spaces as separators (eg 2:3 4:2). Turn on show(prepost) to determine row number. Note that the first row of data is the third row in a panel. New in Version 3
PLUGSYMbol( )	Symbol to display for missing values	User text		26, 31	Filler for the blank cells created by plugging. Default is a blank, but 0 or NA or - etc can be chosen. New in Version 3
SURVEY					
ci2col	Optimise UB and LB in two columns		✓		Requires contents(*lb ub*) and places '[' in first column and ']' in second. Achieves better alignment than contents(*ci*).
CIBnone	Remove parentheses around confidence intervals		✓		
cisep( )	Specify separator for confidence intervals.	User text eg. cisep(-)		16	Default is a comma. User can choose any other separator.
Level( )	Specify level for svy estimates	Number		21	Default is 95%.

Continued on next page...

Option	Meaning	Arguments	Switch	Example	Comment
MULTiplier( )	Multiply proportion	Number		18	For use with svy option when proportions displayed. Can show as percents with mult(100) or rate per thousand with mult(1000). Note this replaces earlier percent option. Changed in Version 3
SEBnone	Suppress standard error parentheses		✓	19	The default setting places ( ) around standard errors and this removes these.
N COUNT					
nlab( )	Label for N counts	User text		2	Label for when npos specified eg. 'Sample size'.
nnoc	N no commas		✓		Suppress display of commas in N counts.
NOFFset( )	Offset for N counts	Number			Control placement of N counts. Number indicates how far from first column N counts should be placed.
npos( )	Position of N counts	*col* *row* *lab* *tufte*		2	
ntc	No table code		✓		Suppress the code around the table which is now automatically produced. Results in 'pure' output of table contents only. Suited to users who wish to customise their tables with their own code. New in Version 3 [tex htm]
nwt( )	N weight	Variable name			Assigns a different weight to the N counts. Suitable for producing population estimates. Note that you need to use the pop option when using survey data in order to achieve weighted population estimates rather than sample counts. See discussion below. See discussion below.
pop	Show population estimates, not sample counts		✓		Uses weight variable specified by nwt. See discussion below.
STATISTICS					
pform( )	p-value format	Number			Number of decimal points in p-values. Default is 3. New in Version 3
plab( )	p-value label	User text		8	New in Version 3

Continued on next page...

Option	Meaning	Arguments	Switch	Example	Comment
ppos( )	p-value position	*none* *only* *below* *beside*			These positions are relative to the statistic. Whether the p-value is in a row or column depends on the stpos option. New in Version 3
stars	Use stars for p-values		✓	8	Stars are used to display significance of p-value. New in Version 3
stats( )	Display various table statistics	*chi2* *gamma* *V* *taub* *lrchi2*		8	Note that only *chi2* is available for svy tables. Note that the font size for the statistics (and their labels) will be 0.9 times the table font size.
stform( )	Format for statistics	Number		8	Number of decimal points in table statistics. Default is 3. New in Version 3
stlab( )	Statistics label	User text		8	New in Version 3
stpos( )	Statistics position	*col* *row*		8	Default is row. Note that this setting also determines absolute position of p-value but relative position of latter is set by ppos. New in Version 3
STYLE SPECIFIC					
doctype( )	Specify document type for LaTeX documents	*article* *report* *book* *letter*		38	Used by the body and topbody options, but can be changed with customised topf( ). New in Version 3 [tex]
LOCation( )	Location of table on spreadsheet	Numbers		40	Row and column coordinates for top left corner. Separated by space. New in Version 3 [xlsx]
ltrim( )	Left trim of mid rules	Number		14	Replaces cltr1 and cltr2. Changed in Version 3 [tex]
PAPer( )	Specify paper size for document	*letter* *legal* *A3* *A4* *B4JIS*			Default for *docx* is *letter*, for *tex* is *A4* but note that only *A4*, *legal* and *letter* are available for *tex* (though this can also be changed in *tex* with customised topf( ). New in Version 3 [tex docx]
ROTate( )	Rotate value labels	Number		37	Rotates the value labels for the 'horizontal' variable. Number specifies the angle, in degrees. [tex]

Continued on next page...

Option	Meaning	Arguments	Switch	Example	Comment
`sheet( )`	Sheet name in spreadsheet	User text		43	Any text (including spaces) is allowed. Default is 'Sheet1'. Required when sending tables to multiple sheets. New in Version 3 [xlsx]
`ssf`	Use sans serif font		✓		Specifies that the second family font be used for the table. New in Version 3 [tex]
`tleft`	Place table against left margin		✓		Default is to place table in the centre. New in Version 3 [tex docx]
DIAGNOSTIC					
`debug`	Display underlying *Stata* commands which build basic tables				Can be useful for confirming your results. Does not show commands for summary tables.
`mi`	Display missing values		✓		Same as `mi` in `tabulate`.
`CHKWTnone`	Don't check legality of weights		✓		*Stata* commands don't allow non-integer frequency weights. **tabout** normally checks for this but this option prevents this checking.
`wide( )`	Width of Mata matrices	Number			Used in conjunction with `show(all)` to set size of Mata matrices displayed.
CUSTOMISED					
`delim( )`	Specify delimiter	User text			Various options require delimiters. Default is vertical bar but this can be changed with this option. [txt tex htm]
`PSymbol( )`	Placeholder symbol	User text			For use in `topf` and `botf` to indicate where strings are inserted. Default is #. Note that in **tabout** Version 2 these needed to be on a newline. In Version3 they can be interspersed throughout the file. [tex htm txt]
`units( )`	Units of measurement	User text			Use meaningful units for your output. Default for *tex* is *cm* but any legitimate LaTeXmeasurement can also be used (eg. `\columnwidth`); default for *htm* is *px*; default for other styles: %. New in Version 3 [tex htm docx xlsx]

Population estimates

The `nwt` option allows you to use a different weight for the N counts in your table, as opposed to the table cells weights specified in the `[weight = exp ]` syntax. This can be useful for specifying population estimates. **tabout** always uses *Stata*'s `iweight` option for this weighting.

When you have survey tables, you need to add the `pop` option as well. This provides you with weighted estimates in your N counts. You can choose to use the same weight as specified in your `svyset` command, or you may specify a different weighting variables. For example, you may want the estimates in the table weighted by 'effective sample size' weights, while you may require that the N counts in the table show population estimates based on 'expansion' weights.