Creating Customized Tables with Stata

Kristin MacDonald

Executive Director of Statistical Services
StataCorp LLC

2022 Stata Conference

With Stata's features for customizable tables, you can ...

Create a table of summary statistics,

Table 1: Summary statistics

	Hypertension		
	No	Yes	Total
Age (years)	42.2 (16.8)	55.0 (14.9)	47.6 (17.2)
Body mass index (BMI)	24.2 (4.1)	27.4 (5.3)	25.5 (4.9)
Sex			
Male	2,611 (43.7%)	2,304 (52.7%)	4,915 (47.5%)
Female	3,364 (56.3%)	2,072 (47.3%)	5,436 (52.5%)
Race			
White	5,317 (89.0%)	3,748 (85.6%)	9,065 (87.6%)
Black	545 (9.1%)	541 (12.4%)	1,086 (10.5%)
Other	113 (1.9%)	87 (2.0%)	200 (1.9%)
Health status			
Excellent	1,649 (27.7%)	758 (17.3%)	2,407 (23.3%)
Very good	1,666 (27.9%)	925 (21.2%)	2,591 (25.1%)
Good	1,572 (26.4%)	1,366 (31.2%)	2,938 (28.4%)
Fair	766 (12.8%)	904 (20.7%)	1,670 (16.2%)
Poor	310 (5.2%)	419 (9.6%)	729 (7.1%)
Serum cholesterol (mg/dL)	208.7 (47.3)	229.9 (49.6)	217.7 (49.4)
Serum triglycerides (mg/dL)	129.2 (83.9)	166.0 (109.2)	143.9 (96.5)
High density lipids (mg/dL)	49.9 (14.1)	49.2 (14.5)	49.6 (14.3)

Mean and standard deviation reported for continuous variables.

And export your table to a Word document,

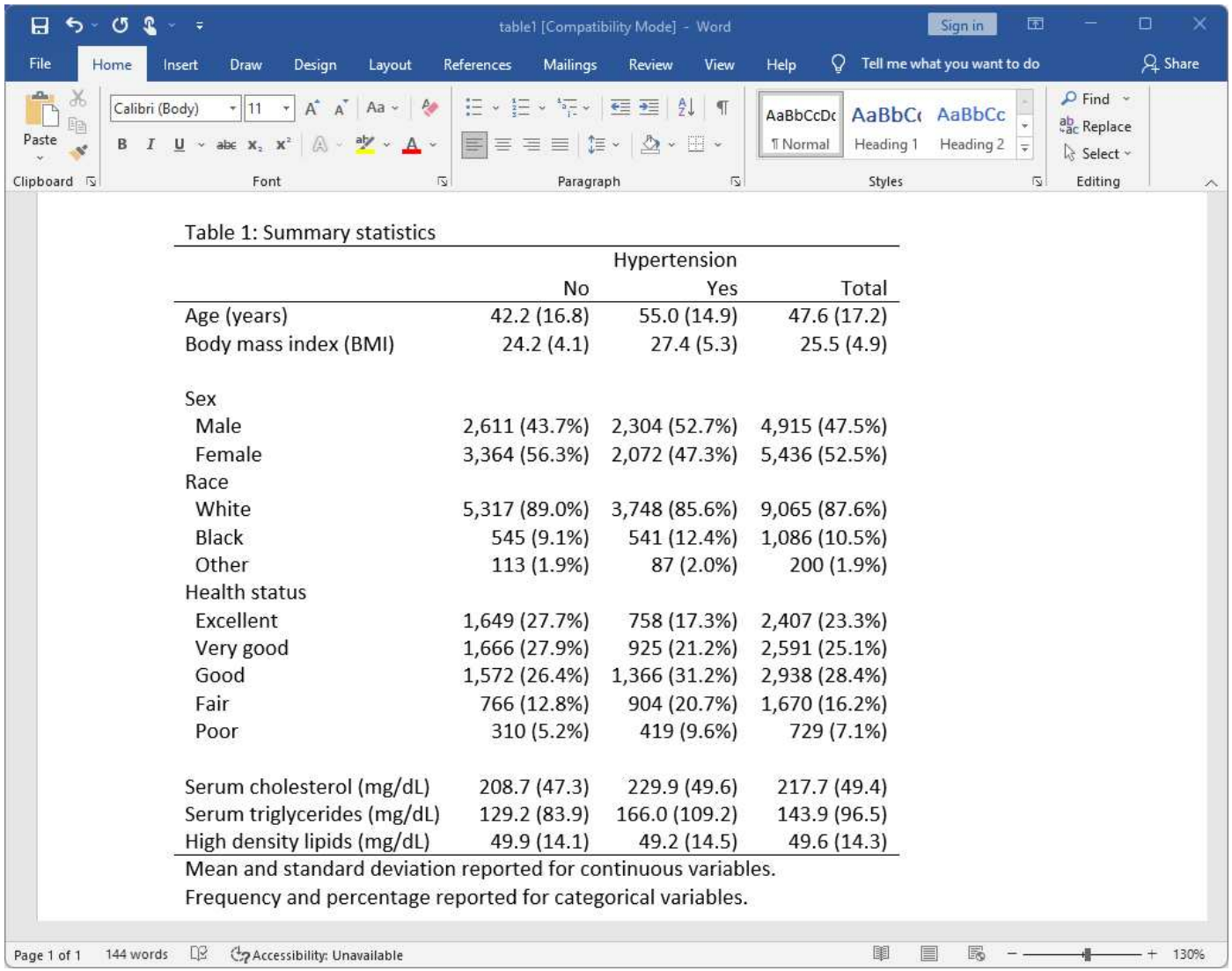


Table 1: Summary statistics

	Hypertension		
	No	Yes	Total
Age (years)	42.2 (16.8)	55.0 (14.9)	47.6 (17.2)
Body mass index (BMI)	24.2 (4.1)	27.4 (5.3)	25.5 (4.9)
Sex			
Male	2,611 (43.7%)	2,304 (52.7%)	4,915 (47.5%)
Female	3,364 (56.3%)	2,072 (47.3%)	5,436 (52.5%)
Race			
White	5,317 (89.0%)	3,748 (85.6%)	9,065 (87.6%)
Black	545 (9.1%)	541 (12.4%)	1,086 (10.5%)
Other	113 (1.9%)	87 (2.0%)	200 (1.9%)
Health status			
Excellent	1,649 (27.7%)	758 (17.3%)	2,407 (23.3%)
Very good	1,666 (27.9%)	925 (21.2%)	2,591 (25.1%)
Good	1,572 (26.4%)	1,366 (31.2%)	2,938 (28.4%)
Fair	766 (12.8%)	904 (20.7%)	1,670 (16.2%)
Poor	310 (5.2%)	419 (9.6%)	729 (7.1%)
Serum cholesterol (mg/dL)	208.7 (47.3)	229.9 (49.6)	217.7 (49.4)
Serum triglycerides (mg/dL)	129.2 (83.9)	166.0 (109.2)	143.9 (96.5)
High density lipids (mg/dL)	49.9 (14.1)	49.2 (14.5)	49.6 (14.3)

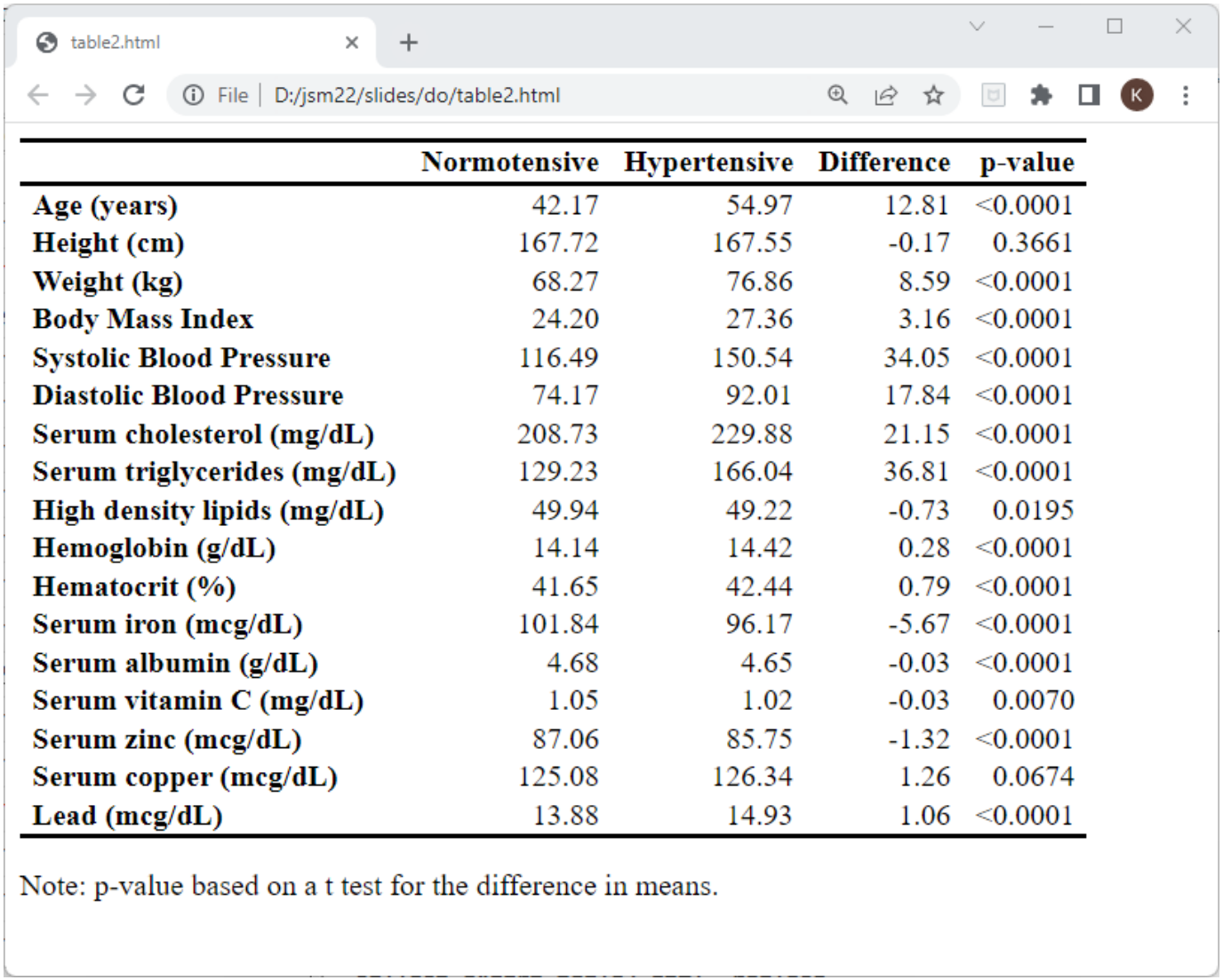
Mean and standard deviation reported for continuous variables.
Frequency and percentage reported for categorical variables.

Create a table of means and *t* tests of differences,

	Normotensive	Hypertensive	Difference	p-value
Age (years)	42.17	54.97	12.81	<0.0001
Height (cm)	167.72	167.55	-0.17	0.3661
Weight (kg)	68.27	76.86	8.59	<0.0001
Body Mass Index	24.20	27.36	3.16	<0.0001
Systolic Blood Pressure	116.49	150.54	34.05	<0.0001
Diastolic Blood Pressure	74.17	92.01	17.84	<0.0001
Serum cholesterol (mg/dL)	208.73	229.88	21.15	<0.0001
Serum triglycerides (mg/dL)	129.23	166.04	36.81	<0.0001
High density lipids (mg/dL)	49.94	49.22	-0.73	0.0195
Hemoglobin (g/dL)	14.14	14.42	0.28	<0.0001
Hematocrit (%)	41.65	42.44	0.79	<0.0001
Serum iron (mcg/dL)	101.84	96.17	-5.67	<0.0001
Serum albumin (g/dL)	4.68	4.65	-0.03	<0.0001
Serum vitamin C (mg/dL)	1.05	1.02	-0.03	0.0070
Serum zinc (mcg/dL)	87.06	85.75	-1.32	<0.0001
Serum copper (mcg/dL)	125.08	126.34	1.26	0.0674
Lead (mcg/dL)	13.88	14.93	1.06	<0.0001

Note: p-value based on a t test for the difference in means.

And export your table to HTML,



	Normotensive	Hypertensive	Difference	p-value
Age (years)	42.17	54.97	12.81	<0.0001
Height (cm)	167.72	167.55	-0.17	0.3661
Weight (kg)	68.27	76.86	8.59	<0.0001
Body Mass Index	24.20	27.36	3.16	<0.0001
Systolic Blood Pressure	116.49	150.54	34.05	<0.0001
Diastolic Blood Pressure	74.17	92.01	17.84	<0.0001
Serum cholesterol (mg/dL)	208.73	229.88	21.15	<0.0001
Serum triglycerides (mg/dL)	129.23	166.04	36.81	<0.0001
High density lipids (mg/dL)	49.94	49.22	-0.73	0.0195
Hemoglobin (g/dL)	14.14	14.42	0.28	<0.0001
Hematocrit (%)	41.65	42.44	0.79	<0.0001
Serum iron (mcg/dL)	101.84	96.17	-5.67	<0.0001
Serum albumin (g/dL)	4.68	4.65	-0.03	<0.0001
Serum vitamin C (mg/dL)	1.05	1.02	-0.03	0.0070
Serum zinc (mcg/dL)	87.06	85.75	-1.32	<0.0001
Serum copper (mcg/dL)	125.08	126.34	1.26	0.0674
Lead (mcg/dL)	13.88	14.93	1.06	<0.0001

Note: p-value based on a t test for the difference in means.

Create a table of regression results,

Table 3: Logistic regression results

	1	2	3
Body mass index (BMI)	1.145	1.147	1.148
	(0.006)	(0.006)	(0.006)
Age (years)	1.046	1.047	1.034
	(0.001)	(0.001)	(0.002)
Female		0.617	0.152
		(0.028)	(0.023)
Female x Age (years)			1.028
			(0.003)
Intercept	0.002	0.003	0.005
	(0.000)	(0.000)	(0.001)
AIC	11812.48	11698.99	11605.77
BIC	11834.21	11727.97	11642.00

Odds ratio (Std. Err.)

And export your table to a PDF,

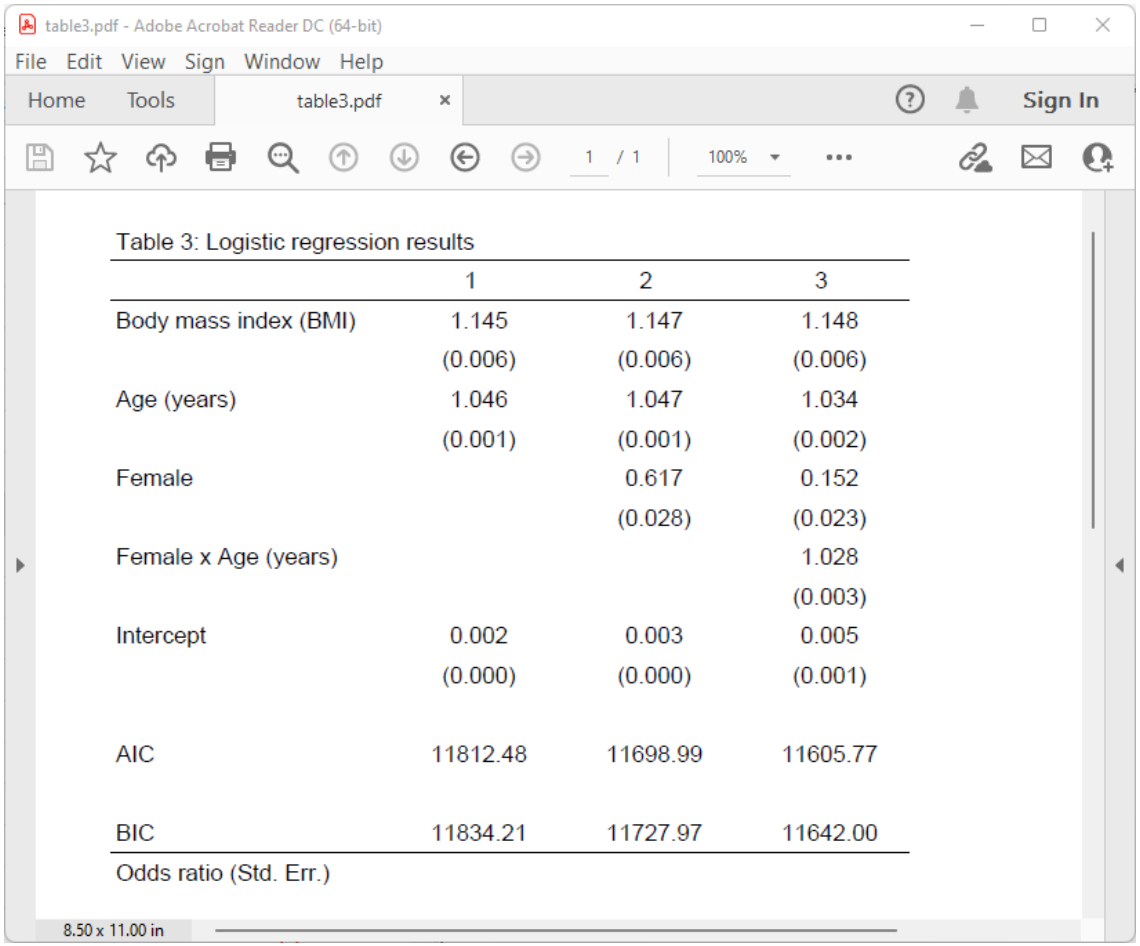


Table 3: Logistic regression results

	1	2	3
Body mass index (BMI)	1.145	1.147	1.148
	(0.006)	(0.006)	(0.006)
Age (years)	1.046	1.047	1.034
	(0.001)	(0.001)	(0.002)
Female		0.617	0.152
		(0.028)	(0.023)
Female x Age (years)			1.028
			(0.003)
Intercept	0.002	0.003	0.005
	(0.000)	(0.000)	(0.001)
AIC	11812.48	11698.99	11605.77
BIC	11834.21	11727.97	11642.00
Odds ratio (Std. Err.)			

With the `table`, `etable`, and `collect` commands, we can create tables of

- Summary statistics, including a classic Table 1
- Results of classical hypothesis tests
- Regression results
- Postestimation tests
- Combinations of the above
- Results returned by any Stata commands

We can customize tables by changing

- Table layout
- Numeric formats
- Labels appearing on rows and columns
- Stars and other added text
- Font type, size, and color
- Shading, borders, margins, alignment, and more

We can export the customized tables to

- Word
- Excel
- LaTeX
- PDF
- Markdown
- HTML
- SMCL
- Plain text

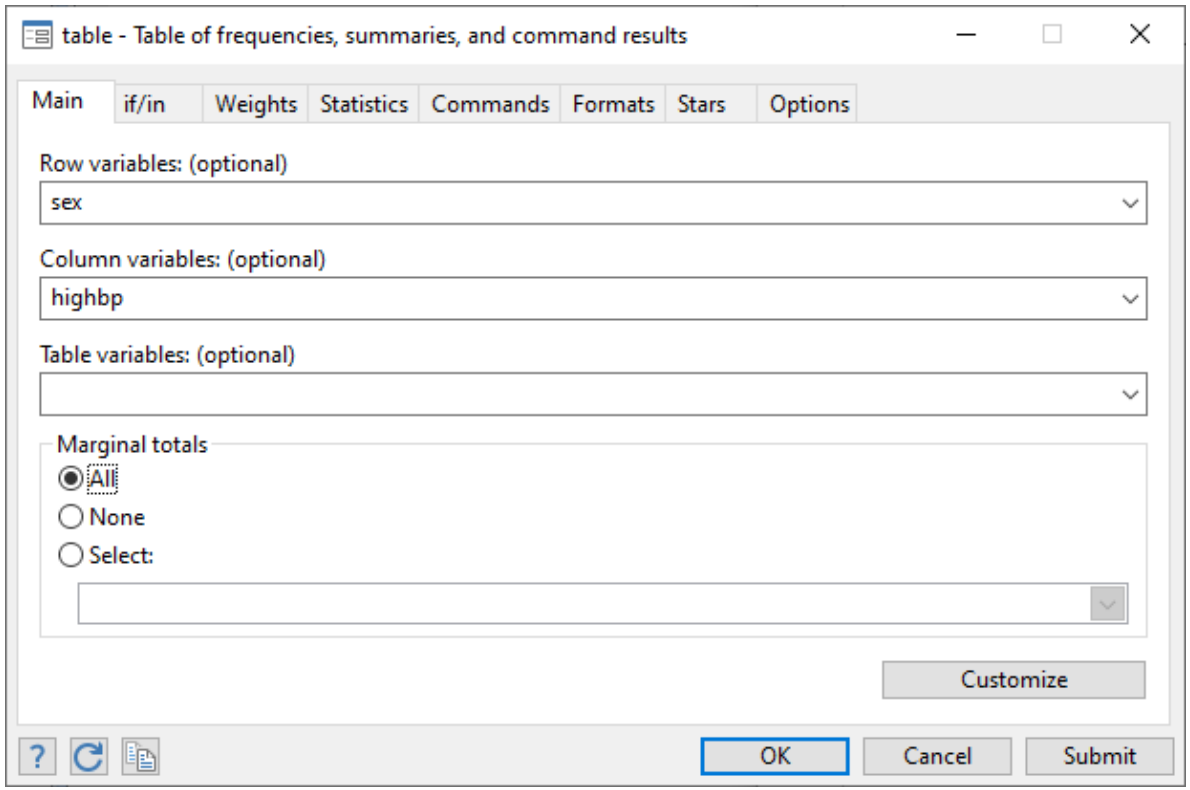
We can also include these tables in reports created by `putdocx`, `putpdf`, `putexcel`, and `dyndoc`.

Creating tables with the updated `table` command

The `table` command produces tabulations, tables of summary statistics, and tables of results from other Stata commands. To demonstrate, we will use NHANES II data.

```
. webuse nhanes2, clear
```

We will focus on the `table` command, but we could point and click to create the same tables with the `table` dialog box:



A simplified syntax for the `table` command is

. `table` (*row variables*) (*column variables*)

To create a one-way tabulation, we can specify a single row variable in the first set of parenthesis or a single column variable in the second set of parentheses.

```
. table (highbp) ()
```

	Frequency
High blood pressure	
0	5,975
1	4,376
Total	10,351

```
. table () (highbp)
```

	High blood pressure		
	0	1	Total
Frequency	5,975	4,376	10,351

To create a two-way tabulation, we specify both a row and a column variable.

```
. table (sex) (highbp)
```

	High blood pressure		
	0	1	Total
Sex			
Male	2,611	2,304	4,915
Female	3,364	2,072	5,436
Total	5,975	4,376	10,351

We can add options to our command to control what is displayed in the table. For instance, we can add the `nototals` option to remove the row frequencies, column frequencies, and overall frequencies from the tabulation.

```
. table (sex) (highbp), nototals
```

	High blood pressure	
	0	1
Sex		
Male	2,611	2,304
Female	3,364	2,072

Alternatively, we could request totals only for columns (highbp) or only for rows (sex).

```
. table (sex) (highbp), totals(highbp)
```

	High blood pressure	
	0	1
Sex		
Male	2,611	2,304
Female	3,364	2,072
Total	5,975	4,376

```
. table (sex) (highbp), totals(sex)
```

	High blood pressure		
	0	1	Total
Sex			
Male	2,611	2,304	4,915
Female	3,364	2,072	5,436

By default, the `table` command reports frequencies. Many additional summary statistics can be specified using the `statistic()` option.

```
. table ( row variables ) ( column variables ),
        statistic( statspec )
```

There are three variations of the `statistic()` option. The first, `statistic(`*freqstat*`)`, requests frequency statistics.

`frequency`	frequency
`sumw`	sum of weights

The second variation, `statistic(`*sumstat varlist*`)`, requests summary statistics for the variables in *varlist*.

`mean`	mean
`semean`	standard error of the mean
`sebinomial`	standard error of the mean, binomial
`sepoisson`	standard error of the mean, Poisson
`variance`	variance
`sd`	standard deviation
`skewness`	skewness
`kurtosis`	kurtosis
`cv`	coefficient of variation
`svycv`	coefficient of variation (svy)

count	number of values
median	median
p#	#th percentile
q1	first quartile
q2	second quartile
q3	third quartile
iqr	interquartile range
min	minimum value
max	maximum value
range	range
...	first, last, total, factor-variable proportion, more

The third variation, `statistic(`*ratiostat* [*varlist*]`)`, requests ratio statistics.

`proportion`	proportion
`percent`	percentage
`rawproportion`	proportion ignoring weights
`rawpercent`	percentage ignoring weights

We can specify the `statistic()` option repeatedly to display multiple statistics in a table. For example, we can request both frequencies and percentages for `highbp`.

```
. table () (highbp),
        statistic(frequency)
        statistic(percent)
```

	High blood pressure		
	0	1	Total
Frequency	5,975	4,376	10,351
Percent	57.72	42.28	100.00

We can also request multiple statistics for our two-way table. This time, we omit the totals.

```
. table (sex) (highbp),
        statistic(frequency)
        statistic(percent)
        nototals
```

	High blood pressure	
	0	1
Sex		
Male		
Frequency	2,611	2,304
Percent	25.22	22.26
Female		
Frequency	3,364	2,072
Percent	32.50	20.02

We might instead want to compare the means and standard deviations of some continuous variables, here age and bmi, across the levels of highbp.

```
. table () (highbp),
      statistic(mean age bmi)
      statistic(sd age bmi)
```

	High blood pressure		
	0	1	Total
Mean			
Age (years)	42.16502	54.97281	47.57965
Body mass index (BMI)	24.20231	27.36081	25.5376
Standard deviation			
Age (years)	16.77157	14.90897	17.21483
Body mass index (BMI)	4.100279	5.332119	4.914969

We can request different statistics for different variables.
In addition to the means and standard deviations of continuous variables, we can compare the frequencies and percentages of the factor variable `sex` across the levels of `highbp`.

```
. table () (highbp),
        statistic(mean age bmi)
        statistic(sd age bmi)
        statistic(fvfrequency sex)
        statistic(fvpercent sex)
```

	High blood pressure		
	0	1	Total
Mean			
Age (years)	42.16502	54.97281	47.57965
Body mass index (BMI)	24.20231	27.36081	25.5376
Standard deviation			
Age (years)	16.77157	14.90897	17.21483
Body mass index (BMI)	4.100279	5.332119	4.914969
Factor variable frequency			
Sex=Male	2,611	2,304	4,915
Sex=Female	3,364	2,072	5,436
Factor variable percent			
Sex=Male	43.70	52.65	47.48
Sex=Female	56.30	47.35	52.52

We can change the numeric format and the string format of the results in our table by using the `nformat()` and `sformat()` options.

```
. table ( row variables ) ( column variables ),
      nformat() sformat()
```

To request that the means and standard deviations are reported with two decimal places, we can add the `nformat(%5.2f mean sd)` option. We can add parentheses around the standard deviations with the `sformat("(%s)" sd)` option. Here the `%s` indicates where the statistic will be placed. We can add a percent sign following the percentages and parentheses around them by specifing the `sformat("(%s%%)" fvpercent)` option.

```
. table () (highbp),
        statistic(mean age bmi)
        statistic(sd age bmi)
        statistic(fvfrequency sex)
        statistic(fvpercent sex)
        nformat(%5.2f mean sd)
        sformat("(%s%%)" fvpercent)
        sformat("(%s)" sd)
```

	High blood pressure		
	0	1	Total
Mean			
Age (years)	42.17	54.97	47.58
Body mass index (BMI)	24.20	27.36	25.54
Standard deviation			
Age (years)	(16.77)	(14.91)	(17.21)
Body mass index (BMI)	(4.10)	(5.33)	(4.91)
Factor variable frequency			
Sex=Male	2,611	2,304	4,915
Sex=Female	3,364	2,072	5,436
Factor variable percent			
Sex=Male	(43.70%)	(52.65%)	(47.48%)
Sex=Female	(56.30%)	(47.35%)	(52.52%)

The `table` command also allows us to specify keywords in the row and column specifications to further control the layout of the table.

. `table` (*rowspec*) (*colspec*), *options*

`result`	requested statistics
`var`	variables from statistic() option
`across`	index across() specifications
`colname`	column names for matrix statistics
`rowname`	row names for matrix statistics
`coleq`	column equation names for matrix statistics
`roweq`	row equation names for matrix statistics
`command`	index option command()
`statcmd`	index options statistic() and command()

We modify our table so that the rows to are organized by variables (`var`) and then statistics (`result`) nested within variables.

```
. table (var result) (highbp),
        statistic(mean age bmi)
        statistic(sd age bmi)
        statistic(fvfrequency sex)
        statistic(fvpercent sex)
        nformat(%5.2f mean sd)
        sformat("(%s%%)" fvpercent)
        sformat("(%s)" sd)
```

	High blood pressure		
	0	1	Total
Age (years)			
Mean	42.17	54.97	47.58
Standard deviation	(16.77)	(14.91)	(17.21)
Body mass index (BMI)			
Mean	24.20	27.36	25.54
Standard deviation	(4.10)	(5.33)	(4.91)
Sex=Male			
Factor variable frequency	2,611	2,304	4,915
Factor variable percent	(43.70%)	(52.65%)	(47.48%)
Sex=Female			
Factor variable frequency	3,364	2,072	5,436
Factor variable percent	(56.30%)	(47.35%)	(52.52%)

The `table` command also allows us to create three-way and higher-way tables, include results from other Stata commands, add titles and notes, and more. Type

```
. help table
```

in Stata to see the full syntax of this command.

Creating estimation tables with the new etable command

After fitting any regression model—whether a linear regression, logistic regression, Cox proportional hazards model, multilevel model, or any other regression model available in Stata—we can use the `etable` command to create a table of regression results. We can also use `etable` to create a table comparing results from multiple models.

To create a table, we simply type `etable` after fitting a model to create the default table. To demonstrate, we first fit a logistic regression model.

```
. logistic highbp bmi age
Logistic regression                                     Number of obs =  10,351
                                                        LR chi2(2)    = 2295.05
                                                        Prob > chi2   =  0.0000
Log likelihood = -5903.2393                             Pseudo R2     =  0.1628
```

highbp	Odds ratio	Std. err.	z	P>|z|	[95% conf.	interval]
bmi	1.145469	.0058173	26.74	0.000	1.134124	1.156928
age	1.046239	.0014489	32.64	0.000	1.043403	1.049083
_cons	.002455	.000372	-39.66	0.000	.0018242	.0033038

Note: _cons estimates baseline odds.

Then we create our table of results.

```
. etable
------------------------------------
                              highbp
------------------------------------
Body mass index (BMI)          1.145
                              (0.006)
Age (years)                    1.046
                              (0.001)
Intercept                      0.002
                              (0.000)
Number of observations         10351
------------------------------------
```

By default, `etable` reports the coefficients or related statistics reported by the regression command (odds ratios in this case). The corresponding standard errors are reported in parentheses.

We can request that different statistics related to each coefficient be reported by specifying the `cstat()` option. The following statistics may be requested:

_r_b	coefficients reported by estimation
_r_se	standard errors of _r_b
_r_z	test statistics for _r_b
_r_z_abs	absolute values of _r_z
_r_p	*p*-values for _r_b
_r_lb	lower bounds of confidence intervals (CIs) for _r_b
_r_ub	upper bounds of CIs for _r_b
_r_ci	CIs for _r_b
_r_crlb	lower bounds of credible intervals for _r_b
_r_crub	upper bounds of credible intervals for _r_b
_r_cri	credible intervals of Bayesian estimates

Instead of reporting odds ratios and standard errors, perhaps we want to report odds ratios and confidence intervals. We type

```
. etable, cstat(_r_b) cstat(_r_ci)
```

	highbp
Body mass index (BMI)	1.145
	[1.134 1.157]
Age (years)	1.046
	[1.043 1.049]
Intercept	0.002
	[0.002 0.003]
Number of observations	10351

Each result is specified separately to allow different formatting options. Similar to `table`, we can specify an `nformat()` and `sformat()` for each result. We can also specify the delimiter to be used between the upper and lower bound of the confidence interval with `cidelimiter()`.

```
. etable, cstat(_r_b) cstat(_r_ci, cidelimiter(", ") nformat(%4.2f))
```

	highbp
Body mass index (BMI)	1.145
	[1.13, 1.16]
Age (years)	1.046
	[1.04, 1.05]
Intercept	0.002
	[0.00, 0.00]
Number of observations	10351

We could request that the *p*-value be displayed alongside each odds ratio. However, another alternative is to request that stars corresponding to specific *p*-value levels be displayed.

```
. etable, showstars showstarsnote
------------------------------------
                             highbp
------------------------------------
Body mass index (BMI)       1.145 **
                           (0.006)
Age (years)                 1.046 **
                           (0.001)
Intercept                   0.002 **
                           (0.000)
Number of observations      10351
------------------------------------
** p<.01, * p<.05
```

We can also request a variety of overall model statistics by specifying the `mstat()` option. The following statistics may be requested:

N	number of observations
aic	Akaike's information criteria
bic	Schwarz's Bayesian information criteria
F	*F* statistic
chi2	chi-squared
ll	log likelihood of fitted model
r2	*R*-squared
r2_a	adjusted *R*-squared
rank	rank of fitted model
scalar	any e() scalar

Rather than the number of observations, we request that AIC and BIC be reported so that we can later compare this model to others.

```
. etable, mstat(aic) mstat(bic)
------------------------------------
                            highbp
------------------------------------
Body mass index (BMI)        1.145
                            (0.006)
Age (years)                  1.046
                            (0.001)
Intercept                    0.002
                            (0.000)
AIC                      11812.48
BIC                      11834.21
------------------------------------
```

Suppose we want to compare the model we fit above to two other models. We can build our table in two ways. The first is to use the append option with etable after fitting the second and third models.

We begin by fitting the first model and using etable just as we did previously.

```
. logistic highbp bmi age
Logistic regression                                     Number of obs =  10,351
                                                        LR chi2(2)    = 2295.05
                                                        Prob > chi2   =  0.0000
Log likelihood = -5903.2393                             Pseudo R2     =  0.1628
```

highbp	Odds ratio	Std. err.	z	P>|z|	[95% conf.	interval]
bmi	1.145469	.0058173	26.74	0.000	1.134124	1.156928
age	1.046239	.0014489	32.64	0.000	1.043403	1.049083
_cons	.002455	.000372	-39.66	0.000	.0018242	.0033038

Note: _cons estimates baseline odds.

```
. etable, mstat(aic) mstat(bic)
------------------------------------
                             highbp
------------------------------------
Body mass index (BMI)         1.145
                             (0.006)
Age (years)                   1.046
                             (0.001)
Intercept                     0.002
                             (0.000)
AIC                          11812.48
BIC                          11834.21
------------------------------------
```

Next, we fit our second model, which includes sex as a regressor, and then specify the append option with etable.

```
. logistic highbp bmi age i.sex
Logistic regression                             Number of obs =  10,351
                                                LR chi2(3)    = 2410.54
                                                Prob > chi2   =  0.0000
Log likelihood = -5845.4948                     Pseudo R2     =  0.1709
```

highbp	Odds ratio	Std. err.	z	P>|z|	[95% conf.	interval]
bmi	1.147006	.005827	27.00	0.000	1.135642	1.158484
age	1.047058	.0014632	32.91	0.000	1.044194	1.04993
sex						
Female	.6172715	.0278625	-10.69	0.000	.5650079	.6743695
_cons	.0029059	.0004428	-38.33	0.000	.0021556	.0039174

Note: _cons estimates baseline odds.

```
. etable, append
```

	highbp	highbp
Body mass index (BMI)	1.145	1.147
	(0.006)	(0.006)
Age (years)	1.046	1.047
	(0.001)	(0.001)
Sex		
Female		0.617
		(0.028)
Intercept	0.002	0.003
	(0.000)	(0.000)
AIC	11812.48	11698.99
BIC	11834.21	11727.97

Similarly, we fit our third model, which includes an interaction between age and sex, and again specify append with etable.

```
. logistic highbp bmi c.age##i.sex
Logistic regression                             Number of obs =  10,351
                                                LR chi2(4)    = 2505.76
                                                Prob > chi2   =  0.0000
Log likelihood = -5797.8856                     Pseudo R2     =  0.1777
```

highbp	Odds ratio	Std. err.	z	P>|z|	[95% conf.	interval]
bmi	1.147615	.0058948	26.80	0.000	1.136119	1.159226
age	1.034097	.0019132	18.12	0.000	1.030354	1.037854
sex						
Female	.1522122	.0232902	-12.30	0.000	.1127732	.2054439
sex#c.age						
Female	1.028	.0029402	9.66	0.000	1.022253	1.033779
_cons	.0052605	.0008567	-32.22	0.000	.003823	.0072384

Note: _cons estimates baseline odds.

```
. etable, append
```

	highbp	highbp	highbp
Body mass index (BMI)	1.145	1.147	1.148
	(0.006)	(0.006)	(0.006)
Age (years)	1.046	1.047	1.034
	(0.001)	(0.001)	(0.002)
Sex			
Female		0.617	0.152
		(0.028)	(0.023)
Sex # Age (years)			
Female			1.028
			(0.003)
Intercept	0.002	0.003	0.005
	(0.000)	(0.000)	(0.001)
AIC	11812.48	11698.99	11605.77
BIC	11834.21	11727.97	11642.00

Finally, we modify our table by adding a title, notes, and numbers for the column headings. We specify the `replay` option to make these modifications to the existing table.

```
. etable, replay column(index)
          title("Table 2: Logistic regression results")
          notes("Odds ratio (Std. Err.)")
```

Table 2: Logistic regression results

	1	2	3
Body mass index (BMI)	1.145	1.147	1.148
	(0.006)	(0.006)	(0.006)
Age (years)	1.046	1.047	1.034
	(0.001)	(0.001)	(0.002)
Sex			
Female		0.617	0.152
		(0.028)	(0.023)
Sex # Age (years)			
Female			1.028
			(0.003)
Intercept	0.002	0.003	0.005
	(0.000)	(0.000)	(0.001)
AIC	11812.48	11698.99	11605.77
BIC	11834.21	11727.97	11642.00

Odds ratio (Std. Err.)

We could have fit all three models and then created the same table with a single etable command as follows:

```
. logistic highbp bmi age
. estimates store M1
. logistic highbp bmi age i.sex
. estimates store M2
. logistic highbp bmi c.age##i.age
. estimates store M3
. etable, estimates(M1 M2 M3) mstat(aic) mstat(bic)
          title("Table 2: Logistic regression results")
          notes("Odds ratio (Std. Err.)")
```

The `etable` command also allows us to create tables of marginal means and marginal effects produced by `margins` and more. Type

```
. help etable
```

in Stata to see the full syntax of this command.

Creating and further customizing tables with the collect suit of commands

- collect is a suite of commands designed to collect results from various Stata commands; place the selected results into a table; customize the labels, formats, and style; and export the final table to Word, Excel, LaTeX, PDF, Markdown, HTML, and more.
- Because the table and etable commands are based on the collect system, we can also use the collect commands to further modify tables originally created by one of these commands.

The basic workflow for collecting results and building tables is

- Collect results from Stata commands—`collect` command, `collect:` prefix, `table`, or `etable`
- Explore the collection—`collect dims`, `collect levelsof`, and `collect label list`
- Define the rows and columns of the table—`collect layout`, `table`, or `etable`
- Modify items within the collection—`collect composite`, `collect recode`, `collect remap`, and `collect addtags`

- Customize the table, specifying formats, labels, font, shading, and more—`collect label`, `collect style`, `collect stars`, ...
- Export the table to Word, Excel, LaTeX, PDF, Markdown, HTML, SMCL, or plain text—`collect export`
- Save the style, labels, and collection to use and modify later—`collect label save`, `collect style save`, and `collect save`

Suppose we want to create a table that displays the results from *t* tests performed using the `ttest` command. We first take a look at the results that this command returns by typing `return list`.

```
. ttest age, by(highbp)
Two-sample t test with equal variances
```

Group	Obs	Mean	Std. err.	Std. dev.	[95% conf.	interval]
0	5,975	42.16502	.2169725	16.77157	41.73968	42.59037
1	4,376	54.97281	.2253767	14.90897	54.53095	55.41466
Combined	10,351	47.57965	.1692044	17.21483	47.24798	47.91133
diff		-12.80779	.3185604		-13.43223	-12.18335

```
    diff = mean(0) - mean(1)                                  t = -40.2052
H0: diff = 0                     Degrees of freedom =    10349

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.0000         Pr(|T| > |t|) = 0.0000          Pr(T > t) = 1.0000
```

```
. return list
scalars:
               r(level) =  95
                  r(sd) =  17.21482923023818
                r(sd_2) =  14.9089715191102
                r(sd_1) =  16.77156676799842
                  r(se) =  .3185603831285
                 r(p_u) =  1
                 r(p_l) =  0
                   r(p) =  0
                   r(t) =  -40.20520433030012
                r(df_t) =  10349
                r(mu_2) =  54.97280621572212
                 r(N_2) =  4376
                r(mu_1) =  42.16502092050209
                 r(N_1) =  5975
```

Before collecting results, we use `collect clear` to make sure any previous results are cleared from the default collection. Then we are ready to store results in the collection. We do this with the `collect` prefix. We want our table to include the mean of each group (stored in `r(mu_1)` and `r(mu_2)`) as well as the *p*-value (stored in `r(p)`).

```
. collect clear
. collect r(mu_1) r(mu_2) r(p): ttest age, by(highbp)
Two-sample t test with equal variances
```

Group	Obs	Mean	Std. err.	Std. dev.	[95% conf.	interval]
0	5,975	42.16502	.2169725	16.77157	41.73968	42.59037
1	4,376	54.97281	.2253767	14.90897	54.53095	55.41466
Combined	10,351	47.57965	.1692044	17.21483	47.24798	47.91133
diff		-12.80779	.3185604		-13.43223	-12.18335

```
    diff = mean(0) - mean(1)                                  t = -40.2052
H0: diff = 0                     Degrees of freedom =    10349

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.0000         Pr(|T| > |t|) = 0.0000          Pr(T > t) = 1.0000
```

```
. collect r(mu_1) r(mu_2) r(p): ttest weight, by(highbp)
Two-sample t test with equal variances
```

Group	Obs	Mean	Std. err.	Std. dev.	[95% conf.	interval]
0	5,975	68.26626	.1755737	13.57152	67.92207	68.61045
1	4,376	76.85565	.2455179	16.24134	76.37431	77.33699
Combined	10,351	71.89752	.1509381	15.35642	71.60165	72.19339
diff		-8.589386	.2936615		-9.165019	-8.013753

```
    diff = mean(0) - mean(1)                                      t = -29.2493
H0: diff = 0                                     Degrees of freedom =    10349

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.0000         Pr(|T| > |t|) = 0.0000          Pr(T > t) = 1.0000
```

```
. collect r(mu_1) r(mu_2) r(p): ttest bmi, by(highbp)
Two-sample t test with equal variances
```

Group	Obs	Mean	Std. err.	Std. dev.	[95% conf.	interval]
0	5,975	24.20231	.053045	4.100279	24.09832	24.30629
1	4,376	27.36081	.0806049	5.332119	27.20279	27.51884
Combined	10,351	25.5376	.0483092	4.914969	25.4429	25.63229
diff		-3.158506	.0927376		-3.34029	-2.976723

```
    diff = mean(0) - mean(1)                                  t = -34.0585
H0: diff = 0                     Degrees of freedom =    10349

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.0000         Pr(|T| > |t|) = 0.0000          Pr(T > t) = 1.0000
```

Values in a collection are organized according to their *tags*. A tag consists of a *dimension* and a *level* within the dimension. A tag is written as `dimension[level]`. Consider the means and *p*-value from the previous `ttest` command.

Value	Tag 1	Tag 2
24.202	`cmdset[3]`	`result[mu_1]`
27.361	`cmdset[3]`	`result[mu_2]`
0.000	`cmdset[3]`	`result[p]`

Our collection includes dimensions `cmdset` and `result`, among others. These results all come from the third `ttest` command, so they have level 3 of dimension `cmdset`. The level of dimension `result` corresponds to the name in the `r()` result.

How do we find out what is in the collection? We can use `collect dims` to see a list of the dimensions.

```
. collect dims

Collection dimensions
Collection: default
```

Dimension	No. levels
Layout, style, header, label	
cmdset	3
program_class	1
result	14
result_type	1
Style only	
border_block	4
cell_type	4

We can see the levels of a specified dimension with `collect levelsof` or `collect label list`. We add option `all` to see the levels that are not labeled in addition to those that are.

```
. collect label list result, all

  Collection: default
   Dimension: result
       Label: Result
Level labels:
         N_1  Sample size n1
         N_2  Sample size n2
        df_t  Degrees of freedom
       level  Confidence level
        mu_1  x1 mean for population 1
        mu_2  x2 mean for population 2
           p  Two-sided p-value
         p_l  Lower one-sided p-value
         p_u  Upper one-sided p-value
          sd  Combined std. dev.
        sd_1  Standard deviation for first variable
        sd_2  Standard deviation for second variable
          se  Std. error
           t  t statistic
```

```
. collect label list cmdset, all

  Collection: default
   Dimension: cmdset
       Label: Command results index
Level labels:
             1
             2
             3
```

The levels of the cmdset dimension correspond to our three ttest commands. The levels of the result dimension correspond to the statistics. We use the dimension names to specify a table layout with commands on the rows and statistics on the columns by typing

```
. collect layout (cmdset) (result)
Collection: default
      Rows: cmdset
   Columns: result
   Table 1: 3 x 3
```

	x1 mean for population 1	x2 mean for population 2	Two-sided p-value
1	42.16502	54.97281	0
2	68.26626	76.85565	9.1e-181
3	24.20231	27.36081	4.5e-241

We could label the values of cmdset to give more information about the rows. Another option is to create meaningful tags when we collect results.

Here, we use the tags() option with the collect prefix to create a new dimension named varname. Its levels correspond to the names of the variables specified in ttest.

```
. collect clear
. collect r(mu_1) r(mu_2) r(p), tags(varname[age]): ttest age, by(highbp)
Two-sample t test with equal variances
```

Group	Obs	Mean	Std. err.	Std. dev.	[95% conf.	interval]
0	5,975	42.16502	.2169725	16.77157	41.73968	42.59037
1	4,376	54.97281	.2253767	14.90897	54.53095	55.41466
Combined	10,351	47.57965	.1692044	17.21483	47.24798	47.91133
diff		-12.80779	.3185604		-13.43223	-12.18335

```
    diff = mean(0) - mean(1)                                  t = -40.2052
H0: diff = 0                     Degrees of freedom =    10349

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.0000         Pr(|T| > |t|) = 0.0000          Pr(T > t) = 1.0000
```

```
. collect r(mu_1) r(mu_2) r(p), tags(varname[weight]): ttest weight, by(highbp)
Two-sample t test with equal variances
```

Group	Obs	Mean	Std. err.	Std. dev.	[95% conf.	interval]
0	5,975	68.26626	.1755737	13.57152	67.92207	68.61045
1	4,376	76.85565	.2455179	16.24134	76.37431	77.33699
Combined	10,351	71.89752	.1509381	15.35642	71.60165	72.19339
diff		-8.589386	.2936615		-9.165019	-8.013753

```
    diff = mean(0) - mean(1)                                      t = -29.2493
H0: diff = 0                                     Degrees of freedom =    10349

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.0000         Pr(|T| > |t|) = 0.0000          Pr(T > t) = 1.0000
```

```
. collect r(mu_1) r(mu_2) r(p), tags(varname[bmi]): ttest bmi, by(highbp)
Two-sample t test with equal variances
```

Group	Obs	Mean	Std. err.	Std. dev.	[95% conf.	interval]
0	5,975	24.20231	.053045	4.100279	24.09832	24.30629
1	4,376	27.36081	.0806049	5.332119	27.20279	27.51884
Combined	10,351	25.5376	.0483092	4.914969	25.4429	25.63229
diff		-3.158506	.0927376		-3.34029	-2.976723

```
    diff = mean(0) - mean(1)                                  t = -34.0585
H0: diff = 0                     Degrees of freedom =    10349

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.0000         Pr(|T| > |t|) = 0.0000          Pr(T > t) = 1.0000
```

We see that the new varname dimension has been added to the collection.

```
. collect dims

Collection dimensions
Collection: default
```

Dimension	No. levels
Layout, style, header, label	
cmdset	3
program_class	1
result	14
result_type	1
varname	3
Style only	
border_block	4
cell_type	4

The levels of this dimension are age, bmi, and weight, as we requested.

```
. collect label list varname, all

  Collection: default
   Dimension: varname
       Label:
Level labels:
         age
         bmi
      weight
```

We can now specify that the varname dimension be used to construct the rows of our table.

```
. collect layout (varname) (result)
Collection: default
      Rows: varname
   Columns: result
   Table 1: 3 x 3
```

	x1 mean for population 1	x2 mean for population 2	Two-sided p-value
age	42.16502	54.97281	0
weight	68.26626	76.85565	9.1e-181
bmi	24.20231	27.36081	4.5e-241

We now have more meaningful labels on the rows, but the column headers have default labels that we might want to modify.

We can use the `collect label` command to specify our preferred labels for the levels of the `result` dimension.

```
. collect label levels result
       mu_1 "Normotensive" mu_2 "Hypertensive" p "p-value", modify
. collect preview
```

	Normotensive	Hypertensive	p-value
age	42.16502	54.97281	0
weight	68.26626	76.85565	9.1e-181
bmi	24.20231	27.36081	4.5e-241

`collect style cell` allows us to make a wide variety of formatting modifications to our table. First, we modify the numeric format of the *p*-values, and we request that numbers less than 0.001 be reported as <0.001 in the table.

```
. collect style cell result[p], nformat(%5.3f) min(0.001)
. collect preview
```

	Normotensive	Hypertensive	p-value
age	42.16502	54.97281	<0.001
weight	68.26626	76.85565	<0.001
bmi	24.20231	27.36081	<0.001

We also request that only two decimal places be reported for the means.

```
. collect style cell result[mu_1 mu_2], nformat(%5.2f)
. collect preview
```

	Normotensive	Hypertensive	p-value
age	42.17	54.97	<0.001
weight	68.27	76.86	<0.001
bmi	24.20	27.36	<0.001

We specify that the values (cell_type[item]) be centered horizontally within in the columns.

```
. collect style cell cell_type[item], halign(center)
. collect preview
```

	Normotensive	Hypertensive	p-value
age	42.17	54.97	<0.001
weight	68.27	76.86	<0.001
bmi	24.20	27.36	<0.001

Finally, we request that the border be removed from the right side of the row headers.

```
. collect style cell border_block, border(right, pattern(nil))
. collect preview
```

	Normotensive	Hypertensive	p-value
age	42.17	54.97	<0.001
weight	68.27	76.86	<0.001
bmi	24.20	27.36	<0.001

Once the table is formatted to our liking, we can export to a number of different formats by specifying the appropriate suffix. Below, we export the table to HTML.

```
. collect export ttest_tab.html, replace
(collection default exported to file ttest_tab.html)
```

If we create tables of a similar style repeatedly, there is no need to remember the `collect` commands each time we create a new table. Instead, we can save the style we like and apply it to tables we create in the future.

```
. collect style save ttest_style, replace
(style from default saved to file ttest_style.stjson)
```

Now, we can apply this style to tables using different variables and even different datasets.

```
. clear all
. sysuse auto, clear
. collect r(mu_1) r(mu_2) r(p), tags(varname[price]): ttest price, by(foreign)
. collect r(mu_1) r(mu_2) r(p), tags(varname[weight]): ttest weight, by(foreign)
. collect r(mu_1) r(mu_2) r(p), tags(varname[length]): ttest length, by(foreign)
. collect style use ttest_style, replace
. collect preview
```

	x1 mean for population 1	x2 mean for population 2	Two-sided p-value
price	6072.42	6384.68	0.680
weight	3317.12	2315.91	<0.001
length	196.13	168.55	<0.001

Now, we need only to make modifications specific to this dataset, such as the labels in the column headers.

```
. collect label levels result
       mu_1 "Domestic cars" mu_2 "Foreign cars" p "p-value", modify
. collect preview
```

	Domestic cars	Foreign cars	p-value
price	6072.42	6384.68	0.680
weight	3317.12	2315.91	<0.001
length	196.13	168.55	<0.001

As we have seen previously, `table` and `etable` make it easy to create tables. They have many options for formatting, but when we don't get exactly what we want from one of these commands, we can often use just a few `collect` commands to finish the customization.

Modifying table results with collect

Below, we create a table command similar to our previous table of summary statistics, but with a few additional variables.

```
. webuse nhanes2l, clear
(Second National Health and Nutrition Examination Survey)
. table (var) (highbp),
        statistic(mean age bmi)
        statistic(sd age bmi)
        statistic(fvfrequency sex race hlthstat)
        statistic(fvpercent sex race hlthstat)
        statistic(mean tcresult tgresult hdresult)
        statistic(sd tcresult tgresult hdresult)
        nformat(%5.2f mean sd)
        sformat("(%s%%)" fvpercent)
        sformat("(%s)" sd)
```

	High blood pressure		
	0	1	Total
Age (years)			
Mean	42.17	54.97	47.58
Standard deviation	(16.77)	(14.91)	(17.21)
Body mass index (BMI)			
Mean	24.20	27.36	25.54
Standard deviation	(4.10)	(5.33)	(4.91)
Sex=Male			
Factor variable frequency	2,611	2,304	4,915
Factor variable percent	(43.70%)	(52.65%)	(47.48%)
Sex=Female			
Factor variable frequency	3,364	2,072	5,436
Factor variable percent	(56.30%)	(47.35%)	(52.52%)
Race=White			
Factor variable frequency	5,317	3,748	9,065
Factor variable percent	(88.99%)	(85.65%)	(87.58%)
Race=Black			
Factor variable frequency	545	541	1,086
Factor variable percent	(9.12%)	(12.36%)	(10.49%)
Race=Other			
Factor variable frequency	113	87	200
Factor variable percent	(1.89%)	(1.99%)	(1.93%)

Health status=Excellent			
Factor variable frequency	1,649	758	2,407
Factor variable percent	(27.65%)	(17.34%)	(23.29%)
Health status=Very good			
Factor variable frequency	1,666	925	2,591
Factor variable percent	(27.94%)	(21.16%)	(25.07%)
Health status=Good			
Factor variable frequency	1,572	1,366	2,938
Factor variable percent	(26.36%)	(31.24%)	(28.43%)
Health status=Fair			
Factor variable frequency	766	904	1,670
Factor variable percent	(12.85%)	(20.68%)	(16.16%)
Health status=Poor			
Factor variable frequency	310	419	729
Factor variable percent	(5.20%)	(9.58%)	(7.05%)
Serum cholesterol (mg/dL)			
Mean	208.73	229.88	217.67
Standard deviation	(47.29)	(49.58)	(49.39)
Serum triglycerides (mg/dL)			
Mean	129.23	166.04	143.90
Standard deviation	(83.93)	(109.20)	(96.50)
High density lipids (mg/dL)			
Mean	49.94	49.22	49.64
Standard deviation	(14.14)	(14.54)	(14.31)

Rather than having means and standard deviations in separate rows, suppose we want them to be side by side in a single column of the table. Likewise, we want frequencies and percentages for factor variables to be side by side.

We use `collect composite` to create a result named `mystat` that combines the available statistics for each variable. We then specify this new result in our table layout.

```
. collect composite define mystat = mean sd fvfrequency fvpercent, trim
. collect style header result, level(hide)
```

```
. collect layout (var) (highbp#result[mystat])
Collection: Table
      Rows: var
   Columns: highbp#result[mystat]
   Table 1: 15 x 3
```

	High blood pressure		
	0	1	Total
Age (years)	42.17 (16.77)	54.97 (14.91)	47.58 (17.21)
Body mass index (BMI)	24.20 (4.10)	27.36 (5.33)	25.54 (4.91)
Sex=Male	2,611 (43.70%)	2,304 (52.65%)	4,915 (47.48%)
Sex=Female	3,364 (56.30%)	2,072 (47.35%)	5,436 (52.52%)
Race=White	5,317 (88.99%)	3,748 (85.65%)	9,065 (87.58%)
Race=Black	545 (9.12%)	541 (12.36%)	1,086 (10.49%)
Race=Other	113 (1.89%)	87 (1.99%)	200 (1.93%)
Health status=Excellent	1,649 (27.65%)	758 (17.34%)	2,407 (23.29%)
Health status=Very good	1,666 (27.94%)	925 (21.16%)	2,591 (25.07%)
Health status=Good	1,572 (26.36%)	1,366 (31.24%)	2,938 (28.43%)
Health status=Fair	766 (12.85%)	904 (20.68%)	1,670 (16.16%)
Health status=Poor	310 (5.20%)	419 (9.58%)	729 (7.05%)
Serum cholesterol (mg/dL)	208.73 (47.29)	229.88 (49.58)	217.67 (49.39)
Serum triglycerides (mg/dL)	129.23 (83.93)	166.04 (109.20)	143.90 (96.50)
High density lipids (mg/dL)	49.94 (14.14)	49.22 (14.54)	49.64 (14.31)

Rather than seeing Sex=Female and Sex=Male, say that we want to see Female and Male listed underneath Race. To make this change, we request a stacked row style. We also specify nobinder to remove the equal sign and spacer to add some vertical space in the table.

```
. collect style row stack, nobinder spacer
```

```
. collect preview
```

	High blood pressure		
	0	1	Total
Age (years)	42.17 (16.77)	54.97 (14.91)	47.58 (17.21)
Body mass index (BMI)	24.20 (4.10)	27.36 (5.33)	25.54 (4.91)
Sex			
Male	2,611 (43.70%)	2,304 (52.65%)	4,915 (47.48%)
Female	3,364 (56.30%)	2,072 (47.35%)	5,436 (52.52%)
Race			
White	5,317 (88.99%)	3,748 (85.65%)	9,065 (87.58%)
Black	545 (9.12%)	541 (12.36%)	1,086 (10.49%)
Other	113 (1.89%)	87 (1.99%)	200 (1.93%)
Health status			
Excellent	1,649 (27.65%)	758 (17.34%)	2,407 (23.29%)
Very good	1,666 (27.94%)	925 (21.16%)	2,591 (25.07%)
Good	1,572 (26.36%)	1,366 (31.24%)	2,938 (28.43%)
Fair	766 (12.85%)	904 (20.68%)	1,670 (16.16%)
Poor	310 (5.20%)	419 (9.58%)	729 (7.05%)
Serum cholesterol (mg/dL)	208.73 (47.29)	229.88 (49.58)	217.67 (49.39)
Serum triglycerides (mg/dL)	129.23 (83.93)	166.04 (109.20)	143.90 (96.50)
High density lipids (mg/dL)	49.94 (14.14)	49.22 (14.54)	49.64 (14.31)

Next, we modify the labels on the dimension highbp and its levels.

```
. collect label dim highbp "Hypertension", modify
. collect label levels highbp 0 "No" 1 "Yes"
```

```
. collect preview
```

	Hypertension		
	No	Yes	Total
Age (years)	42.17 (16.77)	54.97 (14.91)	47.58 (17.21)
Body mass index (BMI)	24.20 (4.10)	27.36 (5.33)	25.54 (4.91)
Sex			
Male	2,611 (43.70%)	2,304 (52.65%)	4,915 (47.48%)
Female	3,364 (56.30%)	2,072 (47.35%)	5,436 (52.52%)
Race			
White	5,317 (88.99%)	3,748 (85.65%)	9,065 (87.58%)
Black	545 (9.12%)	541 (12.36%)	1,086 (10.49%)
Other	113 (1.89%)	87 (1.99%)	200 (1.93%)
Health status			
Excellent	1,649 (27.65%)	758 (17.34%)	2,407 (23.29%)
Very good	1,666 (27.94%)	925 (21.16%)	2,591 (25.07%)
Good	1,572 (26.36%)	1,366 (31.24%)	2,938 (28.43%)
Fair	766 (12.85%)	904 (20.68%)	1,670 (16.16%)
Poor	310 (5.20%)	419 (9.58%)	729 (7.05%)
Serum cholesterol (mg/dL)	208.73 (47.29)	229.88 (49.58)	217.67 (49.39)
Serum triglycerides (mg/dL)	129.23 (83.93)	166.04 (109.20)	143.90 (96.50)
High density lipids (mg/dL)	49.94 (14.14)	49.22 (14.54)	49.64 (14.31)

We then add notes and a title to finish our table.

```
. collect notes 1: Mean and standard deviation reported for continuous variables.
. collect notes 2: Frequency and percentage reported for categorical variables.
. collect title "Table 1: Summary statistics"
```

Table 1: Summary statistics

	Hypertension		
	No	Yes	Total
Age (years)	42.17 (16.77)	54.97 (14.91)	47.58 (17.21)
Body mass index (BMI)	24.20 (4.10)	27.36 (5.33)	25.54 (4.91)
Sex			
Male	2,611 (43.70%)	2,304 (52.65%)	4,915 (47.48%)
Female	3,364 (56.30%)	2,072 (47.35%)	5,436 (52.52%)
Race			
White	5,317 (88.99%)	3,748 (85.65%)	9,065 (87.58%)
Black	545 (9.12%)	541 (12.36%)	1,086 (10.49%)
Other	113 (1.89%)	87 (1.99%)	200 (1.93%)
Health status			
Excellent	1,649 (27.65%)	758 (17.34%)	2,407 (23.29%)
Very good	1,666 (27.94%)	925 (21.16%)	2,591 (25.07%)
Good	1,572 (26.36%)	1,366 (31.24%)	2,938 (28.43%)
Fair	766 (12.85%)	904 (20.68%)	1,670 (16.16%)
Poor	310 (5.20%)	419 (9.58%)	729 (7.05%)
Serum cholesterol (mg/dL)	208.73 (47.29)	229.88 (49.58)	217.67 (49.39)
Serum triglycerides (mg/dL)	129.23 (83.93)	166.04 (109.20)	143.90 (96.50)
High density lipids (mg/dL)	49.94 (14.14)	49.22 (14.54)	49.64 (14.31)

Mean and standard deviation reported for continuous variables.
Frequency and percentage reported for categorical variables.

Modifying etable results with collect

Next we put the finishing touches on our table of logistic regression results.

```
. quietly logistic highbp bmi age
. quietly etable
. quietly logistic highbp bmi age i.sex
. quietly etable, append
. quietly logistic highbp bmi c.age##i.sex
```

```
. etable, append column(index)
          mstat(aic) mstat(bic)
          title("Table 2: Logistic regression results")
          notes("Odds ratio (Std. Err.)")
```

Table 2: Logistic regression results

	1	2	3
Body mass index (BMI)	1.145	1.147	1.148
	(0.006)	(0.006)	(0.006)
Age (years)	1.046	1.047	1.034
	(0.001)	(0.001)	(0.002)
Sex			
Female		0.617	0.152
		(0.028)	(0.023)
Sex # Age (years)			
Female			1.028
			(0.003)
Intercept	0.002	0.003	0.005
	(0.000)	(0.000)	(0.001)
AIC	11812.48	11698.99	11605.77
BIC	11834.21	11727.97	11642.00

Odds ratio (Std. Err.)

Because sex is a binary variable, we do not need the variable label and the level labels. We hide the title for this dimension. We also specify a split row style with an x for the interaction delimiter instead of #. We again add some vertical space to the table.

```
. collect style header sex, title(hide)
. collect style row split, dups(first) delimiter(" x ") spacer
. collect preview
Table 2: Logistic regression results
```

	1	2	3
Body mass index (BMI)	1.145	1.147	1.148
	(0.006)	(0.006)	(0.006)
Age (years)	1.046	1.047	1.034
	(0.001)	(0.001)	(0.002)
Female		0.617	0.152
		(0.028)	(0.023)
Female x Age (years)			1.028
			(0.003)
Intercept	0.002	0.003	0.005
	(0.000)	(0.000)	(0.001)
AIC	11812.48	11698.99	11605.77
BIC	11834.21	11727.97	11642.00

```
Odds ratio (Std. Err.)
```

We now have clean tables that can be used on their own or included in complete reports.

Summary

- The `table` command can now easily create and format tabulations, tables of summary statistics, and tables of results from other Stata commands.
- The `etable` command creates, customizes, and exports tables of regression results.
- The `collect` suite is allows for building even more complex tables as well as customizing those tables and exporting them to many formats.
- Customized tables can also be included in complete reports.
- Saving styles and labels allows you to easily apply your desired customizations to tables you create in the future.

Learn more

- `table`

  https://www.stata.com/manuals/rtableintro.pdf
- `etable`

  https://www.stata.com/manuals/retable.pdf
- `collect`

  https://www.stata.com/manuals/tables.pdf