

Data Processing

with Stata 14.1 Cheat Sheet

For more info see Stata's reference manual (stata.com)

Useful Shortcuts

F2 — keyboard buttons

describe data

Ctrl + **9**

open a new .do file

Ctrl + **8**

open the data editor

Ctrl + **D**

highlight text in .do file, then **ctrl + d** executes it in the command line

clear

delete data in memory

AT COMMAND PROMPT

PgUp

PgDn

scroll through previous commands

Tab

autocompletes variable name after typing part

cls

clear the console (where results are displayed)

Set up

pwd

print current (working) directory

cd "C:\Program Files (x86)\Stata13"

change working drive

dir

display filenames in working directory

fs *.dta

List all Stata files in working directory

underlined parts are shortcuts – use "capture" or "cap"

capture log close

close the log on any existing do files

log using "myDoFile.do", replace

create a new log file to record your work and results

search mdesc

find the package mdesc to install

packages contain extra commands that expand Stata's toolkit

ssc install mdesc

install the package mdesc; needs to be done once

Import Data

sysuse auto, clear

load system data (Auto data) **for many examples, we use the auto dataset.**

use "yourStataFile.dta", clear

load a dataset from the current directory **frequently used commands are highlighted in yellow**

import excel "yourSpreadsheet.xlsx", /*

*** /sheet("Sheet1") cellrange(A2:H11) firstrow**

import an Excel spreadsheet

import delimited "yourFile.csv", /*

*** /rowrange(2:11) colrange(1:8) varnames(2)**

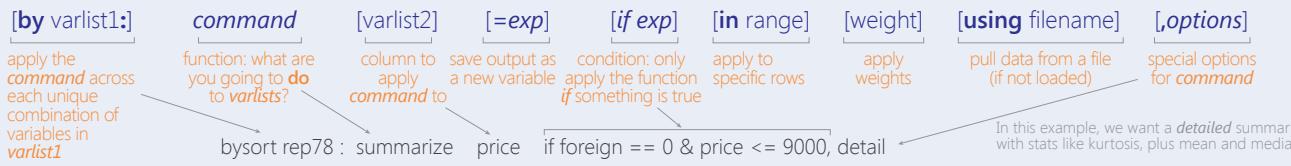
import a .csv file

webuse set "https://github.com/GeoCenter/StataTraining/raw/master/Data2/Data"

webuse "wb_indicators_long"

set web-based directory and load data from the web

All Stata functions have the same format (syntax):



To find out more about any command – like what options it takes – type **help command**

Basic Data Operations

Arithmetic

+ add (numbers)
+ combine (strings)
- subtract
* multiply
/ divide
^ raise to a power

Logic

& and
! or ~ not
| or

if foreign != 1 & price >= 10000

== tests if something is equal
= assigns a value to a variable

== equal
< less than
!= not or
~ equal
<= less than or equal to
> greater than
>= greater or equal to

if foreign != 1 | price >= 10000

make foreign price

| | | |
|---------------|---|--------|
| Chevy Volt | 0 | 3,984 |
| Buick Riviera | 0 | 10,372 |
| Honda Civic | 1 | 4,499 |
| Volvo 260 | 1 | 11,995 |

Change Data Types

Stata has 6 data types, and data can also be missing:

no data **true/false** **words** **numbers**

missing **byte** **string** **int long float double**

To convert between numbers & strings:

1 **gen foreignString = string(foreign)** "1"
tostring foreign, **gen(foreignString)** "1"
decode foreign, **gen(foreignString)** "foreign"

1 **gen foreignNumeric = real(foreignString)** "1"
destring foreignString, **gen(foreignNumeric)** "1"
encode foreignString, **gen(foreignNumeric)** "foreign"

recast double mpg

generic way to convert between types

Summarize Data

include missing values create binary variable for every rep78

tabulate rep78, mi gen(repairRecord)

one-way table: number of rows with each value of rep78

tabulate rep78 foreign, mi

two-way table: cross-tabulate number of observations for each combination of rep78 and foreign

bysort rep78: tabulate foreign

for each value of rep78, apply the command tabulate foreign

tabstat price weight mpg, by(foreign) stat(mean sd n)

create compact table of summary statistics displays stats formats numbers for all data

table foreign, contents(mean price sd price) f(%9.2fc) row

create a flexible table of summary statistics

collapse (mean) price (max) mpg, by(foreign) – replaces data

calculate mean price & max mpg by car type (foreign)

Create New Variables

generate mpgSq = mpg^2 **gen byte lowPr = price < 4000**

create a new variable. Useful also for creating binary variables based on a condition (**generate byte**)

generate id = _n

bysort rep78: gen repairIdx = _n

_n creates a running index of observations in a group

generate totRows = _N

bysort rep78: gen repairTot = _N

_N creates a total count of observations (per group)

pctile mpgQuartile = mpg, nq = 4

create quartiles of the mpg data

egeen meanPrice = mean(price), by(foreign)

see **help egen** for more options

Data Transformation with Stata 14.1 Cheat Sheet

For more info see Stata's reference manual (stata.com)

Select Parts of Data (Subsetting)

SELECT SPECIFIC COLUMNS

drop make

remove the 'make' variable

keep make price

opposite of drop; keep only columns 'make' and 'price'

FILTER SPECIFIC ROWS

drop if mpg < 20 **drop in 1/4**

drop observations based on a condition (left) or rows 1-4 (right)

keep in 1/30

opposite of drop; keep only rows 1-30

keep if inrange(price, 5000, 10000)

keep values of price between \$5,000 – \$10,000 (inclusive)

keep if inlist(make, "Honda Accord", "Honda Civic", "Subaru")

keep the specified values of make

sample 25

sample 25% of the observations in the dataset
(use **set seed #** command for reproducible sampling)

Replace Parts of Data

CHANGE COLUMN NAMES

rename (rep78 foreign) (repairRecord carType)

rename one or multiple variables

CHANGE ROW VALUES

replace price = 5000 if price < 5000

replace all values of price that are less than \$5,000 with 5000

recode price (0 / 5000 = 5000)

change all prices less than 5000 to be \$5,000

recode foreign (0 = 2 "US") (1 = 1 "Not US"), gen(foreign2)

change the values and value labels then store in a new variable, foreign2

REPLACE MISSING VALUES

mvdecode _all, mv(9999) useful for cleaning survey datasets
replace the number 9999 with missing value in all variables

mvencode _all, mv(9999) useful for exporting data
replace missing values with the number 9999 for all variables

Label Data

Value labels map string descriptions to numbers. They allow the underlying data to be numeric (making logical tests simpler) while also connecting the values to human-understandable text.

label define myLabel 0 "US" 1 "Not US"

label values foreign myLabel

define a label and apply it to the values in foreign

label list

list all labels within the dataset

Reshape Data

webuse set <https://github.com/GeoCenter/StataTraining/raw/master/Day2/Data/coffeeMaize.dta>
webuse "coffeeMaize.dta"
load demo dataset

MELT DATA (WIDE → LONG)

reshape variables starting with coffee and maize
unique id variable (key) create new variable which captures the info in the column names

reshape long coffee@ maize@, i(country) j(year) — new variable
convert a wide dataset to long



TIDY DATASETS have each observation in its own row and each variable in its own column.

When datasets are tidy, they have a consistent, standard format that is easier to manipulate and analyze.

CAST DATA (LONG → WIDE)

create new variables named coffee2011, maize2012...
what will be unique id variable (key)
create new variables with the year added to the column name

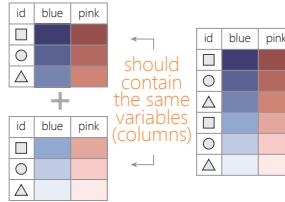
reshape wide coffee maize, i(country) j(year)
convert a long dataset to wide

xpose, clear varname

transpose rows and columns of data, clearing the data and saving old column names as a new variable called "_varname"

Combine Data

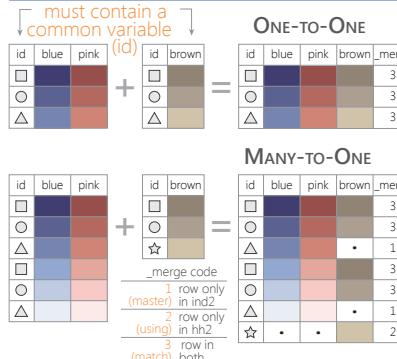
ADDING (APPENDING) NEW DATA



webuse coffeeMaize2.dta, clear
save coffeeMaize2.dta, replace
webuse coffeeMaize.dta, clear
load demo data

append using "coffeeMaize2.dta", gen(filenum)
add observations from "coffeeMaize2.dta" to current data and create variable "filenum" to track the origin of each observation

MERGING TWO DATASETS TOGETHER



webuse ind_age.dta, clear
save ind_age.dta, replace
webuse ind_ag.dta, clear

merge 1:1 id using "ind_age.dta"
one-to-one merge of "ind_age.dta" into the loaded dataset and create variable "_merge" to track the origin

webuse hh2.dta, clear
save hh2.dta, replace
webuse ind2.dta, clear

merge m:1 hid using "hh2.dta"
many-to-one merge of "hh2.dta" into the loaded dataset and create variable "_merge" to track the origin

relink match records from different data sets using probabilistic matching [ssc install relink](#)

jarowinkler create distance measure for similarity between two strings [ssc install jarowinkler](#)

Manipulate Strings

GET STRING PROPERTIES

display length("This string has 29 characters")
return the length of the string

charlist make * user-defined package
display the set of unique characters within a string

display strpos("Stata", "a")
return the position in Stata where a is first found

FIND MATCHING STRINGS

display strmatch("123.89", "?1???.?9")
return true (1) or false (0) if string matches pattern

display substr("Stata", 3, 5)
return the string located between characters 3-5

list make if regexm(make, "[0-9]")
list observations where make matches the regular expression (here, records that contain a number)

list if regexm(make, "(Cad.|Chev.|Datsun)")
return all observations where make contains "Cad.", "Chev." or "Datsun"

list if inlist(word, make, 1), "Cad.", "Chev.", "Datsun")
return all observations where the first word of the make variable contains the listed words

TRANSFORM STRINGS

display regexpr("My string", "My", "Your")
replace string1 ("My") with string2 ("Your")

replace make = **subinstr**(make, "Cad.", "Cadillac", 1)
replace first occurrence of "Cad." with Cadillac in the make variable

display strtrim(" Too much Space")
replace consecutive spaces with a single space

display trim(" leading / trailing spaces ")
remove extra spaces before and after a string

display strlower("STATA should not be ALL-CAPS")
change string case; see also **strupper**, **strproper**

display strtoname("1Var name")
convert string to Stata-compatible variable name

display real("100")
convert string to a numeric or missing value

Save & Export Data

save "myData.dta", **replace** Stata 12-compatible file
saveold "myData.dta", **replace version(12)**

save data in Stata format, replacing the data if a file with same name exists

export excel "myData.xls", /* firstrow(variables) **replace**

export data as an Excel file (.xls) with the variable names as the first row

export delimited "myData.csv", **delimiter(",") replace**
export data as a comma-delimited file (.csv)

Data Analysis

with Stata 14.1 Cheat Sheet

For more info see Stata's reference manual ([stata.com](#))

Results are stored as either **i**-class or **e**-class. See [Programming Cheat Sheet](#)

Summarize Data

Examples use `auto.dta` (`sysuse auto, clear`) unless otherwise noted

univar `price mpg, boxplot`

`scc install univar`

calculate univariate summary, with box-and-whiskers plot

stem `mpg`

return stem-and-leaf display of mpg

summarize `price mpg, detail`

— used commands are highlighted in yellow

calculate a variety of univariate summary statistics

ci `mpg price, level(99)`

compute standard errors and confidence intervals

correlate `mpg price`

return correlation or covariance matrix

pwcorr `price mpg weight, star(0.05)`

return all pairwise correlation coefficients with sig. levels

mean `price mpg`

estimates of means, including standard errors

proportion `rep78 foreign`

estimates of proportions, including standard errors for categories identified in varlist

ratio

estimates of ratio, including standard errors

total `price`

estimates of totals, including standard errors

Statistical Tests

tabulate `foreign rep78, chi2 exact expected`

tabulate foreign and repair record and return chi² and Fisher's exact statistic alongside the expected values

ttest `mpg, by(foreign)`

estimate t test on equality of means for mpg by foreign

prtest `foreign == 0.5`

one-sample test of proportions

ksmirnov `mpg, by(foreign) exact`

Kolmogorov-Smirnov equality-of-distributions test

ranksum `mpg, by(foreign) exact`

equality tests on unmatched data (independent samples)

anova `systolic drug` `webuse systolic, clear`

analysis of variance and covariance

pwmean `mpg, over(rep78) pveffects mcompare(tukey)`

estimate pairwise comparisons of means with equal variances include multiple comparison adjustment

Estimation with Categorical & Factor Variables

CONTINUOUS VARIABLES

measure something

CATEGORICAL VARIABLES

identify a group to which an observations belongs

INDICATOR VARIABLES

denote whether something is true or false

OPERATOR

DESCRIPTION

EXAMPLE

regress

price i.rep78

specify rep78 variable to be an indicator variable

regress

price ib(3).rep78

set the third category of rep78 to be the base category

fvset

base frequent rep78

set the base to most frequently occurring category for rep78

c.

treat variable as continuous

treat mpg as a continuous variable and

o.

omit a variable or indicator

specify an interaction between foreign and mpg

#

specify interactions

set rep78 as an indicator; omit observations with rep78 == 2

##

specify factorial interactions

create a squared mpg term to be used in regression

regress price c.mpg#c.mpg

create all possible interactions with mpg (mpg and mpg²)

By declaring data type, you enable Stata to apply data munging and analysis functions specific to certain data types

Declare Data

webuse sunspot, clear



tset time, yearly

declare sunspot data to be yearly time series

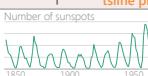
tsreport

report time series aspects of a dataset

generate lag_spot = L1.spot

create a new variable of annual lags of sun spots

tsline plot



tsline spot

plot time series of sunspots

arima spot, ar(1/2)

estimate an auto-regressive model with 2 lags

TIME SERIES OPERATORS

L. lag x_{t-1}

L2. 2-period lag x_{t-2}

F. lead x_{t+1}

F2. 2-period lead x_{t+2}

D. difference x_t - x_{t-1}

D2. difference of difference x_{t-1} - (x_{t-1} - x_{t-2})

S. seasonal difference x_t - x_{t-12}

S2. lag-2 (seasonal difference) x_t - x_{t-24}

USEFUL ADD-INS

tscollap

compact time series into means, sums and end-of-period values

carryforward

carry non-missing values forward from one obs. to the next

tsspell

identify spells or runs in time series

webuse drugtr, clear



SURVIVAL ANALYSIS

tset studytime, failure(died)

declare survey design for a dataset

stsum

summarize survival-time data

stcox drug age

estimate a cox proportional hazard model

PANEL / LONGITUDINAL

xtset id year

declare national longitudinal data to be a panel

xtdescribe

report panel aspects of a dataset

xtsum hours

summarize hours worked, decomposing standard deviation into between and within components

xline ln_wage if id <= 22, tlabel(#3)

plot panel data as a line plot

xtreg ln_w c.age##c.age ttl_exp, fe vce(robust)

estimate a fixed-effects model with robust standard errors

SURVEY DATA

svset psuid [pweight = finalwgt], strata(stratid)

declare survey design for a dataset

svydescribe

report survey data details

svy: mean age, over(sex)

estimate a population mean for each subpopulation

svy, subpop(rural): mean age

estimate a population mean for rural areas

svy: tabulate sex heartatk

report two-way table with tests of independence

svy: reg zinc c.age##c.age female weight rural

estimate a regression using survey weights

2 Diagnostics

not appropriate with robust standard errors

estat hettest

test for heteroskedasticity

ovtest

test for omitted variable bias

vif

report variance inflation factor

dfbeta(length)

calculate measure of influence

rvpplot, yline(0)

plot residuals against fitted values

avplots

plot all partial-leverage plots in one graph

Type `help regress postestimation plots` for additional diagnostic plots

regress price headroom length

Used in all postestimation examples

display _b[length]

return coefficient estimate or standard error for mpg from most recent regression model

margins, dydx[length]

returns e-class information when post option is used

margins, dydx[length]

return the estimated marginal effect for mpg

margins, eyex[length]

return the estimated elasticity for price

predict yhat if e(sample)

create predictions for sample on which model was fit

predict double resid, residuals

calculate residuals based on last fit model

test mpg = 0

test linear hypotheses that mpg estimate equals zero

lincom headroom - length

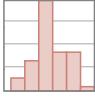
test linear combination of estimates (headroom = length)

Data Visualization with Stata 14.1 Cheat Sheet

For more info see Stata's reference manual ([stata.com](#))

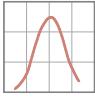
ONE VARIABLE

CONTINUOUS



histogram mpg, width(5) freq kdensity kdenopts(bwidth(5))

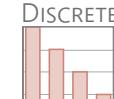
histogram
bin(# • width(#)) • density • fraction • frequency • percent • addlables
addalbopts(<options>) • normal • normopts(<options>) • kdensity
kdenopts(<options>)



kdensity mpg, bwidth(3)

smoothed histogram
bwidth • kernel(<options>)
normal • normopts(<line options>)

main plot-specific options;
see help for complete set



graph bar (count), over(foreign, gap(*0.5)) intensity(*0.5)

bar plot draws horizontal bar charts

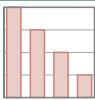
(asis) • (percent) • (count) • over(<variable>, <options: gap(#)> •
relabel • descending • reverse) • cw • missing • nofill • allcategories •
percentages • stack • bargap(#) • intensity(#*) • yalternate • xlabelname

graph bar (percent), over(rep78) over(foreign)

grouped bar plot draws horizontal bar charts

(asis) • (percent) • (count) • over(<variable>, <options: gap(#)> •
relabel • descending • reverse) • cw • missing • nofill • allcategories •
percentages • stack • bargap(#) • intensity(#*) • yalternate • xlabelname

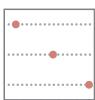
DISCRETE X, CONTINUOUS Y



graph bar (median) price, over(foreign)

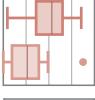
graph hbar

bar plot (asis) • (percent) • (count) • (stat: mean median sum min max ...)
over(<variable>, <options: gap(#)> • relabel • descending • reverse
sort(<variable>)) • cw • missing • nofill • allcategories • percentages
stack • bargap(#) • intensity(#*) • yalternate • xlabelname



graph dot (mean) length headroom, over(foreign) m(1, ms(S))

dot plot (asis) • (percent) • (count) • (stat: mean median sum min max ...)
over(<variable>, <options: gap(#)> • relabel • descending • reverse
sort(<variable>)) • cw • missing • nofill • allcategories • percentages
linegap(# • marker(#, <options>) • linetype(dot | line | rectangle)
dots(<options>) • lines(<options>) • rectangles(<options>) • rwidth



graph hbox mpg, over(rep78, descending) by(foreign) missing

graph box draws vertical boxplots

over(<variable>, <options: total • gap(#)* • relabel • descending • reverse
sort(<variable>)) • missing • allcategories • intensity(#*) • boxgap(#)
medtype(line | marker) • medline(<options>) • medmarker(<options>)



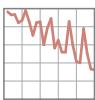
vioplot price, over(foreign)

ssc install vioplot

violin plot over(<variable>, <options: total • missing>) • nofill •
vertical • horizontal • obs • kernel(<options>) • bwidth(#)
barwidth(# • dscale(# • ygap(# • ogap(# • density(<options>)
bar(<options>) • median(<options>) • obsopts(<options>)

Plot Placement

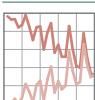
JUXTAPOSE (FACET)



twoway scatter mpg price, by(foreign, norescale)

total • missing • colfirst • rows(# • cols(# • holes(<numlist>)
compact • nojelabel • nojrescale • nojrescal • nojxrescale
nojyaxes • nojxaxes • nojyjtick • nojxitick • nojylabel
nojxlabel • nojytitle • nojxititle • imargin(<options>)

SUPERIMPOSE



graph combine plot1.gph plot2.gph...

combine 2+ saved graphs into a single plot

scatter y3 y2 y1 x, marker(i o i) mlabel(var3 var2 var1)

plot several y values for a single x value

graph twoway scatter mpg price in 27/74 || scatter mpg price /*
*/ if mpg < 15 & price > 12000 in 27/74, mlabel(make) m(i)
combine twoway plots using ||

BASIC PLOT SYNTAX:

| | | | | |
|---|-----------------------|-------------------|--|-------------------------------|
| graph <plot type> variables: y first y ₁ y ₂ ... y _n x [in] [if], <plot options> | plot-specific options | facet by(var) | xline(xint) yline(yint) text(y x "annotation") | annotations |
| titles | | | | axes |
| <marker, line, text, axis, legend, background options> | | custom appearance | | |
| | | scheme(s1mono) | plot size | save |
| | | play(customTheme) | xsize(5) ysize(4) | saving("myPlot.gph", replace) |

TWO+ CONTINUOUS VARIABLES



graph matrix mpg price weight, half
scatter plot of each combination of variables
half • jitter(# • jitterseed(#)) • diagonal • [aweights(<variable>)]



twoway scatter mpg weight, jitter(7)
scatter plot
jitter(# • jitterseed(#)) • sort • cmissing(yes | no)
connect(<options>) • [aweight(<variable>)]



twoway scatter mpg weight, mlabel(mpg)
scatter plot with labelled values
jitter(# • jitterseed(#)) • sort • cmissing(yes | no)
connect(<options>) • [aweight(<variable>)]



twoway connected mpg price, sort(price)
scatter plot with connected lines and symbols
jitter(# • jitterseed(#)) • sort • see also line
connect(<options>) • cmissing(yes | no)



twoway area mpg price, sort(price)
line plot with area shading
sort • cmissing(yes | no) • vertical • horizontal
base(#)



twoway bar price rep78
bar plot
vertical • horizontal • base(#) • barwidth(#)



twoway dot mpg rep78
dot plot
vertical • horizontal • base(#) • ndots(#)
dcolor(<color>) • dfcolor(<color>) • dlcolor(<color>)
dsizex(<marker size>) • dsymbol(<marker type>)
dlwidth(<stroke size>) • dotextend(yes | no)



twoway dropline mpg price in 1/5
dropped line plot
vertical • horizontal • base(#)



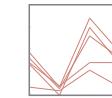
twoway rcpasym length headroom price
range plot (y₁ ÷ y₂) with capped lines
vertical • horizontal
see also rcp



twoway rarea length headroom price, sort
range plot (y₁ ÷ y₂) with area shading
vertical • horizontal • sort
cmissing(yes | no)



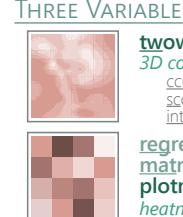
twoway rbar length headroom price
range plot (y₁ ÷ y₂) with bars
vertical • horizontal • barwidth(# • mwidth
msize(<marker size>))



twoway pcspike wage68 ttl_exp68 wage88 ttl_exp88
Parallel coordinates plot
vertical • horizontal
([sysuse nlswide1](#))

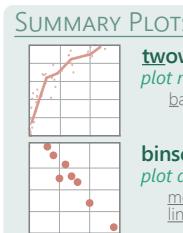


twoway pccapsym wage68 ttl_exp68 wage88 ttl_exp88
Slope/bump plot
vertical • horizontal • headlabel
([sysuse nlswide1](#))



twoway contour mpg price weight, level(20) crule(intensity)
3D contour plot
ccuts(#s) • levels(# • minmax • crule(hue | chue | intensity)) •
scolor(<color>) • ecolor(<color>) • colors(<colorlist>) • heatmap
interp(thinplaine | shepard | none)

regress price mpg trunk weight length turn, nocons
matrix regmat = e(V)
matrix regmat = **mat**(regmat) **color**(green)
plotmatrix, **mat**(regmat) **color**(green)
heatmap mat(<variable>) • split(<options>) • color(<color>) • freq



twoway mband mpg weight || scatter mpg weight
plot median of the y values
bands(#)

binscatter weight mpg, line(none)
plot a single value (mean or median) for each x value
medians • nquantiles(# • discrete • controls(<variables>)) •
linetype(lfit | qfit | connect | none) • [aweight(<variable>)]



twoway lfitci mpg weight || scatter mpg weight
calculate and plot linear fit to data with confidence intervals
level(# • stdp • stdf • nofit • fitplot(<plottype>) • ciplot(<plottype>)) •
range(# • n(# • atobs • estopts(<options>)) • predopts(<options>))



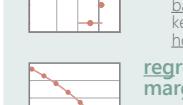
twoway lowess mpg weight || scatter mpg weight
calculate and plot lowess smoothing
bwidht(# • mean • noweight • logit • adjust)



twoway qfici mpg weight, alwidth(none) || scatter mpg weight
calculate and plot quadratic fit to data with confidence intervals
level(# • stdp • stdf • nofit • fitplot(<plottype>) • ciplot(<plottype>)) •
range(# • n(# • atobs • estopts(<options>)) • predopts(<options>))



regress price mpg headroom trunk length turn
coefplot, drop_cons xline(0)
Plot regression coefficients
baselevels • b(<options>) • at(<options>) • noci • levels(#)
keep(<variables>) • drop(<variables>) • rename(<list>)
horizontal • vertical • generate(<variable>)

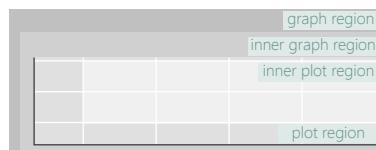


regress mpg weight length turn
margins, eyex(weight) at(weight = (1800(200)4800))
marginsplot, noci
Plot marginal effects of regression
horizontal • noci

Plotting in Stata 14.1

Customizing Appearance

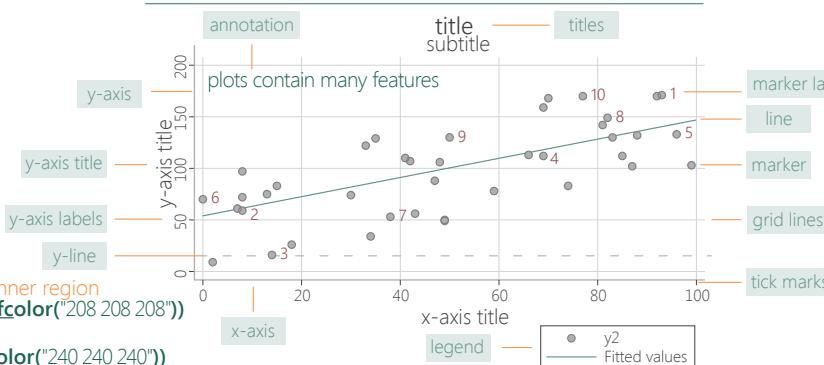
For more info see Stata's reference manual (stata.com)



scatter price mpg, graphregion(fcolor("192 192 192")) ifcolor("208 208 208")
specify the fill of the background in RGB or with a Stata color

scatter price mpg, plotregion(fcolor("224 224 224")) ifcolor("240 240 240")
specify the fill of the plot background in RGB or with a Stata color

ANATOMY OF A PLOT



SYNTAX

marker <marker options>

arguments for the plot objects (in green) go in the options portion of these commands (in orange)

for example:
`scatter price mpg, xline(20, lwidth(vthick))`

COLOR

mcolor("145 168 208") **mcolor(None)**
specify the fill and stroke of the marker in RGB or with a Stata color

mfcolor("145 168 208") **mfcolor(None)**
specify the fill of the marker

SIZE / THICKNESS

| | |
|----------------------|--------------------------|
| msize(medium) | specify the marker size: |
| | ehuge |
| | vhuge |
| | huge |
| | vlarge |
| | large |
| | medhuge |
| | medvhuge |
| | medsmall |
| | small |
| | vsmal |
| | tiny |
| | vtiny |

APPEARANCE

| | |
|--------------------|----------------------------|
| msymbol(Dh) | specify the marker symbol: |
| ● O | D |
| ● o | d |
| ○ Oh | Dh |
| ○ oh | dh |
| + | X |
| + | p |
| + | none |
| + | i |
| ▲ T | S |
| ▲ t | s |
| △ Th | Sh |
| △ th | sh |

POSITION

| | |
|----------------------|-------------------------------|
| jitter(#) | randomly displace the markers |
| jitterseed(#) | set seed |

Apply Themes

Schemes are sets of graphical parameters, so you don't have to specify the look of the graphs every time.

USING A SAVED THEME

twoway scatter mpg price, scheme(customTheme)

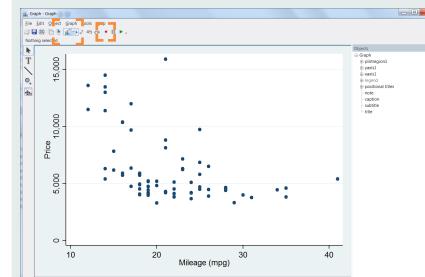
help scheme entries Create custom themes by saving options in a .scheme file
see all options for setting scheme properties

adopath ++ " ~/<location>/StataThemes" set path of the folder (StataThemes) where custom .scheme files are saved
set as default scheme

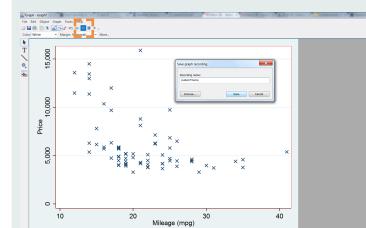
set scheme customTheme, permanently change the theme

USING THE GRAPH EDITOR

twoway scatter mpg price, play(graphEditorTheme)



Select the Graph Editor



Click Record

Double click on symbols and areas on plot, or regions on sidebar to customize

Unclick Record

Save theme as a .grec file

Save Plots

graph twoway scatter y x, saving("myPlot.gph") replace
save the graph when drawing

graph save "myPlot.gph", replace
save current graph to disk

graph combine plot1.gph plot2.gph...
combine 2+ saved graphs into a single plot

graph export "myPlot.pdf", as(.pdf)
export the current graph as an image file see options to set size and resolution