

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 + **8**

open the data editor

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

or

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

follow us @StataRGIS and @flaneuseks

All Stata functions have the same format (syntax):

[by varlist1:]
apply the **command** across each unique combination of variables in **varlist1**

command
function: what are you going to **do** to **varlist**?

[varlist2]
column to apply **command** to

[=exp]
save output as a new variable

[if exp]
condition: only apply the function if something is true

[in range]
apply to specific rows

[weight]
apply weights

[using filename]
pull data from a file (if not loaded)

[options]
special options for **command**

In this example, we want a **detailed** summary with stats like kurtosis, plus mean and median

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

== tests if something is equal
!= or ~ equal

if foreign != 1 & price >= 10000

make foreign price

Chevy Colt 0 3.984

Buick Riviera 0 10.372

Honda Civic 1 4.499

Volvo 260 1 11.995

== assigns a value to a variable
< less than
<= less than or equal to

> greater than
>= greater or equal to

if foreign != 1 | price >= 10000

make foreign price

Chevy Colt 0 3.984

Buick Riviera 0 10.372

Honda Civic 1 4.499

Volvo 260 1 11.995

Explore Data

VIEW DATA ORGANIZATION

describe make price

display variable type, format, and any value/variable labels

count

count if price > 5000

number of rows (observations)
Can be combined with logic

ds, has(type string)

lookfor "in."

search for variable types, variable name, or variable label

isid mpg

check if mpg uniquely identifies the data

browse

or **Ctrl** + **8**
open the data editor

list make price if price > 10000 & price < .

list the make and price for observations with price > \$10,000

display price[4]

display the 4th observation in price; only works on single values

gsort price mpg

(ascending)

Missing values are treated as the largest positive number. To exclude missing values, ask whether the value is less than "."

clist ... (compact form)

gsort -price -mpg (descending)

sort in order, first by price then miles per gallon

duplicates report

finds all duplicate values in each variable

levelsof rep78

display the unique values for rep78

SEE DATA DISTRIBUTION

codebook make price

overview of variable type, stats, number of missing/unique values

summarize make price mpg

print summary statistics (mean, stdev, min, max) for variables

inspect mpg

show histogram of data, number of missing or zero observations

histogram mpg, frequency

plot a histogram of the distribution of a variable



BROWSE OBSERVATIONS WITHIN THE DATA

Missing values are treated as the largest positive number. To exclude missing values, ask whether the value is less than "."

clist ... (compact form)

gsort -price -mpg (descending)

sort in order, first by price then miles per gallon

duplicates report

finds all duplicate values in each variable

levelsof rep78

display the unique values for rep78

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 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 = mpg² **gen** byt lowPr = price < 4000

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

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 running count of the total observations per group

pctile mpgQuartile = mpg, nq = 4

create quartiles of the mpg data

egen meanPrice = **mean**(price), **by**(foreign) see help egen for more options

calculate mean price for each group in foreign

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 numerics. 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
webuse "coffeeMaize.dta" load demo dataset

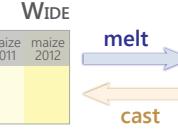
MELT DATA (WIDE → LONG)

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

country	coffee 2011	coffee 2012	maize 2011	maize 2012
Malawi				
Rwanda				
Uganda				

country	year	coffee	maize
Malawi	2011		
Malawi	2012		
Rwanda	2011		
Rwanda	2012		
Uganda	2011		
Uganda	2012		



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

CAST DATA (LONG → WIDE)

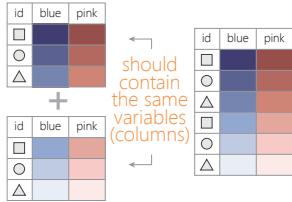
create new variables named coffee2011, maize2012... what will be unique id variable (key) create new variables with the year added to the column name

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

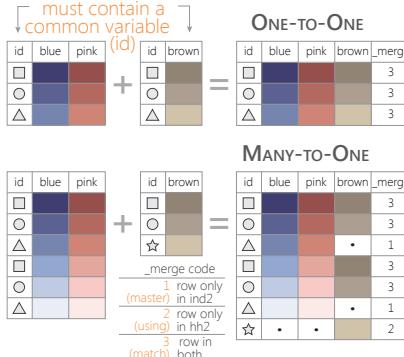
ADDING (APPENDING) NEW DATA



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"

compare the given list against the first word in make

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`

— highlighted in yellow

calculate a variety of univariate summary statistics

ci `mean mpg price, level(99)` — for Stata 13: `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 χ^2 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

Declare Data

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

TIME SERIES

tset `time, yearly`

declare sunspot data to be yearly time series



tssreport

report time series aspects of a dataset

generate `lag_spot = L1.spot`

create a new variable of annual lags of sun spots

tline

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}

F. lead x_{t+1}

D. difference $x_t - x_{t-1}$

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

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

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

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

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

SURVIVAL ANALYSIS

`webuse drugtr, clear`

tset `studytime, failure(died)`

declare survey design for a dataset



stsum

summarize survival-time data

stcox `drug age`

estimate a cox proportional hazard model

1 Estimate Models

stores results as **e**-class

regress `price mpg weight, robust`

estimate ordinary least squares (OLS) model

on mpg weight and foreign, apply robust standard errors

regress `price mpg weight if foreign == 0, cluster(rep78)`

regress price only on domestic cars, cluster standard errors

rreg `price mpg weight, genwt(reg_wt)`

estimate robust regression to eliminate outliers

probit `foreign turn price, vce(robust)`

estimate probit regression with robust standard errors

logit `foreign headroom mpg, or`

estimate logistic regression and report odds ratios

bootstrap, reps(100): regress `mpg /*`

`*/ weight gear foreign`

estimate regression with bootstrapping

jackknife `r(mean), double: sum` `mpg`

jackknife standard error of sample mean

ADDITIONAL MODELS

pca ← built-in Stata command

factor

poisson • **nbreg**

tobit

ivregress `ivreg2`

diff

rd ssc install `ivreg2`

xtabond `xtabond2`

psmatch2

synth

oaxaca

principal components analysis

factor analysis

count outcomes

censored data

instrumental variables

difference-in-difference

regression discontinuity

dynamic panel estimator

propensity score matching

synthetic control analysis

Blinder-Oaxaca decomposition

PANEL / LONGITUDINAL

xset `id year`

declare national longitudinal data to be a panel

`webuse nlswwork, clear`



xtdescribe

report panel aspects of a dataset

xsum `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



sydescribe

report survey data details

svy: mean `age, over(sex)`

estimate a population mean for each subpopulation

svy, subpop(rural): mean

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

estat `hettest` test for heteroskedasticity

ovtest test for omitted variable bias

vif report variance inflation factor

Type `help regress postestimation plots` for additional diagnostic plots

dfbeta(length)

calculate measure of influence

rvpplot, yline(0)

plot residuals against fitted values

avplots

plot all partial-leverage leverage plots in one graph

3 Postestimation

commands that use a fitted model

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

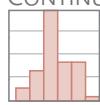
lincom `headroom - length`

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

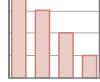


histogram mpg, width(5) freq kdensity kdenopts(bwidth(5))

kdensity mpg, bwidth(3)
smoothed histogram
bwidth • kernel(<options>) • normal • normopts(<line options>)

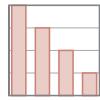
main plot-specific options;
see help for complete set

DISCRETE

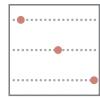


graph bar (count), over(foreign, gap(*0.5)) intensity(*0.5)
bar plot
bar plot draws horizontal bar charts
(asis) • (percent) • (count) • over(<variable>, <options: gap(#)> • relabel • descending • reverse) • cw • missing • nofill • allcategories • percentages • stack • bargap(#) • intensity(#) • yalternate • xlabel
graph bar (percent), over(rep78) over(foreign)
grouped bar plot
grouped bar plot
(asis) • (percent) • (count) • over(<variable>, <options: gap(#)> • relabel • descending • reverse) • cw • missing • nofill • allcategories • percentages • stack • bargap(#) • intensity(#) • yalternate • xlabel

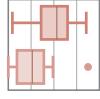
DISCRETE X, CONTINUOUS Y



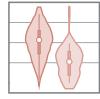
graph bar (median) price, over(foreign)
bar plot
bar plot
(asis) • (percent) • (count) • (stat: mean median sum min max ...) over(<variable>, <options: gap(#)> • relabel • descending • reverse sort(<variable>)) • cw • missing • nofill • allcategories • percentages stack • bargap(#) • intensity(#) • yalternate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



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



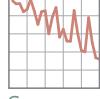
graph hbox mpg, over(rep78, descending) by(foreign)
missing box plot
graph box draws vertical boxplots
over(<variable>, <options: total • gap(#)> • relabel • descending • reverse sort(<variable>)) • missing • allcategories • intensity(#) • boxgap(#) medtype(line | line | marker) • medline(<options>) • medmarker(<options>)



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

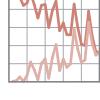
Plot Placement

JUXTAPOSE (FACET)



twoway scatter mpg price, by(foreign, norescale)
total • missing • colfirst • rows(# • cols#) • cols(<numlist>) keep(<variables>) • drop(<variables>) • bwidth(#)
compact • nojedgelabel • nojrescale • nojyrescale • nojxrescale
nojiyaxes • nojiyaxes • nojiytitle • nojxitick • nojiylabel
nojixlabel • nojiytitle • nojxititle • imargin(<options>)

SUPERIMPOSE



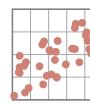
graph combine plot1.gph plot2.gph...
combine 2+ saved graphs into a single plot
scatter y3 y2 y1 x, marker(i o) mlabel(var3 var2 var1)
plot several y values for a single x value
graph twoway scatter mpg price in 27/74 || scatter mpg price /*
*/ if mpg < 15 & price > 12000 in 27/74, mlabel(make) m(i)
combine twoway plots using ||

BASIC PLOT SYNTAX:
graph <plot type> variables: y first
y₁ y₂ ... y_n x [in] [if], <plot options>
titles plot-specific options
titles by(var) facet axes
titles xtitle("x-axis title") ytitle("y axis title") xscale(range(low high)) log reverse off noline yscale(<options>)
titles xline(xint) yline(yint) text(y x "annotation")
titles axes
titles custom appearance
titles scheme(s1mono) play(customTheme) xsize(5) ysize(4) saving("myPlot.gph", replace)
titles plot size
titles save
titles

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>))



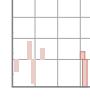
twoway scatter mpg weight, mlabel(mpg)
scatter plot with labelled values
jitter(#) • jitterseed(#) • sort • cmissing(yes | no) • connect(<options>) • aweight(<variable>))



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



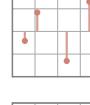
twoway area mpg price, sort(price)
line plot with area shading
sort • cmissing(yes | no) • vertical • horizontal base(#)



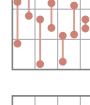
twoway bar price rep78
bar plot
vertical • horizontal • base(#) • barwidth(#)



twoway dot mpg rep78
dot plot
vertical • horizontal • base(#) • ndots(#) • dcolor(<color>) • dcfcolor(<color>) • dcolor(<color>) • dsizel(<marker size>) • dsymbol(<marker type>) • dlwidth(<stroke size>) • dotted(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 msizel(<marker size>))



twoway pspike 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](#))

THREE VARIABLES



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(thinplatespline | shepard | none)

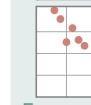


regress price mpg trunk weight length turn, nocons
matrix regmat = e(V)
plotmatrix, mat(regmat) color(green)
heatmap
mat(<variable>) • split(<options>) • color(<color>) • freq

SUMMARY PLOTS



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



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

FITTING RESULTS



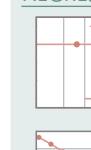
twoway lfitci mpg weight || scatter mpg weight
calculate and plot linear fit to data with confidence intervals
level(# • stdp • stdf • nofit • fitplot(<plottype>) • ciplot(<plottype>) • range(# • n(# • atobs • estopts(<options>) • preopts(<options>))



twoway lowess mpg weight || scatter mpg weight
calculate and plot lowess smoothing
bwwidth(# • 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>) • preopts(<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>)

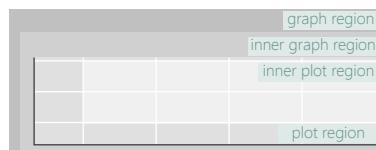


regress mpg weight length turn
margins, eyex(weight) at(weight = (1800(200)4800))
marginsplot, noci
Plot marginal effects of regression
horizontal • noci

Plotting in Stata 14.1

Customizing Appearance

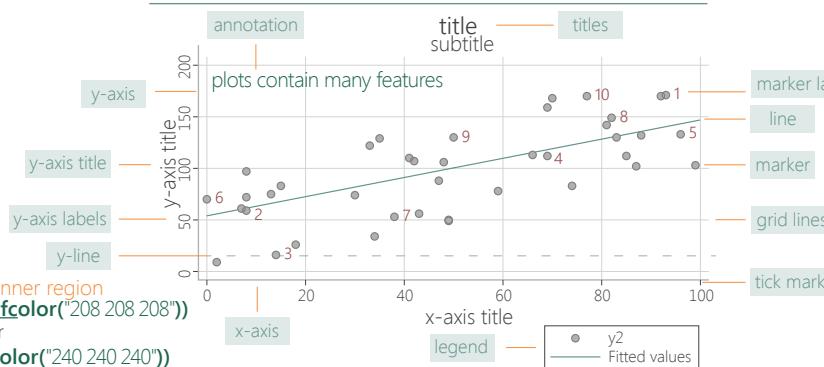
For more info see Stata's reference manual (stata.com)



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))

LINES / BORDERS

line
<line options>
xline(...)
yline(...)

marker
<marker options>

axes
xscale(...)
yscale(...)

tick marks
grid lines

legend
legend(region(...))

TEXT

marker label
<marker options>

titles
title(...)
subtitle(...)

annotation
text(...)

axis labels
xlabel(...)
ylabel(...)

legend
legend(...)

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	medlarge
vhuge	medium
huge	medsmall
vlarge	small
large	vsmall
	tiny
	vtiny

lwidth(medthick)
specify the thickness (stroke) of a line:
marker
tick marks
grid lines

vvvthick	medthin
vvthick	thin
vthick	vthin
thick	vvthin
medthick	vvvthin
	none

size(medsmall)
specify the size of the text:
marker label **mlabsize(medsmall)**
axis labels **labsize(medsmall)**

28 pt.	vhuge	10 pt.	medsmall
20 pt.	huge	8 pt.	small
16 pt.	vlarge	6 pt.	vsmall
14 pt.	large	4 pt.	tiny
12 pt.	medlarge	2 pt.	half_tiny
11 pt.	medium	1.3 pt.	third_tiny
		1 pt.	quarter_tiny
		1 pt.	minuscule

APPEARANCE

msymbol(Dh)	specify the marker symbol:
○	D
o	d
○ Oh	Dh
○ oh	dh
+	X
	p
	none
	i

line
axes
grid lines

ipattern(dash)
gpattern(dash)

solid **longdash** **longdash_dot**

dash **shortdash** **shortdash_dot**

dot **dash_dot** **blank**

axes **noline**
tick marks **noticks**
grid lines **nogrid**

off no axis/labels
ticks **length(2)**
grid **nogmin** **nogmax**

POSITION

jitter(#)
randomly displace the markers

jitterseed(#)
set seed

tick marks **xlabel(#10, tposition(crossing))**
number of tick marks, position (outside | crossing | inside)

marker label **mlabposition(5)**
label location relative to marker (clock position: 0 – 12)

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

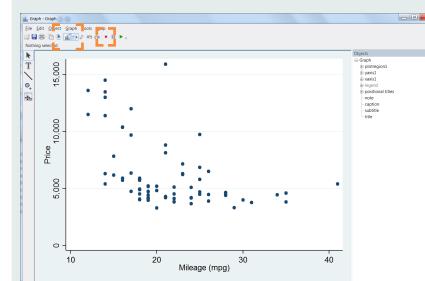
set as default scheme

set scheme customTheme, permanently change the theme

net inst brewscheme, from("https://wbuchanan.github.io/brewscheme/") replace install William Buchanan's package to generate custom schemes and color palettes (including ColorBrewer)

USING THE GRAPH EDITOR

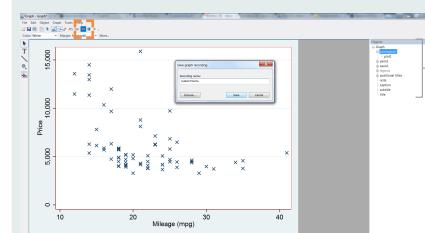
twoway scatter mpg price, play(graphEditorTheme)



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

Programming with Stata 14.1 Cheat Sheet

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

1 Scalars both r- and e-class results contain scalars

scalar `x1 = 3`
create a scalar `x1` storing the number 3
scalar `a1 = "I am a string scalar"`
create a scalar `a1` storing a string

Scalars can hold numeric values or arbitrarily long strings

2 Matrices e-class results are stored as matrices

matrix `a = (4\ 5\ 6)`
create a `3 x 1` matrix
matrix `d = b'` transpose matrix `b`; store in `d`
matrix `ad1 = a \ d`
row bind matrices
matselrc `b x, c(1 3)` findit matselrc
select columns 1 & 3 of matrix `b` & store in new matrix `x`
mat2txt, **matrix(ad1) saving**(textfile.txt) replace
export a matrix to a text file

matrix `b = (7, 8, 9)`
create a `1 x 3` matrix
matrix `ad2 = a , d`
column bind matrices

DISPLAYING & DELETING BUILDING BLOCKS

[scalar | matrix | macro | estimates] [list | drop] `b`

list contents of object `b` or drop (delete) object `b`

[scalar | matrix | macro | estimates] dir

list all defined objects for that class

matrix list b **matrix dir** **scalar drop x1**
list contents of matrix `b` list all matrices delete scalar `x1`

3 Macros public or private variables storing text

GLOBAL available through Stata sessions **PUBLIC**

global `pathdata "C:/Users/SantasLittleHelper/Stata"`

define a global variable called `pathdata`

cd \$pathdata — add a `$` before calling a global macro
change working directory by calling global macro

global myGlobal price mpg length

summarize \$myGlobal
summarize price mpg length using global

LOCALS available only in programs, loops, or .do files **PRIVATE**

local `myLocal price mpg length`

create local variable called `myLocal` with the strings price mpg and length

summarize \myLocal `\` add a `\` before and a `*` after local macro name to call summarize contents of local `myLocal`

levelsof `rep78, local(levels)`

create a sorted list of distinct values of `rep78`, store results in a local macro called `levels`

local `varLab: variable_label foreign` can also do with value labels
store the variable label for `foreign` in the local `varLab`

TEMPVARS & TEMPFILES special locals for loops/programs

tempvar `temp1` — initialize a new temporary variable called `temp1`

generate `'temp1' = mpg^2` — save squared mpg values in `temp1`

summarize `'temp1'` — summarize the temporary variable `temp1`

tempfile `myAuto` create a temporary file to be used within a program

see also `tempname`

Building Blocks basic components of programming

R- AND E-CLASS: Stata stores calculation results in two* main classes:

r return results from general commands such as `summary` or `tabulate` **e** return results from estimation commands such as `regress` or `mean`

To assign values to individual variables use:

- 1 SCALARS **r** individual numbers or strings
- 2 MATRICES **e** rectangular array of quantities or expressions
- 3 MACROS **e** pointers that store text (global or local)

* there's also s- and n-class

4 Access & Save Stored r- and e-class Objects

Many Stata commands store results in types of lists. To access these, use `return` or `ereturn` commands. Stored results can be scalars, macros, matrices or functions.

summarize `price, detail`

return list

returns a list of scalars

scalars:
`r(N) = 74`
`r(mean) = 6165.25...`
`r(var) = 86995225.97...`
`r(sd) = 2949.49...`
...

Results are replaced each time an r-class / e-class command is called

scalars:
`e(df_r) = 73`
`e(N_over) = 1`
`e(N) = 73`
`e(k_eq) = 1`
`e(rank) = 1`

generate `p_mean = r(mean)`

create a new variable equal to average of price

preserve create a temporary copy of active dataframe

restore restore temporary copy to original point

set restore points to test code that changes data

ACCESSING ESTIMATION RESULTS

After you run any estimation command, the results of the estimates are stored in a structure that you can save, view, compare, and export

regress `price weight`

estimates store `est1`

store previous estimation results `est1` in memory

Use `estimates store` to compile results for later use

eststo est2: regress `price weight mpg`

eststo est3: regress `price weight mpg foreign`

estimate two regression models and store estimation results

estimates table `est1 est2 est3`

print a table of the two estimation results `est1` and `est2`

EXPORTING RESULTS

The `estout` and `outreg2` packages provide numerous, flexible options for making tables after estimation commands. See also `putexcel` command.

esttab `est1 est2, se star(* 0.10 ** 0.05 *** 0.01) label`

create summary table with standard errors and labels

esttab using "auto_reg.txt", replace plain se

export summary table to a text file, include standard errors

outreg2 [est1 est2] using "auto_reg2.txt", see replace

export summary table to a text file using `outreg2` syntax

Additional Programming Resources

bit.ly/statacode

download all examples from this cheat sheet in a .do file

adoupdate

Update user-written .ado files

adolist

List/copy user-written .ado files

net install package, from (<https://raw.githubusercontent.com/username/repo/master>)

install a package from a Github repository

s https://github.com/andreweheiss/SublimeStataEnhanced

configure Sublime text for Stata 11-14

Loops: Automate Repetitive Tasks

ANATOMY OF A LOOP

Stata has three options for repeating commands over lists or values: **foreach**, **forvalues**, and **while**. Though each has a different first line, the syntax is consistent:

foreach `x of varlist var1 var2 var3` { open brace must appear on first line
temporary variable used only within the loop
requires local macro notation
command `"x"`, option command(s) you want to repeat
can be one line or many
...
} close brace must appear on final line by itself

FOREACH: REPEAT COMMANDS OVER STRINGS, LISTS, OR VARIABLES

foreach `x in|of [local, global, varlist, newlist, numlist]` {
Stata commands referring to 'x'
}

list types: objects over which the commands will be repeated

STRINGS

foreach `x in auto.dta auto2.dta` {
sysuse "x", clear
tab rep78, missing
}

same as...
append using file1.dta
append using file2.dta

LISTS

foreach `x in "Dr. Nick" "Dr. Hibbert"` {
display length ("Dr. Nick")
display length ("Dr. Hibbert")
}

When calling a command that takes a string, surround the macro name with quotes.

VARIABLES

foreach `x in mpg weight` {
summarize "x"
}

must define list type

foreach `x of varlist mpg weight` {
summarize "x"
}

foreach of requires you to state the list type, which makes it faster

summarize mpg
summarize weight

FORVALUES: REPEAT COMMANDS OVER LISTS OF NUMBERS

forvalues `i = 10(10)50` {
display "i"
}

numeric values over which loop will run

Use display command to show the iterator value at each step in the loop
display 10
display 20
...

ITERATORS
`i = 10/50` → 10, 11, 12, ...
`i = 10(10)50` → 10, 20, 30, ...
`i = 10 20 to 50` → 10, 20, 30, ...

DEBUGGING CODE

set trace on (off)

trace the execution of programs for error checking

PUTTING IT ALL TOGETHER

generate `car_make = word(make, 1)` — pull out the first word from the make variable
levelsof `car_make, local(cmake)` — calculate unique groups of car_make and store in local `cmake`
local `i = 1`
local `cmake_len : word count `cmake'` — store the length of local `cmake` in local `cmake_len`
foreach `x of local cmake {
display in yellow "Make group `i' is `x'"
if i == `cmake_len' {
display "The total number of groups is `i'"
local i = `i' + 1 — increment iterator by one
}`