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 directory

dir
display filenames in working directory

fs *.dta
List all Stata data in working directory

capture log close
close the log on any existing do files

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

log using "myDoFile.txt", **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/Day2/Data"
webuse "wb_indicators_long"
set web-based directory and load data from the web

Basic Syntax

All Stata functions have the same format (syntax):

[**by** varlist1**:**] *command* [varlist2] [=*exp*] [*if exp*] [**in** range] [weight] [**using** filename] [,*options*]

[**by** varlist1**:**]	apply the *command* across each unique combination of variables in *varlist1*
command	function: what are you going to **do** to *varlists*?
[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*

bysort rep78 : summarize price if foreign == 0 & price <= 9000, detail

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

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

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

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

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

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

recast double mpg
generic way to convert between types

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

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

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

Missing values are treated as the largest positive number. To exclude missing values, use the !missing(varname) syntax

list make price **if** price > 10000 & !**missing**(price)
clist ... (compact form)
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) **gsort** –price –mpg (descending)
sort in order, first by price then miles per gallon

duplicates report
finds all duplicate values in each variable

assert price!=.
verify truth of claim

levelsof rep78
display the unique values for rep78

Summarize Data

include missing values — create binary variable for every rep78 value in a new variable, repairRecord

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

formats numbers for all data — displays stats

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 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)
calculate mean price for each group in 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 variables '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 the values in foreign

label list
list all labels within the dataset

note: data note here
place note in dataset

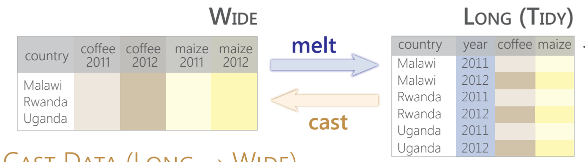
Reshape Data

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



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)

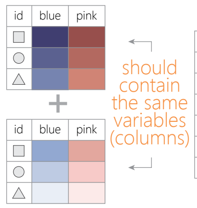
create new variables named coffee2011, maize2012... — what will be unique id variable (key) — create new variables with the year added to the column name

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

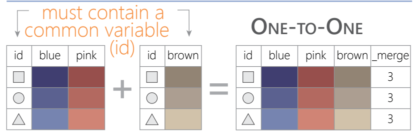
Adding (Appending) New Data



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

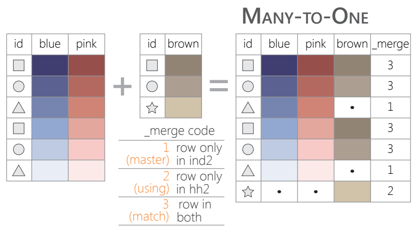
append using "coffeeMaize2.dta", **gen(**filenum**)**
add observations from "coffeeMaize2.dta" to current data and create variable "filenum" to track the origin of each observation

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

Fuzzy Matching: Combining Two Datasets without a Common ID

reclink match records from different data sets using probabilistic matching ssc install reclink

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 regexr("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 stritrim(" 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

compress
compress data in memory

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)

Tim Essam (tessam@usaid.gov) • Laura Hughes (lhughes@usaid.gov)
follow us @StataRGIS and @flaneuseks

Data Visualization with Stata 14.1 Cheat Sheet

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

Basic Plot Syntax:

graph <plot type> y_1 y_2 … y_n x **[in]** **[if],** <plot options> **by(var)** **xline(xint) yline(yint) text(y x "annotation")**

variables: y first · plot-specific options · facet · annotations

title("title") subtitle("subtitle") xtitle("x-axis title") ytitle("y axis title") xscale(range(low high) log reverse off noline) yscale(<options>)

titles · axes

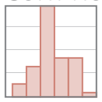
<marker, line, text, axis, legend, background options> **scheme(s1mono) play(customTheme) xsize(5) ysize(4) saving("myPlot.gph", replace)**

custom appearance · plot size · save

One Variable

sysuse auto, clear

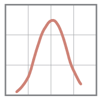
Continuous



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

histogram

bin(#) • width(#) • density • fraction • frequency • percent • addlabels addlabopts(<options>) • normal • normopts(<options>) • kdensity kdenopts(<options>)



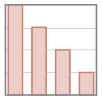
kdensity mpg, **bwidth(3)**

smoothed histogram

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

main plot-specific options; see help for complete set

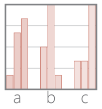
Discrete



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

bar plot — graph hbar draws horizontal bar charts

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

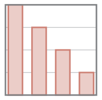


graph bar (percent), over(rep78) over(foreign)

grouped bar plot — graph hbar ...

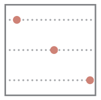
(asis) • (percent) • (count) • over(<variable>, <options: gap(*#) • relabel • descending • reverse>) • cw • missing • nofill • allcategories • percentages • stack • bargap(#) • intensity(*#) • yalternate • xalternate

Discrete X, Continuous Y



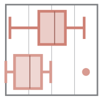
graph bar (median) price, over(foreign) — graph hbar ...

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 • xalternate



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

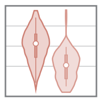
dot plot (asis) • (percent) • (count) • (stat: mean median sum min max ...) over(<variable>, <options: gap(*#) • relabel • descending • reverse sort(<variable>)>) • cw • missing • nofill • allcategories • percentages linegap(#) • marker(#, <options>) • linetype(dot | line | rectangle) dots(<options>)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



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

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>)

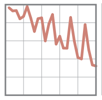
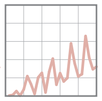


vioplot price, over(foreign) — ssc install vioplot

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

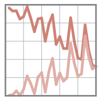
Juxtapose (Facet)

twoway scatter mpg price, **by(foreign, norescale)**

total • missing • colfirst • rows(#) • cols(#) • holes(<numlist>) compact • [no]edgelabel • [no]rescale • [no]yrescal • [no]xrescale [no]iyaxes • [no]ixaxes • [no]iytick • [no]ixtick [no]iylabel [no]ixlabel • [no]iytitle • [no]ixtitle • imargin(<options>)

Superimpose



graph combine plot1.gph plot2.gph...

combine 2+ saved graphs into a single plot

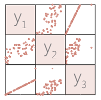
scatter y3 y2 y1 x, marker(i o i) mlabel(var3 var2 var1)

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 ||

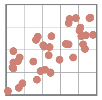
Two+ Continuous Variables



graph matrix mpg price weight, **half**

scatter plot of each combination of variables

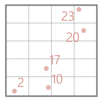
half • jitter(#) • jitterseed(#) diagonal • [aweights(<variable>)]



twoway scatter mpg weight, **jitter(7)**

scatter plot

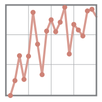
jitter(#) • jitterseed(#) • sort • cmissing(yes | no) connect(<options>) • [aweight(<variable>)]



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)**

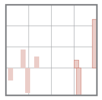
scatter plot with connected lines and symbols

jitter(#) • jitterseed(#) • sort connect(<options>) • cmissing(yes | no) — see also line

twoway area mpg price, **sort(price)**

line plot with area shading

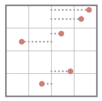
sort • cmissing(yes | no) • vertical, • horizontal base(#)



twoway bar price rep78

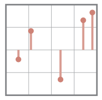
bar plot

vertical, • horizontal • base(#) • barwidth(#)



twoway dot mpg rep78

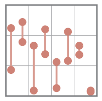
dot plot vertical, • horizontal • base(#) • ndots(#) dcolor(<color>) • dfcolor(<color>) • dlcolor(<color>) dsize(<markersize>) • dsymbol(<marker type>) dlwidth(<strokesize>) • dotextend(yes | no)



twoway dropline mpg price **in** 1/5

dropped line plot

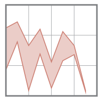
vertical, • horizontal • base(#)



twoway rcapsym length headroom price

range plot (y_1 ÷ y_2) with capped lines

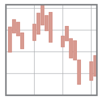
vertical • horizontal — see also rcap



twoway rarea length headroom price, **sort**

range plot (y_1 ÷ y_2) with area shading

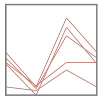
vertical • horizontal • sort cmissing(yes | no)



twoway rbar length headroom price

range plot (y_1 ÷ y_2) with bars

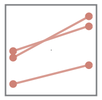
vertical • horizontal • barwidth(#) • mwidth msize(<marker size>)



twoway pcspike wage68 ttl_exp68 wage88 ttl_exp88

Parallel coordinates plot — **(sysuse nlswide1)**

vertical, • horizontal

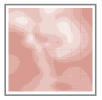


twoway pccapsym wage68 ttl_exp68 wage88 ttl_exp88

Slope/bump plot — **(sysuse nlswide1)**

vertical • horizontal • headlabel

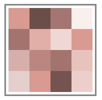
Three Variables



twoway contour mpg price weight, **level(20) crule(intensity)**

3D contour plot

ccuts(#s) • levels(#) • minmax • crule(hue | chue| intensity) • scolor(<color>) • ecolor (<color>) • ccolors(<colorlist>) • heatmap interp(thinplatespline | shepard | none)



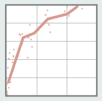
regress price mpg trunk weight length turn, **nocons**

matrix regmat = **e(V)** — ssc install plotmatrix

plotmatrix, mat(regmat) **color**(green)

heatmap mat(<variable) • split(<options>) • color(<color>) • freq

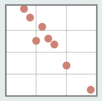
Summary Plots



twoway mband mpg weight **|| scatter** mpg weight

plot median of the y values

bands(#)

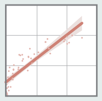


binscatter weight mpg, **line(none)** — ssc install binscatter

plot a single value (mean or median) for each x value

medians • nquantiles(#) • discrete • controls(<variables>) • linetype(lfit | qfit | connect | none) • aweight[<variable>]

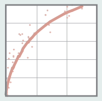
Fitting Results



twoway lfitci mpg weight **|| scatter** mpg weight

calculate and plot linear fit to data with confidence intervals

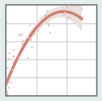
level(#) • stdp • stdf • nofit • fitplot(<plottype>) • ciplot(<plottype>) • range(# #) • n(#) • atobs • estopts(<options>) • predopts(<options>)



twoway lowess mpg weight **|| scatter** mpg weight

calculate and plot lowess smoothing

bwidth(#) • mean • noweight • logit • adjust

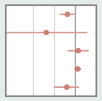


twoway qfitci mpg weight, **alwidth(none) || scatter** mpg weight

calculate and plot quadriatic fit to data with confidence intervals

level(#) • stdp • stdf • nofit • fitplot(<plottype>) • ciplot(<plottype>) • range(# #) • n(#) • atobs • estopts(<options>) • predopts(<options>)

Regression Results

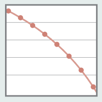


regress price mpg headroom trunk length turn

coefplot, drop(_cons) xline(0) — ssc install coefplot

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

Laura Hughes (lhughes@usaid.gov) • Tim Essam (tessam@usaid.gov)

follow us @flaneuseks and @StataRGIS

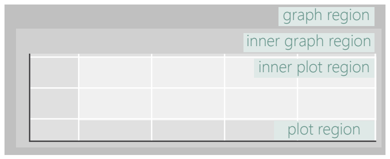
inspired by RStudio's awesome Cheat Sheets (rstudio.com/resources/cheatsheets)

geocenter.github.io/StataTraining

updated February 2016

Plotting in Stata 14.1
Customizing Appearance

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

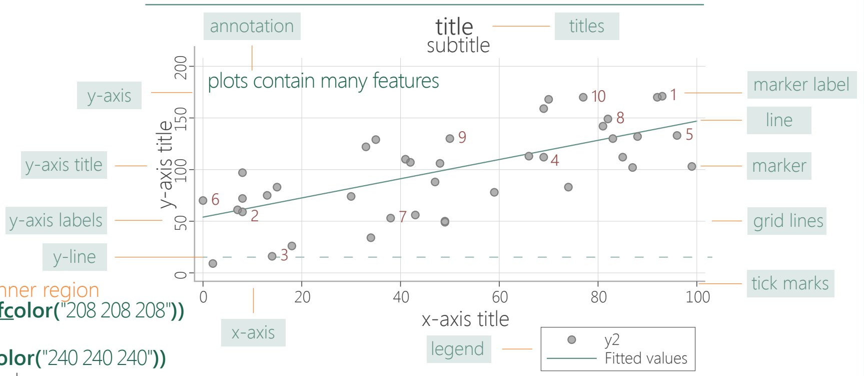


outer region inner region

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



Symbols | Lines / Borders | Text

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**))**

line <line options> xline(...) yline(...)
marker <marker options> legend legend(region(...))
axes xscale(...) yscale(...)
tick marks grid lines xlabel(...) ylabel(...)

marker label <marker options> annotation text(...)
titles title(...) subtitle(...) xtitle(...) ytitle(...)
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

lcolor("145 168 208"**)** **lcolor(**none**)**
specify the stroke color of the line or border
marker **mlcolor(**"145 168 208"**)**
tick marks **tlcolor(**"145 168 208"**)**
grid lines **glcolor(**"145 168 208"**)**

color("145 168 208"**)** **color(**none**)**
specify the color of the text
marker label **mlabcolor(**"145 168 208"**)**
axis labels **labcolor(**"145 168 208"**)**

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 **mlwidth(**thin**)**
tick marks **tlwidth(**thin**)**
grid lines **glwidth(**thin**)**

vvvthick	medthin
vvthick	thin
vthick	vthin
thick	vvthin
medthick	vvvthin
medium	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:

● O	◆ D	▲ T	■ S
• o	◆ d	▲ t	■ s
○ Oh	◇ Dh	△ Th	□ Sh
○ oh	◇ dh	△ th	□ sh
+ +	× X	· p	none i

line axes **lpattern(**dash**)** specify the line pattern
grid lines **glpattern(**dash**)**

solid	longdash	longdash_dot
dash	shortdash	shortdash_dot
dot	dash_dot	blank

axes **noline** axes **off** no axis/labels
tick marks **noticks** tick marks **tlength(**2**)**
grid lines **nogrid** **nogmin** **nogmax**

marker label **mlabel(**foreign**)**
label the points with the values of the foreign variable
axis labels **nolabels**
no axis labels
axis labels **format(**%12.2f **)**
change the format of the axis labels
legend **off**
turn off legend
legend **label(**# "label"**)**
change legend label text

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**)**

Create custom themes by saving options in a .scheme file

help scheme entries
see all options for setting scheme properties

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

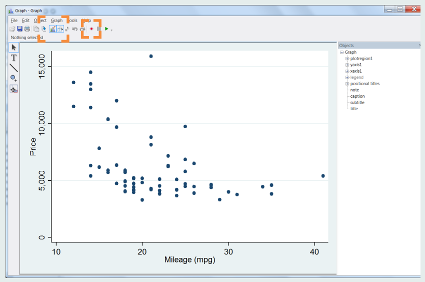
set as default scheme

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

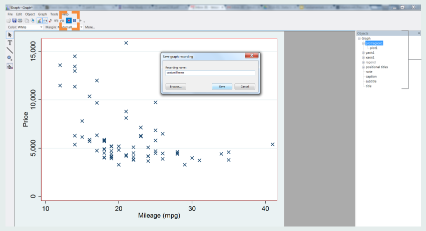
twoway scatter mpg price, **play(**graphEditorTheme**)**



Select the Graph Editor

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

Laura Hughes (lhughes@usaid.gov) • Tim Essam (tessam@usaid.gov)
follow us @flaneuseks and @StataRGIS

inspired by RStudio's awesome Cheat Sheets (rstudio.com/resources/cheatsheets)

geocenter.github.io/StataTraining

updated June 2016

Data Analysis with Stata 14.1 Cheat Sheet

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

Results are stored as either r -class or e -class. See Programming Cheat Sheet

Summarize Data

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

univar price mpg, **boxplot** — ssc install univar
calculate univariate summary, with box-and-whiskers plot

stem mpg
return stem-and-leaf display of mpg

summarize price mpg, **detail** — frequently used commands are 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 chi^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

Declare Data

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

Time Series

webuse sunspot, clear

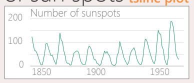
tsset time, **yearly**
declare sunspot data to be yearly time series

tsreport
report time series aspects of a dataset

generate lag_spot = L1.spot
create a new variable of annual lags of sun spots

tsline spot
plot time series of sunspots

tsline plot



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 $x_t - x_{t-1} - (x_{t-1} - x_{t-2})$
S.	seasonal difference $x_t - x_{t-1}$	S2.	lag-2 (seasonal difference) $x_t - x_{t-2}$

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

stset 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

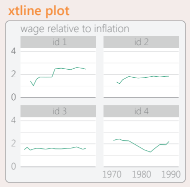
webuse nlswork, clear

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

xtline plot



xtline 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

webuse nhanes2b, clear

svyset 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

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	principal components analysis
factor	factor analysis
poisson • **nbreg**	count outcomes
tobit	censored data
ivregress **ivreg2**	instrumental variables
diff user-written	difference-in-difference
rd ssc install ivreg2	regression discontinuity
xtabond **xtabond2**	dynamic panel estimator
psmatch2	propensity score matching
synth	synthetic control analysis
oaxaca	Blinder-Oaxaca decomposition

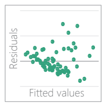
2 Diagnostics

not appropriate after robust cluster( )

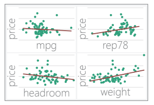
estat	**hettest**	test for heteroskedasticity
	ovtest	test for omitted variable bias
	vif	report variance inflation factor

dfbeta(length)
calculate measure of influence

Type help regress postestimation plots for additional diagnostic plots



rvfplot, yline(0)
plot residuals against fitted values



avplots
plot all partial-regression 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]   **display** _se[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
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)

Estimation with Categorical & Factor Variables

more details at http://www.stata.com/manuals14/u25.pdf

Continuous Variables
measure something

Categorical Variables
■ ● ▲ identify a group to which an observations belongs

Indicator Variables
T F denote whether something is true or false

Operator	Description	Example	
i.	specify indicators	regress price **i.**rep78	specify rep78 variable to be an indicator variable
ib.	specify base indicator	regress price **ib(**3**).**rep78	set the third category of rep78 to be the base category
fvset	command to change base	fvset base frequent rep78	set the base to most frequently occurring category for rep78
c.	treat variable as continuous	regress price i.foreign#**c.**mpg i.foreign	treat mpg as a continuous variable and specify an interaction between foreign and mpg
o.	omit a variable or indicator	regress price **io(**2**).**rep78	set rep78 as an indicator; omit observations with rep78 == 2
#	specify interactions	regress price mpg c.mpg#c.mpg	create a squared mpg term to be used in regression
##	specify factorial interactions	regress price c.mpg##c.mpg	create all possible interactions with mpg (mpg and mpg^2)

Tim Essam (tessam@usaid.gov) • Laura Hughes (lhughes@usaid.gov)
follow us @StataRGIS and @flaneuseks

inspired by RStudio's awesome Cheat Sheets (rstudio.com/resources/cheatsheets)

geocenter.github.io/StataTraining

updated June 2016

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 b = (7, 8, 9)
create a 1 x 3 matrix

matrix d = b' transpose matrix b; store in d

matrix ad1 = a \ d
row bind matrices

matrix ad2 = a , d
column 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 — ssc install mat2txt

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

Globals — 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
save \`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
r **return** list
returns a list of scalars

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

mean price
e **ereturn** list
returns list of scalars, macros, matrices and functions

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

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

generate p_mean = r(mean)
create a new variable equal to average of price

generate meanN = e(N)
create a new variable equal to obs. in estimation command

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 — ssc install estout
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 — ssc install **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

https://github.com/andrewheiss/SublimeStataEnhanced
configure Sublime text for Stata 11-14

Loops: Automate Repetitive Tasks

Anatomy of a Loop — see also **while**

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 {
    command `x', option
    ...
}
```

objects to repeat over: var1 var2 var3
x — temporary variable used only within the loop
{ — open brace must appear on first line
\`x' — requires local macro notation
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

loops repeat the same command over different arguments:

Strings

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

same as...

```
sysuse "auto.dta", clear
tab rep78, missing
sysuse "auto2.dta", clear
tab rep78, missing
```

Lists

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

```
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'
}

foreach x **of** varlist mpg weight {
summarize \`x'
}

must define list type

- foreach in takes any list as an argument with elements separated by spaces
- 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'
}

i — iterator
10(10)50 — 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) — see also **capture** and scalar _rc
trace the execution of programs for error checking

Putting it All Together — sysuse auto, clear

```
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                            — define the local i to be an iterator
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' {        — tests the position of the iterator, executes contents in brackets when the condition is true
            display "The total number of groups is `i'"
        }
    local i = `++i'                    — increment iterator by one
}
```

Tim Essam (tessam@usaid.gov) • Laura Hughes (lhughes@usaid.gov)
follow us @StataRGIS and @flaneuseks

inspired by RStudio's awesome Cheat Sheets (rstudio.com/resources/cheatsheets)

geocenter.github.io/StataTraining

updated June 2016