

Data Processing with Stata Cheat Sheet

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

Useful Shortcuts

clear

delete data set in memory

Ctrl + **9**

open a new .do file

Ctrl + **D** — keyboard buttons

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

Ctrl + **8**

open the data editor

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

capture log close

close the log on any existing do files

log using "\$pathlog/myDoFile.do", replace

create a new log file to record your work and results

findit estout

find the package estout to install

ssc install estout

install the package estout; needs to be done once

packages contain
extra functions that
expand Stata's toolkit

Import Data

sysuse auto2, clear

load system data (Auto data)

For many of these examples,
we use the auto2 dataset.

use "auto2.dta", clear

load the auto dataset from the current directory

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 "auto2.dta"

load data from the web

use first row in the cellrange
as the column names

Basic Syntax

All Stata functions have the same format (syntax):

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

apply the **command** across each group in **varlist1** function: what are you going to **do** to **varlists**? column to apply **command** to save output as a new variable condition: only apply the function **if** something is true apply to specific rows apply weights pull data from a file (if not loaded) 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)
concatenate (strings)

- subtract

***** multiply

/ divide

^ raise to a power

Logic

& and **<** less than

! or **~** not **<=** less than or equal

| or **>** greater than

== equal **>=** greater or equal

!= or **~=** not equal

Data Types

	ex.	range	checking type	converting
byte	1	0 or 1	checking type	converting
int	-1	-32,767 -32,740	R: is.int(x)	R: int(x)
long	62,134	~ -2B +2B	checking type	converting
float	5.14	range	checking type	converting
double	5.14	range	checking type	converting
string	"hello"		checking type	converting
missing	.		checking type	converting

Explore Data

VIEW HOW DATA ARE ORGANIZED

desc 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

List variables matching name patterns or other characteristics

isid mpg

check if mpg uniquely identifies the data

confirm numeric variable price make

check that price and make are numeric

mdesc

view how many observations are missing for each variable

SEE HOW DATA ARE DISTRIBUTED

codebook make price

overview of variable type, stats (range, mean, stdev, percentiles), number of missing or unique values

summarize make price mpg

Print summary statistics (mean, stdev, min, max) for selected variables

inspect mpg

display histogram of data and number of missing/zero/positive/negative observations

BROWSE OBSERVATIONS WITHIN THE DATA

browse or **Ctrl** + **8**

open the data editor

display price[4]

display the fourth observation in price.

doesn't work with vectors (more than one value)

Summarize Data

tabulate rep78, mi gen(repairRecord) include missing values save binary variables for each category in a new variable, repairRecord

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

tabulate rep78 foreign, mi

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

bysort rep78: tabulate foreign

for each value of rep78, apply the command tabulate foreign

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

Create compact table of summary statistics

collapse (mean) price (max) mpg, by(foreign)

calculate mean price and max mpg by car type. * Replaces all the data

Create New Variables

generate mpgSquared = mpg * mpg TRANSFORM EXISTING COLUMNS

generate byte lowPrice = price < 4000

create or change contents of a variable. Useful also for creating binary

egen unique_ID = group(var1 var2...)

create a unique id from a combination of variables

pctile mpgQuartile = mpg, nq = 4

create quartiles of the mpg data

clonevar mpg2 = mpg

create a unique id from a combination of variables

rename (rep78 foreign) (repairRecord carType)

variable name	storage type	display format	value label	variable label
make	str18	%-18s		Make and Model
price	int	%8.0gc		Price

```
price
type: numeric (int)
range: (3291,15906) units: 1
unique values: 74 missing.: 0/74
mean: 6165.26
std. dev: 2949.5
percentiles: 10% 25% 50% 75% 90%
3895 4195 5006.5 6342 11385
```

Variable	Obs	Mean	Std. Dev.	Min	Max
make	0				
price	74	6165.257	2949.496	3291	15906
mpg	74	21.2973	5.785503	12	41

mpg	Missing	mpg	Total	Integers	Nonintegers
1	0	1	1	1	0
2	0	2	2	2	0
3	0	3	3	3	0
4	0	4	4	4	0
5	0	5	5	5	0
6	0	6	6	6	0
7	0	7	7	7	0
8	0	8	8	8	0
9	0	9	9	9	0
10	0	10	10	10	0
11	0	11	11	11	0
12	0	12	12	12	0
13	0	13	13	13	0
14	0	14	14	14	0
15	0	15	15	15	0
16	0	16	16	16	0
17	0	17	17	17	0
18	0	18	18	18	0
19	0	19	19	19	0
20	0	20	20	20	0
21	0	21	21	21	0
22	0	22	22	22	0
23	0	23	23	23	0
24	0	24	24	24	0
25	0	25	25	25	0
26	0	26	26	26	0
27	0	27	27	27	0
28	0	28	28	28	0
29	0	29	29	29	0
30	0	30	30	30	0
31	0	31	31	31	0
32	0	32	32	32	0
33	0	33	33	33	0
34	0	34	34	34	0
35	0	35	35	35	0
36	0	36	36	36	0
37	0	37	37	37	0
38	0	38	38	38	0
39	0	39	39	39	0
40	0	40	40	40	0
41	0	41	41	41	0
42	0	42	42	42	0
43	0	43	43	43	0
44	0	44	44	44	0
45	0	45	45	45	0
46	0	46	46	46	0
47	0	47	47	47	0
48	0	48	48	48	0
49	0	49	49	49	0
50	0	50	50	50	0
51	0	51	51	51	0
52	0	52	52	52	0
53	0	53	53	53	0
54	0	54	54	54	0
55	0	55	55	55	0
56	0	56	56	56	0
57	0	57	57	57	0
58	0	58	58	58	0
59	0	59	59	59	0
60	0	60	60	60	0
61	0	61	61	61	0
62	0	62	62	62	0
63	0	63	63	63	0
64	0	64	64	64	0
65	0	65	65	65	0
66	0	66	66	66	0
67	0	67	67	67	0
68	0	68	68	68	0
69	0	69	69	69	0
70	0	70	70	70	0
71	0	71	71	71	0
72	0	72	72	72	0
73	0	73	73	73	0
74	0	74	74	74	0
75	0	75	75	75	0
76	0	76	76	76	0
77	0	77	77	77	0
78	0	78	78	78	0
79	0	79	79	79	0
80	0	80	80	80	0
81	0	81	81	81	0
82	0	82	82	82	0
83	0	83	83	83	0
84	0	84	84	84	0
85	0	85	85	85	0
86	0	86	86	86	0
87	0	87	87	87	0
88	0	88	88	88	0
89	0	89	89	89	0
90	0	90	90	90	0
91	0	91	91	91	0
92	0	92	92	92	0
93	0	93	93	93	0
94	0	94	94	94	0
95	0	95	95	95	0
96	0	96	96	96	0
97	0	97	97	97	0
98	0	98	98	98	0
99	0	99	99	99	0
100	0	100	100	100	0
101	0	101	101	101	0
102	0	102	102	102	0
103	0	103	103	103	0
104	0	104	104	104	0
105	0	105	105	105	0
106	0	106	106	106	0
107	0	107	107	107	0
108	0	108	108	108	0
109	0	109	109	109	0
110	0	110	110	110	0
111	0	111	111	111	0
112	0	112	112	112	0
113	0	113	113	113	0
114	0	114	114	114	0
115	0	115	115	115	0
116	0	116	116	116	0
117	0	117	117	117	0
118	0	118	118	118	0
119	0	119	119	119	0
120	0	120	120	120	0
121	0	121	121	121	0
122	0	122	122	122	0
123	0	123	123	123	0
124	0	124	124	124	0
125	0	125	125	125	0
126	0	126	126	126	0
127	0	127	127	127	0
128	0	128	128	128	0
129	0	129	129	129	0
130	0	130	130	130	0
131	0	131	131	131	0
132	0	132	132	132	0
133	0	133	133	133	0
134	0	134	134	134	0
135	0	135	135	135	0
136	0	136	136	136	0
137	0	137	137	137	0
138	0	138	138	138	0
139	0	139	139	139	0
140	0	140	140	140	0
141	0	141	141	141	0
142	0	142	142	142	0
143	0	143	143	143	0
144	0	144	144	144	0
145	0	145	145	145	0
146	0	146	146	146	0
147	0	147	147	147	0
148	0	148	148	148	0
149	0	149	149	149	0
150	0	150	150	150	0
151	0	151	151	151	0
152	0	152	152	152	0
153	0	153	153	153	0
154	0	154	154	154	0
155	0	155	155	155	0
156	0	156	156	156	0
157	0	157	157	157	0
158	0	158	158	158	0
159	0	159	159	159	0
160	0	160	160	160	0
161	0	161	161	161	0
162	0	162	162	162	0
163	0	163	163	163	0
164	0	164	164	164	0
165	0	165	165	165	0
166	0	166	166	166	0
167	0	167	167	167	0
168	0	168	168	168	0
169	0	169	169	169	0
170	0	170	170	170	0
171	0	171	171	171	0
172	0	172	172	172	0
173	0	173	173	173	0
174	0	174	174	174	0
175	0	175	175	175	0
176	0	176	176	176	0
177	0	177	177	177	0
178	0	178	178	178	0
179	0	179	179	179	0
180	0	180	180	180	0
181	0	181	181	181	0
182	0	182	182	182	0
183	0	183	183	183	0
184	0	184	184	184	0
185	0	185	185	185	0
186	0	186	186	186	0
187	0	187	187	187	0
188	0	188	188	188	0
189	0	189	189	189	0
190	0	190	190	190	0
191	0	191	191	191	0
192	0	192	192	192	0
193	0	193	193	193	0
194	0	194	194	194	0
195	0	195	195	195	0
196	0	196	196	196	0
197	0	197	197	197	0
198	0	198	198	198	0
199	0	199	199	199	0
200	0	200	200	200	0
201	0	201	201	201	0
202	0	202	202	202	0
203	0	203	203	203	0
204	0	204	204	204	0
205	0	205	205	205	0
206	0	206	206	206	0
207	0	207	207	207	0
208	0	208	208	208	0
209	0	209	209	209	0
210	0	210	210	210	0
211	0	211	211	211	0
212	0	212	212	212	0
213	0	213	213	213	0
214	0	214	214	214	0
215	0	215	215	215	0
216	0	216	216	216	0
217	0	217	217	217	0
218	0	218	218	218	0
219	0	219	219	219	0
220	0	220	220	220	0
221	0	221	221	221	0
222	0	222	222	222	0
223	0	223	223	223	0
224	0	224	224	224	0
225	0	225	225	225	0
226	0	226	226	226	0
227	0	227	227	227	0
228	0	228	228	228	0
229	0	229	229	229	0
230	0	230	230	230	0
231	0	231	231	231	0
232	0	232	232	232	0
233	0	233	233	233	0
234	0	234	234	234	0
235	0	235	235	235	0
236	0	236	236	236	0
237	0	237	237	237	0
238	0	238	238	238	0
239	0	239	239	239	0
240	0	240	240	240	0
241	0	241	241	241	0
242	0	242	242	242	0
243	0	243	243	243	0
244	0	244	244	244	0
245	0	245	245	245	0
246	0	246	246	246	0
247	0	247	247	247	0
248	0	248	248	248	0
249	0	249	249	249	0
250	0	250	250	250	0
251	0	251	251	251	0
252	0	252	252	252	0
253	0	253	253	253	0
254	0	254	254	254	0
255	0	255	255	255	0
256	0	256	256	256	0
257	0	257	257	257	0
258	0	258	258	258	0
259	0	259	259	259	0
260	0	260	260	260	0
261	0	261	261	261	0
262	0	262			