Data Processing with Stata 15 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

dir *.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

ssc install mdesc
install the package mdesc; needs to be done once

(packages contain extra commands that expand Stata's toolkit)

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

import excel "yourSpreadsheet.xlsx", **/***
***/ sheet**("Sheet1") **cellrange**(A2:H11) **firstrow**
import an Excel spreadsheet

(frequently used commands are highlighted in yellow)

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 commands 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
! or **~** not
| or

== equal (== tests if something is equal, = assigns a value to a variable)
!= or **~=** not equal
< 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

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

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, ask whether the value is less than ".")

list make price **if** price > 10000 & !**missing**(price)
list the make and price for observations with price > $10,000

clist ... (compact form)

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

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

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

recast double mpg
generic way to convert between types

Summarize Data

tabulate rep78, **mi gen**(repairRecord)
(include missing values; create binary variable for every rep78 value in a new variable, 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

table foreign, **contents**(mean price sd price) f(%9.2fc) **row**
(displays stats for all data; formats numbers)
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)

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

Data Transformation with Stata 15 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

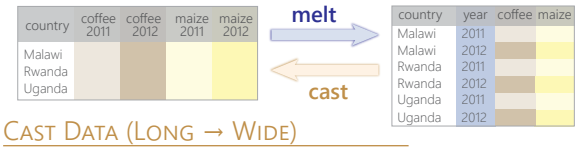
Reshape Data

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

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

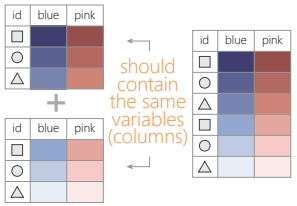
reshape wide coffee maize**, i(**country**) j(**year**)**
convert a long dataset to wide

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

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

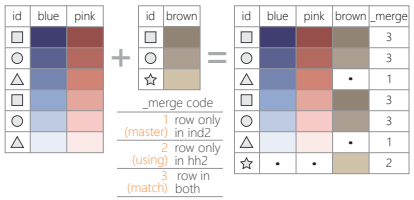
append using "coffeeMaize2.dta"**, gen(**filenum**)**
add observations from "coffeeMaize2.dta" to current data and create variable "filenum" to track the origin of each observation

Merging Two Datasets Together

must contain a common variable (id)

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 string of 5 characters starting with position 3

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

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

geocenter.github.io/StataTraining

updated June 2016