Basics in Stata

– Fachschaftsseminar für Bachelor –

Felix Albrecht

June 15, 2016

Course Goals and Outline

Course Goals

Giving you . . .

1. an introduction into the fundamentals of using Stata.
2. a basic understanding of the workings of Stata.
3. the ability to read and program *Do* code.
4. a roadmap for how to approach a dataset using Stata.
5. practice in using Stata. (. . . a loooot of exercise.)

Course Goals II

Help you to help yourselves.

- own dataset - new problems
- other questions
- next time in maybe a year

Roadmap Friday

1. Stata as a software
2. **Do** programming language
3. Exercise

Roadmap Saturday

1. Understanding data
2. Exercise II
3. Extending Stata
4. Getting Insights from data
5. Exercise III

Stata

In General

- Statistical Software
- Common in Social Sciences (used by > 90%)
- **Do** programming language
  - additional mathematical language: **Mata**
- Closed Source
- Uses binary file format for data storage
- Current version: 14
- Available to students in Bonn: 13

Versions of Stata

	Max. no. of variables	Max. no. of right-hand variables	Max. no. of observations
Stata/MP	32,767	10,998	20 billion*
Stata/SE	32,767	10,998	2.14 billion
Stata/IC	2,047	798	2.14 billion
Small Stata	99	98	1,200

Do Programming Language

Example

```
log using "LOG/example.log", replace
use "DATA/example.dta", clear

/* data descritives */
describe
su contrib PUN ID session cell  /// mean, min ,max ...
tab contrib if ID == 2

/* histogram and line graphs */
hist contrib
lgraph contrib cell session, title("Line graph contribution")

/* initial regression analysis */
reg contrib cell, cluster(ID)
log close
```

Comments

```
* single line type 1

/// single line type 2

/* multi line
comment */
```

- used for code documentation
- ignored by Stata when running code

Commands - Syntax

```
command var [var2 ...] condition , options
```

- **command** you want to execute
- **var** = object you want to work on (usually variables)

- **condition** = if else conditions
- **options** = comma indicates start of command options

Commands - *Examples*

```
/// cross tabulate contrib and PUN + show missings
tab contrib PUN, m


/* load example data file from folder DATA
+ after clearing data matrix*/
use "DATA/example.dta", clear


/* linear regression of PUN on contribution for subjects
 where 18 < age < 25 using clustered SE */
regress PUN contribution if age > 18 & age < 25, cluster(ID)
```

Logic operators

- == 'equal'
- != or ~= 'not equal'
- >, < 'larger', 'smaller'
- >=, <= 'larger or equal' , "smaller or equal"
- & 'and'
- | 'and or'
- () used to 'group' logics

Logics - *Examples*

```
/// means of contributions for age older than 18
su contribution if age > 18


/// remove treatment 3 and 5 observations from dataset
```

```
drop if treatment == 3 | treatment == 5


/* keep all observations of females older then 25 and
 males younger than 18 remove the rest */
keep if (gender == "f" & age > 25) | (gender == "m" & age < 18)
```

Graphs

```
/// 2 line graphs conditional on treatment
graph twoway (connected varY varX if treatment == 1) ///
 (connected varY varX if treatment == 2), general options

/// scatter plot with fitted line
graph twoway (scatter varY varX, m(S)) (lfit varY varX)
graph export "PATH_TO/graph.eps", replace
```

- **graph twoway** - extended graphics library of Stata
- **()** - indicate separate graph elements
  - layered on top of another
- **graph export** to store graph to HDD

Data Matrix in Stata

- single matrix for all data
- completely in RAM
  - problematic for large datasets on small PCs
- Stata prevents loading a new dataset when another one is loaded
  - have to **clear** data matrix before loading new data

Saving Files

- Stata prevents replacing files by default
- overwriting has to be allowed explicitly
- option to allow replacement **replace**
  - seen in graphs slide

```
graph export "PATH_TO/graph.eps", replace
```

General Rules when Programming with Data

Part I

1. **Never change the original data!**
   - Always work on a copy
2. **Document what you do!**
   - Use comments in your script files
   - Rather a little more than too little
3. **Avoid doing steps manually > script!**

Part II

1. **Create a meaningful folder structure in the beginning**
   - number of output files quickly increases
2. **Whitespaces are evil!**
   - filesystem paths, file names > more complicated with whitspaces
   - don't work in variable names
3. **Be precise! and clear.**
   - var -> varNew -> varNewNew

Part III

1. **When you are unsure what Stata code does, try reading it out aloud.**
    - Stata syntax is very human readable.
2. **K.I.S.S**
    - Keep It Short and Simple
3. **Back up! Back up! Back up!**
    - data and work loss is the worst
    - *Hint:* versioning systems are helpful, e.g., github

Exercise I

See exercise_ 1.pdf.

But before you start:

Finding 'help' in Stata

```
help COMMAND
```

provides the manual to commands and subcommands

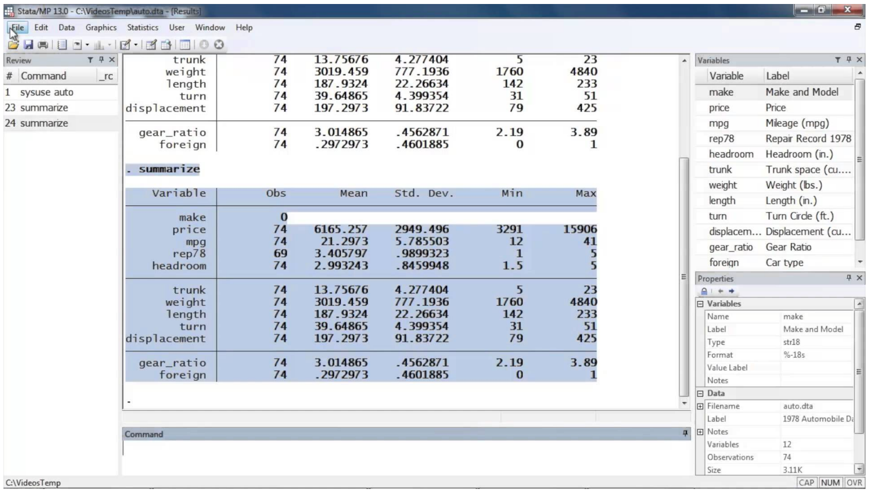
```
findit QUERY
```

searches the local and online database for your query

Online Ressources

- UCLA - Stata Help Website
- Stata Corp - Stata Graphics
- UCLA - Stata Graphics Help
- UCLA - What's the correct analysis?

Stata interface



#	Command	_rc
1	sysuse auto	
23	summarize	
24	summarize	

```
       trunk |         74    13.75676    4.277404          5         23
      weight |         74    3019.459    777.1936       1760       4840
      length |         74    187.9324    22.26634        142        233
        turn |         74    39.64865    4.399354         31         51
displacement |         74    197.2973    91.83722         79        425
-------------+---------------------------------------------------------
  gear_ratio |         74    3.014865    .4562871       2.19       3.89
     foreign |         74    .2972973    .4601885          0          1

. summarize

    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
       rep78 |         69    3.405797    .9899323          1          5
    headroom |         74    2.993243    .8459948        1.5          5
-------------+---------------------------------------------------------
       trunk |         74    13.75676    4.277404          5         23
      weight |         74    3019.459    777.1936       1760       4840
      length |         74    187.9324    22.26634        142        233
        turn |         74    39.64865    4.399354         31         51
displacement |         74    197.2973    91.83722         79        425
-------------+---------------------------------------------------------
  gear_ratio |         74    3.014865    .4562871       2.19       3.89
     foreign |         74    .2972973    .4601885          0          1

.
```

Variables

Variable	Label
make	Make and Model
price	Price
mpg	Mileage (mpg)
rep78	Repair Record 1978
headroom	Headroom (in.)
trunk	Trunk space (cu....
weight	Weight (lbs.)
length	Length (in.)
turn	Turn Circle (ft.)
displacem...	Displacement (cu...
gear_ratio	Gear Ratio
foreign	Car type

Properties

Variables	
Name	make
Label	Make and Model
Type	str18
Format	%-18s
Value Label	
Notes	
Data	
Filename	auto.dta
Label	1978 Automobile D
Notes	
Variables	12
Observations	74
Size	3.11K

Stata Uni Bonn

- CIP-Pool computers
- Personal Laptop
  - Network drive: \.jura.uni-bonn.de

Data

What you need to know.

Variable Data Types

- **Numeric** - black
  - Binary variables
  - Categorical variables
    - can be marked with 'value labels' - blue
- **Strings** - red
  - Can be stored but
- **Missing**

Binary Variables

- alternatively called 'Dummies'
- represent 'Yes' / 'No' cases

```
gen impossible = (speed > 300000)
```

speed	impossible
400000	1
3000	0
60	0

Categorical Variables

- describes different categories
  - e.g. survey answers

```
gen risk = 1 if survey == "dislike strongly"
replace risk = 2 if survey == "dislike moderately"
replace risk = 3 if survey == "neither nor"
replace risk = 4 if survey == "like moderately"
replace risk = 5 if survey == "like strongly"
```

survey	risk
"dislike strongly"	1
"neither nor"	3

Strings

- Need to be converted to numeric variable
- Stata has Regex support
- When working with strings you need to use: "string"

```
/// when importing from CSV
destring _all

/// when conversions
tostring VAR_NAME

/// replacing in string
replace VAR = subinstr(VAR,QUERY,REPLACEMENT,.)
```

Missing

- ‘missing observations‘
  - ‘empty‘ cells in data matrix
- represented by `.` (dot)
- very large number
  - consider this for conditions
    - e.g. if var $>$ 1,000,000 is **true for missing**

Digression - Interactions

- measures additional effects
- combination of binary variable and 2nd variable

```
/// continuous var and dummy - slope change
gen IConAge = contrib * DOldAge

/// dummy and dummy - level change
gen IAgeGen = gender * DOldAge
```

contrib	gender	DOldAge	IConAge	IAgeGen
5	1	0	0	0
10	0	1	10	0
10	1	1	10	1

Data Types

Type	Obs per Unit	Time	indicate
Cross-Section	*many*	one	default
Time-Series	one	*many*	tsset TIME-VAR
Panel	*many*	*many*	xtset UNIT TIME

Cross-Section *Example*

country	year	gdp
Germany	1990	1.756
France	1990	1.275
UK	1990	1.067

Time-Series *Example*

country	year	gdp
Germany	1990	1.756
Germany	1991	1.862
Germany	...	...
Germany	2013	3.73

Panel *Example*

country	year	gdp
Germany	1990	1.756
Germany	1991	1.862
France	1990	1.275
France	1991	1.276
UK	1990	1.067
UK	1991	1.116

Panel Data Table Formats

WIDE

- observations are stored column-wise

LONG

- observations are stored row-wise

Stata works with data in long format

WIDE Format *Example*

country	gdp1990	gdp1991	gdp1992	...	gdp2013
Germany	1.756	1.862	2.123	...	3.73
France	1.275	1.276	1.409	...	2.806
UK	1.067	1.116	1.158	...	2.678

LONG Format *Example*

country	year	gdp
Germany	1990	1.756
Germany	1991	1.862
Germany	...	...
Germany	2013	3.73
France	1990	1.275
France	1991	1.276
France	...	...
France	2013	2.806
UK	1990	1.067
UK	1991	1.116

Conversion command

```
/// reshape to long from wide
reshape long gdp, i(country) j(year)

/// reshape to wide from long
reshape wide gdp, i(country) j(year)
```

Joining dataset vertically

```
append using dataset2.dta
```

- adds observations at the bottom of the data matrix
- joins by variable name
- variable names are case sensitive
  - *e.g.* Contrib is not contrib
- creates new variables if non-existent

Joining datasets horizontally

```
merge m:n identifiers using dataset2.dta
```

- matches observations based on identifiers
- m:n = relationship between obs. in datasets (master:new)
  - *1:1 = one to one*
  - *1:n / m:1 = one to many*
    - e.g. serveral obs. per subject in m merged with age and gender data from n
  - *m:n = many to many*
    - e.g. both datasets have several obs. per subject but they don't match perfectly

Reducing dataset

subject	month	income	age
101	1	1000	25
101	2	1100	25
102	1	500	21
102	2	600	21

```
collapse income age, by(subject) /// default is mean
```

subject	income	age
101	1050	25
102	550	21

Making changes reversable

```
/// start
preserve
/// put your code here
drop if age > 20

/// end
restore
```

- stores current state of dataset
- you can apply changes
- *restore* saved dataset
- you can only *preserve* one dataset at any given time

Exercise II

See exercise_ 2.pdf.

AddOns

Installation

```
ssc install pluginName
```

- Stata has an ‘App store‘
  - user written extensions for specific tasks

To find functions if you don’t know the module’s name use *findit*.

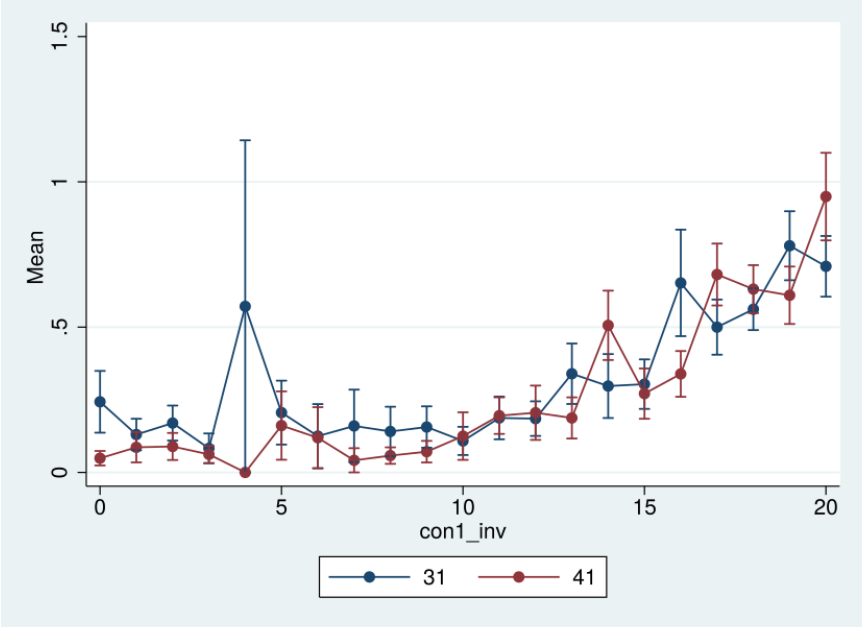
Recommendations

lgraph

"quick linegraphs with conditional grouping and errorbars"

```
lgraph PUN con1_inv treatment, errortype(se)
```

lgraph *Example*

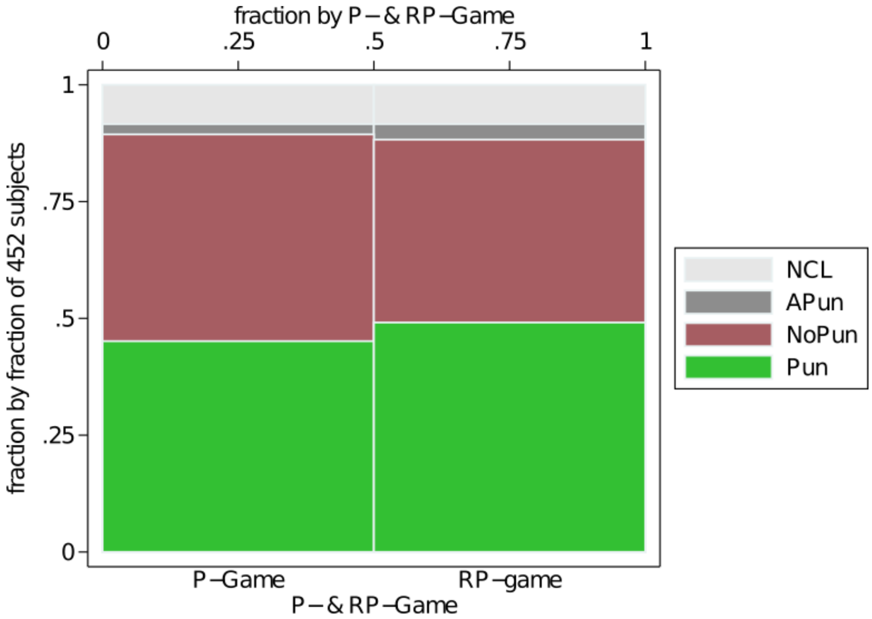


spineplot

"graph to compare categorical variables across treatments"

```
spineplot t_con1_inv31 treatment
```

spineplot *Example*



outreg2

- publication ready regression tables
- puts in significance stars
- sorts the table content

```
xtreg contrib l1.realPunRecSum, cluster(sid)
outreg2 using "PATH_TO/output.xls", replace

xtreg realPunRecSum predPunRecSum31, cluster(sid)
outreg2 using "PATH_TO/output.xls", replace

xtreg contrib l1.realPunRecSum l1.residual1, cluster(sid)
outreg2 using "PATH_TO/output.xls", append addstat("F test model", e(F))
```

outreg2 *Example*

	(1)	(2)	(3)
VARIABLES	contrib	realPunRecSum	contrib
L.realPunRecSum	-0.190***		-1.067***
	(0.0321)		(0.0532)
predPunRecSum31		1.181***	
		(0.0281)	
L.residual1			1.285***
			(0.0640)
Constant	13.02***	0.347***	14.23***
	(0.0799)	(0.0374)	(0.0967)
Observations	3,852	4,280	3,852
R-squared	0.010	0.314	0.114
Number of subjects	428	428	428
F test model			221.1

(xt)ivreg2

- enhanced instrumental variable regression
- and panel version as well

Getting Insights from Data

Correlation

Linear Pearson White

```
pwcorr var1 var2, sig
```

Non-Parametric Spearman Rank Correlation

```
spearman var1 var2
```

Tests

Normally Distributed T-Test

```
ttest var1 = var2
```

Non-Parametric Sign-Rank (Within subject)

```
signrank var1 = var2
```

Non-Parametric Ranksum (Between subject)

```
ranksum var, by(treatment)
```

Regression Analysis - CS

```
/// linear regression with cluster robust standard error
reg dependentVar descriptiveVar1 ..., cluster(id)
```

- dependent variable is continuous and vars a iid
- **cluster robust** - standard errors are calculated separately for each clusters
  - larger SEs -> lower probability for signifcance

Special case

- dependent variable is binary
  - "Linear Probability Model"

Regression Analysis -CS II

```
/// maximum likelihood for binary dependent var
logit dependentVar descriptiveVar1 ...

probit dependentVar descriptiveVar1 ...

/// calculates effect sizes dependent on mean
margins
```

- for binary dependent variable
- values are not directly interpretable
- signs are
- **margins** calculates effect sizes at mean

Regression Analysis - Panel

```
/// define panel
xtset id year

/// panel estimation with unit fixed effects and cluster robust SEs
xtreg dependent descriptive1 ..., cluster(id) fe
```

Accessing additional statistics I

- descriptive statistics & tests store in **vector r()** ‘results‘

```
. su contrib

Variable |  Obs        Mean    Std. Dev.  Min   Max
---------+-----------------------------------------
 contrib | 7480    10.82821    7.363504     0    20

. return list

scalars:
  r(N) =  7480
  r(sum_w) =  7480
   r(mean) =  10.82820855614973
r(Var) =  54.22118571930103
 r(sd) =  7.363503630697891
r(min) =  0
r(max) =  20
r(sum) =  80995
```

```
di r(mean)
10.828209
```

Accessing additional statistics II

- estimators (regressions) store in **vector e()** ‘estimates‘

```
reg PUN con1, cluster(sid)

Linear regression                 Number of obs =    7480
  F(  1,   747) =   37.61
  Prob > F      =  0.0000
  R-squared     =  0.0134
  Root MSE      =  .93068

(Std. Err. adjusted for 748 clusters in sid)
-----------------------------------------------------------------------
 Robust
  PUN |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
------+----------------------------------------------------------------
 con1 |  -.0174465   .0028447    -6.13   0.000    -.0230311   -.0118619
_cons |   .5994308   .0438951    13.66   0.000     .5132583    .6856034
-----------------------------------------------------------------------

di e(rmse)
.93067724
```

What is this good for?

- quick calculations

```
di e(r2) - e(r2_a)
.00013194
```

- adding statistics to outreg2 tables

```
reg PUN con1, cluster(sid)
outreg2 using "PATH_TO/file.xls", replace addstat("RMSE", e(rmse))
reg PUN con1 con2, cluster(sid)
outreg2 using "PATH_TO/file.xls", replace addstat("RMSE", e(rmse))
```

Exercise III

See exercise_ 3.pdf.

Final Remarks

Stata Pros

- Stata is very good for data crunching for dataset of considerable size
- Stata syntax is great, easy to learn *(I hope you did.)*, very readable
- The way Stata is programmed it belongs to the most efficient data tools out there

Stata Cons

- Considerable cost
- Not good for big data as all is done in the RAM
- Automation *(not covered here)* possibilities within Stata are good and easy to learn BUT are non-existent when it comes to 3rd party apps

Alternatives I

Commercial (Closed Source)

- **SAS** - business applications,e.g. , combined with SAP
- **SPSS** - marketing research & social sciences
- **Eviews** - marketing research, especially time series analysis

Alternatives II

Open Source

- **R-project** - all fields of statistics; 2015 commercial branch has been acquired by Microsoft - new project *Microsoft R Open* - backend for Microsoft Azure
- **Python** - allround programming language, also for Statistics (classes taught in Master in Bonn)
- **GRETL** - teaching software, quick to learn for small projects

Many more. Find a list on Wikipedia .

Take Away

I hope you ...

- learned the basics of Stata programming.

I want you to ...

- keep an open mind. The software world is rapidly changing.
- be able to look for the right tool for the job.
- be open to new solutions.

I recommend to ...

- try to understand the underlying structure of things. When you do new tools are not 'really' new.

The End

Have a recreational Sunday.

Appendix

Loading Data

Commands

- a Stata binary file *.dta

```
use "PATH_TO_FILE/FILE_NAME.dta", clear
```

- load a CSV format

```
insheet using "PATH_TO_FILE/FILE_NAME.csv", names delimiter(;) clear
```

- importing MS Excel

```
import excel using "PATH_TO_FILE/FILE_NAME.xlsx" ///
, sheet(SHEET_NAME) cellrange(START_CELL) firstrow clear
```

Command options I

- All -

- **clear** = clears data matrix

- CSV -

- **names** = first line contains variable names
- **delimiter(;)** = symbol that separates columns in CSV
  - other common possibilities are **commas** or **Tabs** ()⁀

Command options II

- Excel -

- **sheet()** = indicate which sheet to load from
- **cellrange()** = cell to extract
  - single value "B2" indicates *upper left* start
- **firstrow** = first row contains variable names

Writing Data

Commands

- a Stata binary file *.dta

```
save "PATH_TO_FILE/FILE_NAME.dta", replace
```

- a CSV format

```
outsheet using "PATH_TO_FILE/FILE_NAME.csv", ///
(nonames) delimiter(;) replace
```

Command options

- All -

- **replace** = allows overwriting files
  - Stata blocks overwriting files by default

- CSV -

- **delimiter(;)** = defines output delimiter
  - might be important for importing into other programs
- **nonames** = supresses writing variable names to csv file