Stata Introduction

Jan Marvin Garbuszus

16. April 2013

Literature

- U. Kohler und F. Kreuter (2012). *Datenanalyse mit Stata*. 4. Aufl. München: Oldenbourg Wissenschaftsverlag GmbH
- A. C. Acock (2012). *A Gentle Introduction to Stata*. 3. Aufl. Lakeway Drive, Texas: Stata Press

Introduction

Introduction Contents

Syntax

We write Syntax.

- Reproducable
- Less . . .
  - clicking
  - repeating
  - error-prone

Syntaxfiles

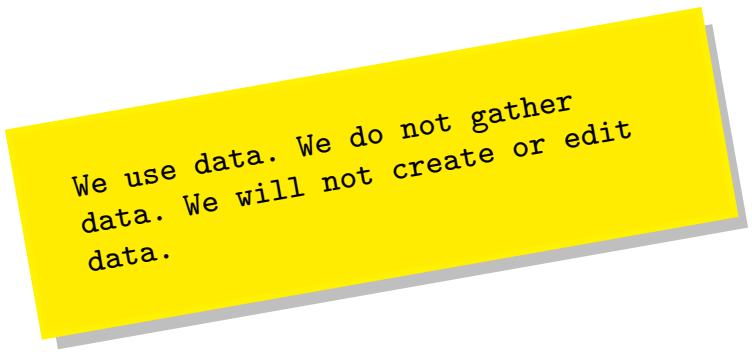
- Stata syntax is handled in do-files.
- Starting the editor

```
doedit
```

- Fileending of do-files is .do.
- do-files can be opend and modified with any texteditor.

secondary data

- primary data
- secondary data



dta-files

- Data is stored in dta-files.
- Data filending is .dta.
- Stata dataobjects are readable only by Stata and some other statistical software e.g. R (R Core Team, 2013).

Importing data

- Showing the content of a directory

```
dir
```

- Changing the directory

```
cd "D:/Data/"
```

- Importing data

```
use <dataname.dta>
```

Avoid directory- or filenames containing umlauts or whitespaces. Stata does not like those.

do-file I

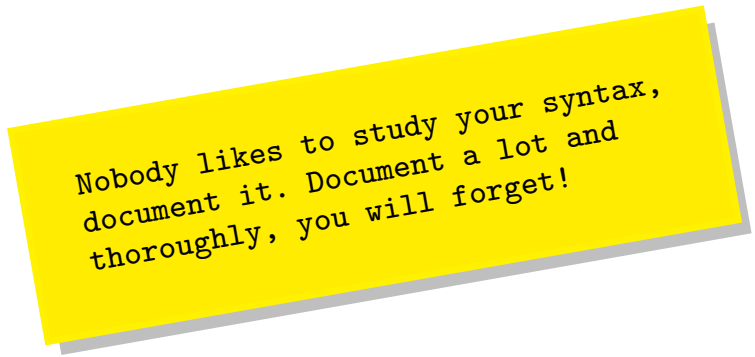
- file header: should have informations like filename, author, creation and/or modification date, a short description of its content and for longer files a short table of contents.

```
/*
Name: read-soep.do
Author: Garbuszus
Date: 2016-02-28
Content: Importing SOEP-data
*/
```

do-file la

Document your code!

- Teamwork
- Obliviousness



do-file Ib

- /* (begin) und */ (end) define a comment. Alternative this can be done at the beginning of a line with * or inline with //

```
** We write code for Stata version 8 and later
version 8 // This runs with Stata 8 and might not work with Stata 7
```

- Commands in comments will be skipped!

do-file lc

- This is followed by

```
** remove everything from our data memory
clear all
** disables more functionalty
set more off
```

clear all errases everything from Statas memory, securing, that everything in the following do files is run on exactly the data we define and not on some earlyer created data artifacts. It's reproducable!

do-file II

- Import some testdata

```
use "D:/Data/testdaten.dta"
```

- Describe

```
describe
```

- Help for command

```
help describe
```

help

`tabulate varname1 varname2 [if] [in] [weight] [, options]`

options	Description
Main	
chi2	report Pearson's chi-squared
exact[(#)]	report Fisher's exact test
...	

`regress depvar [indepvars] [if] [in] [weight] [, options]`

options	Description
Model	
noconstant	suppress constant term
hascons	has user-supplied constant
...	

findit

And how does one find commands?

```
findit describe
```

findit can be helpfull, but it's not guaranteed to find exactly what you're looking for.

do-File III

Different tasks, different do-files.

- Import data
- Preparation
- Descriptive analysis
- etc.

Not every do-file is required to be rerun all the time. For instance importing and preparation of the data should be a one time task. It's not required to do this for every single crosstable or boxplot. Time is money!

do-file IV

since do-files may be executed by other do-files nesting some of them in a master file helps with housekeeping

```
** call other do-files
do "01-Data_Import.do"
do "02-Data_Preparation.do"
do "03-Descriptive_Analysis.do"
```

This tightens your syntax and the correct order of do-files is assured.

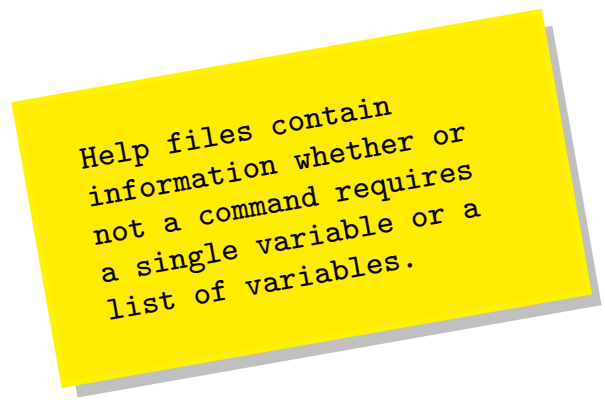
Description of data in memory

Description of data in memory

```
describe
codebook
list
browse
```

Studying describes help page (`help describe`) presents additional options like

```
describe, simple
```



Descriptive Analysis

Descriptive analysis of variables [1]

```
tabulate var1
tabulate var2
tabulate var1 var2
summarize var1
summarize var1-var4
```

Studying summarzies help page (`help summarize`) presents additional options like

```
summarize var1, detail
```

[1] `var1` and `var2` are placeholders.

Variables

Variables Contents

Generate

Time and time again generation of variables is required

```
** clone variable
clonevar female = sex

** generate variable
gen gender = .
recode gender . = 1 if sex == 1
recode gender . = 2 if sex == 2

** spot the difference
tab female
tab gender
```

Rename

Thinking about it we dislike the variable name `gender`.

```
drop sex // throw this variable away
** gender is now called sex
rename gender sex
```

Moment `generate ... = . ?`

Missing values in Stata are identified by the dot.

```
clonevar edu = school

** create a missing
recode edu 10 = .
** different missing values (Stata knows .a - .z)
recode edu 9 = .a
recode edu 8 = .b
recode edu 7 = .c
```

Additionaly the dot is $\pm\infty$ and is Statas biggest value.

A moment please ... if ... == 1 ?

if-conditions

```
** sex in county 1 and school 1 or 2
  tab sex if county == 1 & ///
    (school == 1 | school == 2)
```

The condition is following the if-statement. Different contditions can be linked using & and |. Linked conditions require variable names within each condition!

and tab ?

tab is short for

```
help tabulate twoway
tabulate varname1 varname2 [if] [ ...
```

The underlined part shows the minimum letters Stata requires for identification of a command.

Label

To label a variable, a label has to be defined and afterwards applied

```
** label a variable
label variable sex "Sex"
** define label for values - labsex is the label name
label define labsex 1 "male" 2 "female"
** assign label to variable
label values sex labsex
```

Now labels are assigned

```
tab sex
tab sex, nolabel
```

Generate II

It is possible to create variables using mathematical expressions

```
// create fantasy weight and height
replace weight = weight / 2.2
replace height = height * 0.004

// bmi for lbs and in
gen bmi = (weight / (height ^ 2))

/* other examples
** householdincome
gen hhinc = inc_female + inc_male
** age in 2016
gen age = 2016 - gebjahr
*/
```

Generate III

A new variable is created from recoded values. The original variable is not modified.

```
recode school ///
  (10=.) ///
  (1=1 "One") ///
  (2=2 "Two") ///
  (3 5 6 7 8 = 99 "Ninetynine") ///
  (4=3 "Three") ///
  (9=5 "Five"), gen (school_rec)
```

Variable names begin with a character (a-z or A-Z).

A moment please /// ?

Three following *Slashes* indicate to Stata, a command will continue in the following line.

```
** everything in a single line
display 1 + 1
** now with a (unneeded) linebreak
display 1 + ///
  1
```

This way two and more lines can be connected. This makes your syntax more readable. Else check help delimit

Rekodieren

Sometimes it is helpful to categorize metric variables

```
** create rounded values
gen bmi_r = round(bmi)
recode bmi_r ///
  (0/18.5=1) ///
  (18.5/25.0=2) ///
  (25.0/30.0=3) ///
  (30.0/60=4)

tab bmi_r
```

0/18 means from 0 to 18. 25/30 from 25 to 30. Be careful with the category range!

Data preparation

Lets have a look at our fantasy weights data. We replace a few values with different values

```
** weight
replace weight = -1 if weight < 40
replace weight = -2 if weight > 140

mean weight
```

Now we decode these values as missing values

```
mvdecode weight, mv (-1=.a\-2=.b)
mean weight
```

In the SOEP negative values are typically missing values.

Mistakes

After hours of work you figure that you have made a mistake and want to modify the just created variable `bmi_r`. What now?

- Luckily you have not touched the original data. Just run your do-file up to the point where the mistake is located.
- You already spotted the error and want to rerun the variable creation. Simply remove the variable

```
drop bmi_r
```

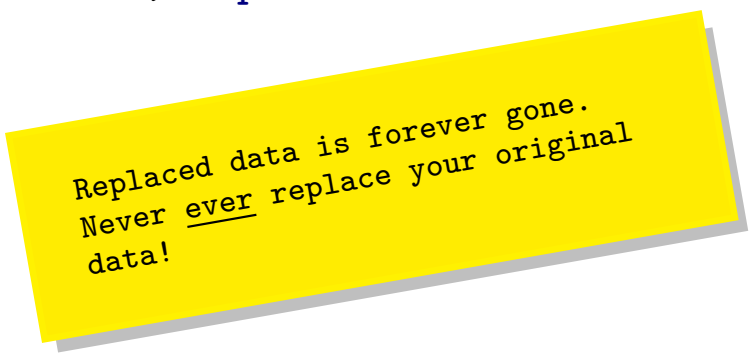
Save Data

Save your do-file into a directory on your computer (e.g. *do-files*) and your data into a different folder(e.g. *data*).

```
save "D:/Data/data/testdata_recoded.dta"
```

If a file with the same name exists, Stata does not want to erase anything by mistake and asks you for the `replace` option

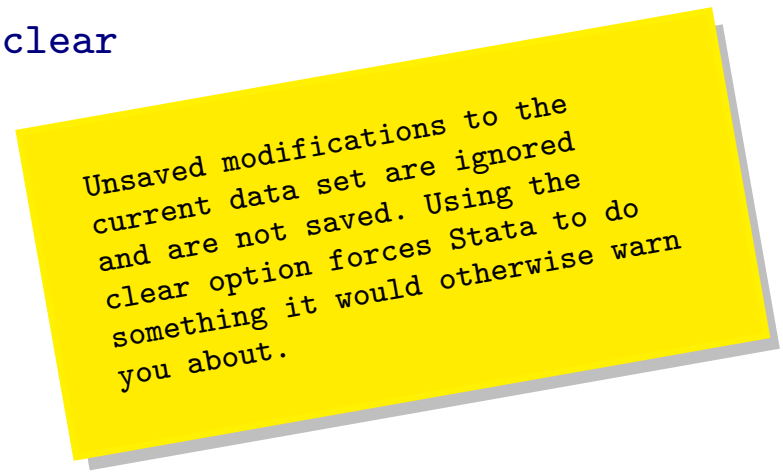
```
save "D:/Data/data/testdata_recoded.dta", replace
```



Import Data

With `use ..., clear` the currently imported data set will be replaced

```
use "D:/Data/data/testdata.dta", clear
```



Graphics

Graphics Contents

Graphics

The next slides will provide an example of different types of graphics. Detailed examples and syntax for modification of graphic title, titles for the axis or combination of multiple graphics see Kohler und Kreuter (2012, S. Chap. 6) and in-depth with lots of images Mitchell (2012).

Preface ...

Warning!

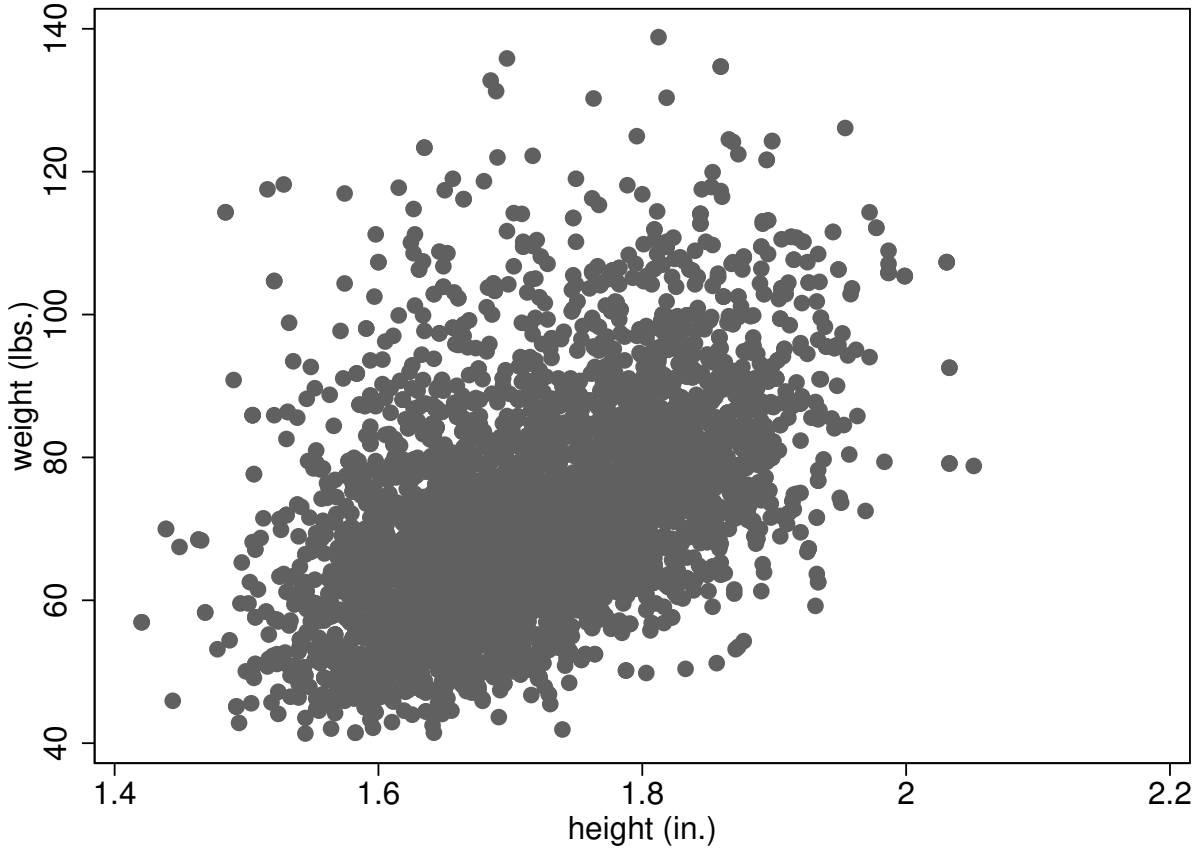
Don't do any pie, pizza, circle or however you name them charts. Just don't do it!

Additionaly

```
** otherwise we see colors
set scheme s1mono
```

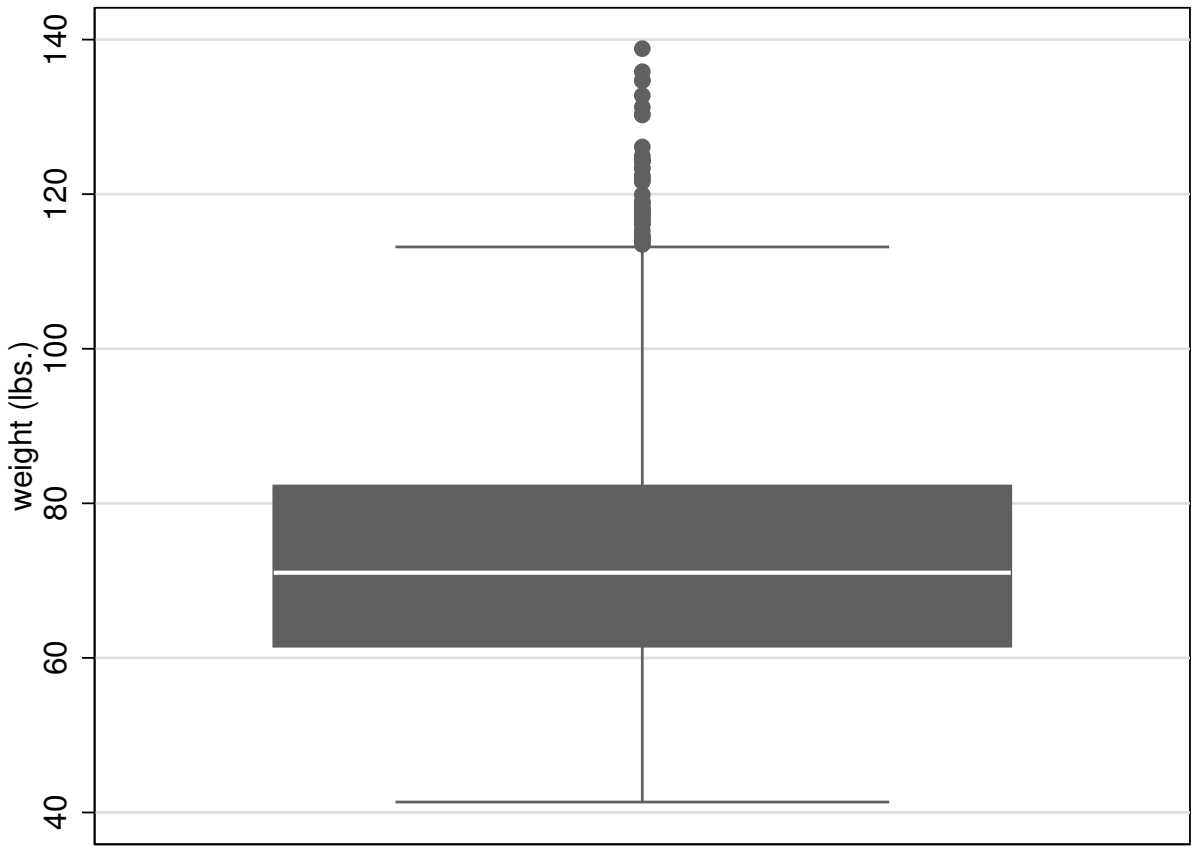
Scatterplots

```
scatter weight height
```



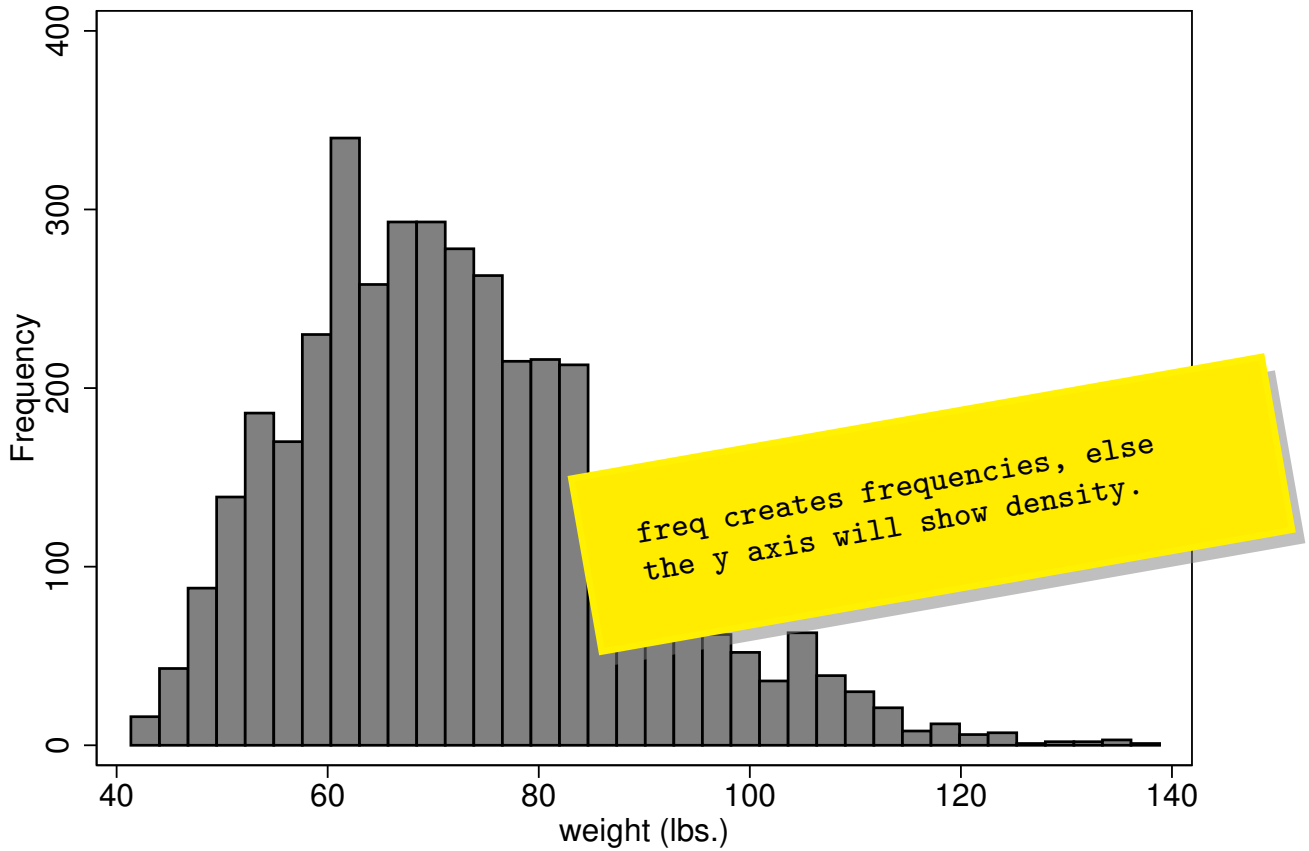
Boxplots

```
graph box weight
```



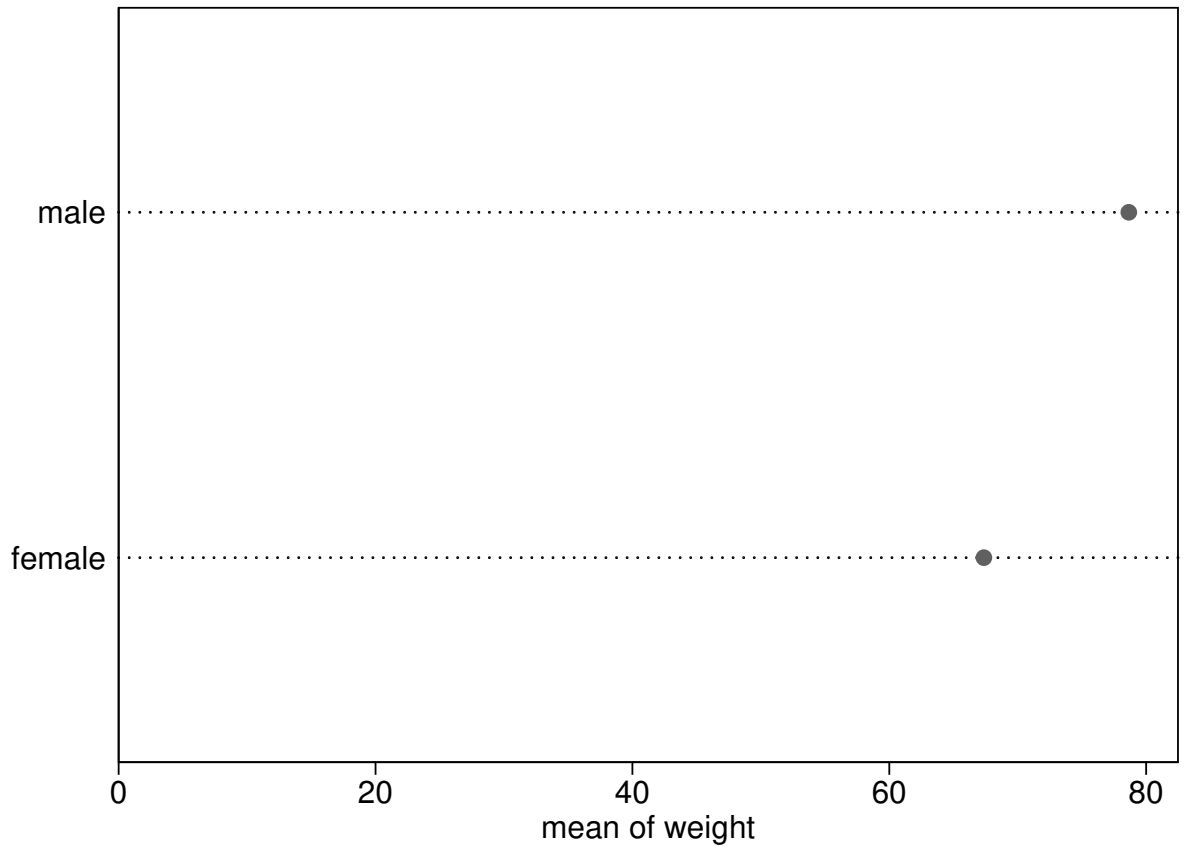
Histograms

```
hist weight, freq
```



Dot-Charts

```
graph dot (mean) weight, over(sex)
```



Export of Graphics

```
graph export "D:/Data/graphics/graph1.pdf", replace
```

- For export there are many different formats
- .ps .pes .wmf .emf .pdf .png .tif
- For usage in MS-Office a good choice is .png
- For usage in TeX a good choice is either .pdf or .eps

Improved Datahandling

Improved Datahandling Contents

by I

Select subsets

```
summarize weight if sex == 1
summarize weight if sex == 2
```

Do it in a single line

```
by edu, sort: summarize hhinc
```

```
by is not allowed with every
command, in doubt contact the
help. To use by cases have to be
sorted. Alternativ: bysort
```

by II

It is possible to use more than one variable

```
by sex county, sort: summarize weight
```

in

Select only a few cases

```
** select the first
list sex in 1
** select the first ten
list sex in 1/10
```

Usually single cases are not interesting, but this can be helpful to spot strange cases.

Loops I

Loops in Stata. Example from Kohler und Kreuter (2012, S. 69f.)

```
** Generate variables r1 - r10
** Newlist
foreach var of newlist r1-r10 {
  gen `var' = runiform()
}

** Numeric list
foreach num of numlist 1/10 {
  replace r`num' = runiform()
}
```

Loops II

```
** multiple commands in a loop
foreach var of varlist ybirth income {
  summarize `var', meanonly
  generate `var'_c = `var' - r(mean)
  label variable `var'_c "`var' (centered)"
}

** Forvalues
forvalues num = 1/10 {
  replace r`num' = runiform()
}
```

log-files

- Results can be stored in log-files.
- These contain all commands and their results.
- Logging begins after

```
log using <dateiname.log>
```

- log-files can be opened with every texteditor.

log-files II

- Define a name and path where the log-file is saved. Usually the log-file name should reflect the do-files name.

```
log using read-soep.log, replace
```

- At the end of the do-file

```
** commands will not be logged after this
log close
```

- Thrown errors stop the do-files execution, therefore a manual closing of the log-file might be required because Stata cannot handle two open log-files

```
log close
```

```
capture log close atop log using
ensures that all open log-files
will be closed.
```

Macros

With longer pathes and bigger projects it is helpful to use so called Macros. Macros may be global (it is known everywhere inside your code) or local (it is known only in a specific loop). We use macros for paths

```
** global Macroname Pathname
global DATA "D:/Data/data/original/"
global OUT "D:/Data/data/bearbeitet/"
```

Now the macros can be called like this

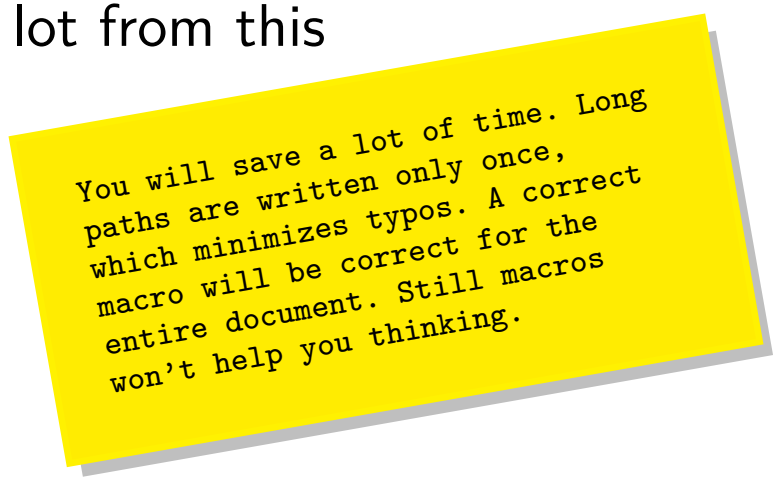
```
cd "${DATA}"
```

Macros II

It is especially helpfull handling long or different directorys for input, output, images or log files. Some additional examples

```
use "${DATA}testdata.dta", clear
save "${OUT}testdata_recoded.dta", replace
```

The syntax readability benefits a lot from this



using

Stata can handle only a limited range of variables, so a little housekeeping is advised.[2] Handling the SOEP this limit is not as far away as you might believe.

```
** Import a minimal ppfad
** kepp only hhnr persnr sex gebjahr
use hhnr persnr sex gebjahr using "${DATA}ppfad.dta"
** Sort cases by persnr and gebjahr
sort persnr gebjahr
```

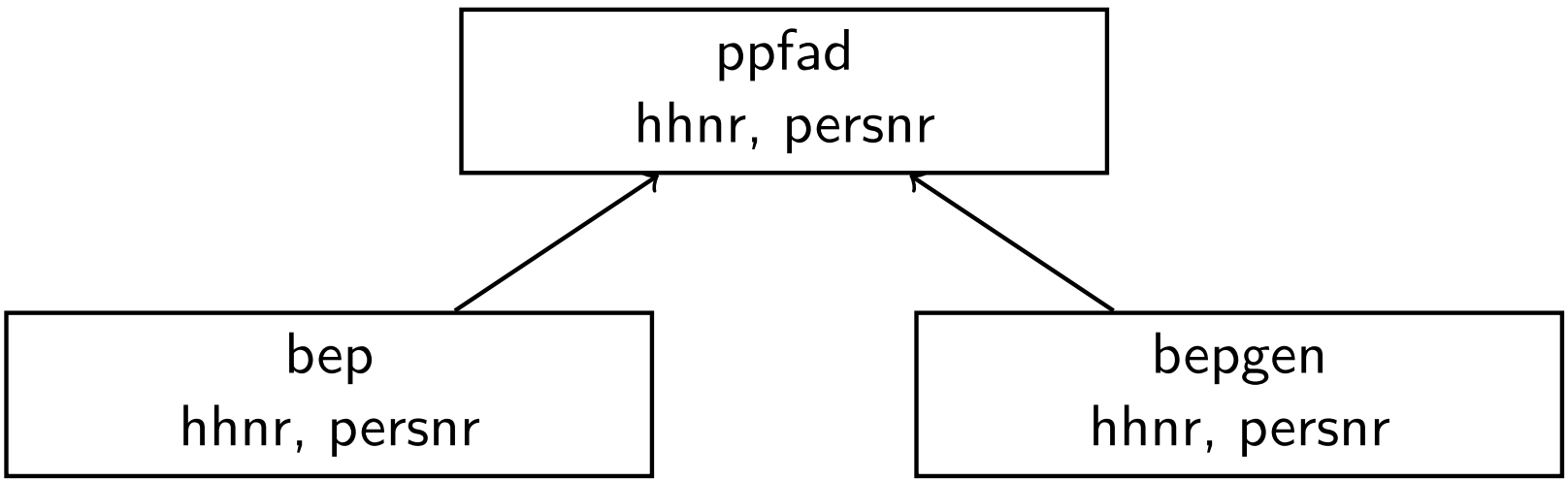
[2] Different Stata versions have different limits: IC (the version your working with) can handle 2,047, SE can handle 32,767 variables. Handling more variables is rather expensive. Prices for 2013: $189 and $395 for students. Non academic single user licenses $1545 and $2090.

merge

In the Sozio-oekonomischen Panel (SOEP) (s. Wagner et al., 2007) each years data is split into different smaller data files. This has historical and maybe even practical reasons.
Starting from a file different of these data files share, we can combine the smaller files to generate a complete data set. This combination is called *merging*.

merge II

As example: We merge **ppfad** with **bep** and **bepgen**. ppfad is the key file for the person or p-context. **ppfad** is the data file every p-file shares. To make sure that we combine our data correctly for the p-context, we seek for matches in household-id (hhnr) and person-id (persnr).



merge III

Running `merge`

```
** merge ppfad with 2014 bep
merge 1:1 persnr using "${DATA}bep.dta"
```

Result	# of obs.	
not matched	25,834	
from master	25,834	(_merge==1)
from using	0	(_merge==2)
matched	10,471	(_merge==3)

This means

- $25,834$ cases of `ppfad` were not found in `bep`.
- 0 cases of `bep` were not found in `ppfad`
- $10,471$ cases of `bep` were found in `ppfad`

merge IV

A new variable _merge is created containing the _merge== values. For now we remove all cases where we could not find a match from using which is _merge==2. These are potentially problematic as we have no information about these persons sex or year of birth. Luckily at our current stage these are zero cases. Afterwards we remove _merge.

```
** Drop if using does not match master
drop if _merge==2
** drop _merge
drop _merge
```

The coice which merge is dropped is depending on the analysis.

merge V

Example: merge with the generated variable highest educational achievement `bepsbil` from data-file **bepgen**.

```
merge 1:1 hhnr persnr using "${DATA}bepgen.dta", keepusing(bepsbil)
drop if _merge==2
drop _merge
```

bepgen contains 65 variables, but all we need is the educational achievement.

Operators

Stata lists the following operators, which can be used with variables. Using `var1^2` every value of `var1` will be squared.

```
== // equals
!= // not equal ~=
& // and
| // or
! // not
< // smaller
> // greater
<= // smaller equal
>= // greater equal
+ // plus
- // minus
/ // divide
* // mutiply
^ // power
```

Functions

Additionaly Stata contains a range of built in functions (`help functions`). A selection:

```
abs() // absolute value/modulus |-2| == 2
max() // greates value
min() // smalles value
exp() // e-function
ln() // logarithm
round() // round
sin() // Sinus
sqrt() // square root
runiform() // random number
```

Statistics

Statistics Contents

Statistic I

Arithmetic mean and some additional statistics

```
mean height
summarize height
summarize height, detail
```

mean returns the standard error: $s/\sqrt{n}$. summarize returns the standard deviation: $\sqrt{\sum(x-\bar{x})^2)/n}$.

Statistic II

Minimum, maximum, arithmetic mean, median, number of observations and quartils.

```
tabstat height, statistic(min max mean median p50)
tabstat height, statistic(min max range mean count q) by(county)
```

```
range = max - min
```

Statistics III

Standard deviation, standard error, variance und interquartil range.

```
tabstat height, statistic(sd sem var q iqr)
```

Skewness and kurtosis

```
tabstat height, statistics(skewness kurtosis)
```

```
iqr = q3 - q1
```

Tables

We already had `summarize` and `tab`, let us combine them

```
tab county, summarize(height)
```

Cross tabs

Create a cross tab with

```
tab school county
```

And some more tables

```
** tabs of each variable
tab1 sex county school
** cross tabs of ab bc ac
tab2 sex county school
```

Cross tabs II

Let us recapitulate Statistics I, we will do some tests [3]

```
** chi^2
tab sex county, chi
** cramers v
tab sex county, V

** manual
tab sex county, exp col row
** phi and for 2x2 tabs V
di (1013*1002 - 925*1131) / (2144*1927*1938*2133)^(1/2)
** chi^2
di 4071 *(-.00753735)^2
** usual way to compute V
di (.23128021 / 4071 * (2 - 1) )^(1/2) // V
```

[3] or see e.g. Kühnel und Krebs (2010) oder Agresti und Finlay (2009).

Numlabel

You can add numeric values to your variable labels

```
** numeric labels on for all variables
numlabel _all, add
** and off
numlabel _all, remove
```

T-Test

And finally let us have a look at some test-statistics

```
** compare
tab sex, sum(bmi)
ttest bmi, by(sex)
```

And much more to come . . .

Regression

- `regress`
- `logit`
- `mlog`

To be continued.

Index and Literature

Index

Literature I

- Acock, A. C. (2012). *A Gentle Introduction to Stata*. 3. Aufl. Lakeway Drive, Texas: Stata Press.
- Agresti, A. und B. Finlay (2009). *Statistical Methods for the Social Science*. 4. Aufl. London: Pearson.
- Kohler, U. und F. Kreuter (2012). *Datenanalyse mit Stata*. 4. Aufl. München: Oldenbourg Wissenschaftsverlag GmbH.
- Kühnel, S. M. und D. Krebs (2010). *Statistik für die Sozialwissenschsaften. Grundlagen, Methoden, Anwendungen*. 5. Aufl. Reinbek: Rowohlt.
- Mitchell, M. N. (2012). *A Visual Guide to Stata Graphics*. 3. Aufl. Lakeway Drive, Texas: Stata Press.
- R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria.

Literature II

Wagner, G. G., J. R. Frick und J. Schupp (2007). *The German Socio-Economic Panel Study (SOEP): Scope, Evolution and Enhancements*. SOEPpapers on Multidisciplinary Panel Data Research 1. DIW Berlin, The German Socio-Economic Panel (SOEP).