Stata Einführung

Jan Marvin Garbuszus

16. April 2013

Literatur

- 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

Einstieg

Einstieg Inhalt

Syntax

Wir schreiben Syntax.

- Reproduzierbar!
- Weniger ...
  - klicken
  - wiederholen
  - fehleranfällig

Syntaxdateien

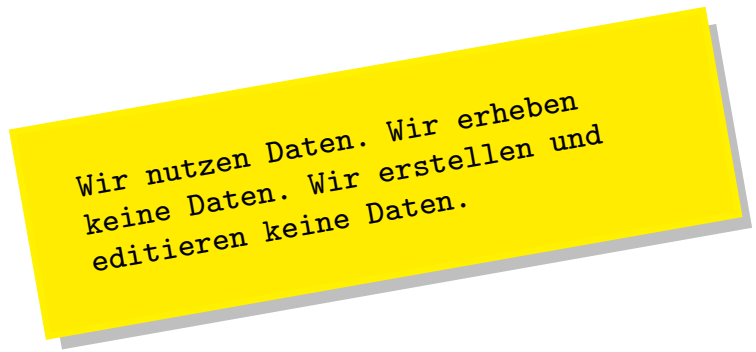
- Sytax in Stata in do-Dateien.
- Aufruf des Editors

```
doedit
```

- Dateiendung von do-Dateien ist .do.
- do-Dateien sind mit jedem Texteditor zu öffnen und zu bearbeiten.

Sekundärdaten

- Primärdaten
- Sekundärdaten



dta-Dateien

- Daten werden als dta-Dateien gespeichert
- Daten enden auf .dta
- Stata Datenobjekte können von Stata und einigen Statistik-Programmen z. B. R (R Core Team, 2013) gelesen werden.

Daten einlesen

- Verzeichnis anzeigen

```
dir
```

- Verzeichnis wechseln

```
cd "D:/Daten/data/"
```

- Daten laden

```
use <dateiname.dta>
```

Vermeiden Sie Ordner- und Dateinamen mit Umlauten oder Leerzeichen. Stata mag das nicht.

do-File I

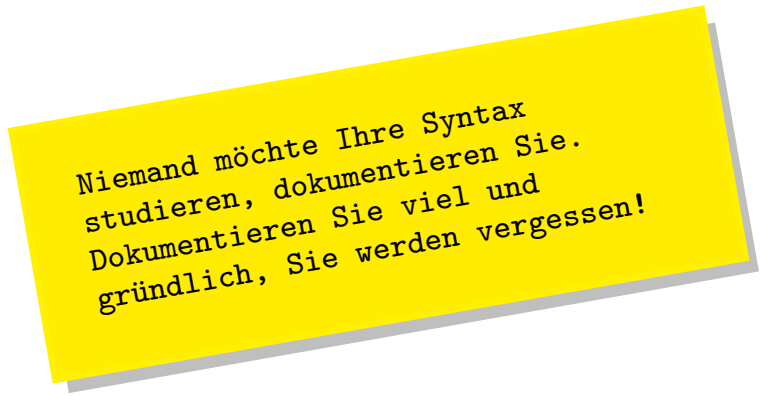
- Dateikopf: Dateiname, der Name des Erstellers, das Erstellungsdatum, eine kurze Information, was die Datei macht und bei längeren Dateien ein kurzes Inhaltsverzeichnis.

```
/*
Name: read-soep.do
Autor: Garbuszus
Datum: 2013-02-28
Inhalt: Einlesen der SOEP-Daten
*/
```

do-File Ia

Dokumentieren Sie!

- Teamarbeit
- Vergesslichkeit



do-File Ib

- /* (Anfang) und */ (Ende) definieren einen Kommentar. Alternativ geht das auch zu Beginn einer Zeile mit * oder nach einem Befehl mit //

```
** Wir schreiben den Dozenten zuliebe Code fuer Version 12
version 12 // Wir haben zwar Stata 13, die aber nicht.
```

- Befehle in Kommentaren werden nicht ausgeführt!

do-File Ic

- Darunter folgt noch

```
** Den Speicher leeren
clear all
** more Funktion abschalten
set more off
```

clear all löscht alles aus dem Speicher, damit ist sichergestellt, dass alles was ausgeführt wird, auf das/die ausgeführte/n do-File/s zurück geht.

do-File II

- Testdaten eingelesen.

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

- Beschreiben

```
describe
```

- Hilfe zum Befehl

```
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

Und wie kommt man an Befehle?

```
findit describe
```

findit kann Hinweise liefern, muss aber nicht. Suchen Sie in jedem Fall nach den englischen Fachbegriffen.

do-File III

Verschiedene Aufgaben, verschiedene do-Files.

- Daten einlesen
- Aufbereitung
- Deskriptive Auswertungen
- usw.

Nicht jedes do-File wird immer gebraucht. Lesen Sie z. B. einmal die Daten ein, bereiten Sie sie einmal auf. Das muss nicht für jede neue Kreuztabelle erfolgen. Zeit ist Geld!

do-File IV

do-Files können ihrerseits auch aus do-Files aufgerufen werden

```
** do-Files aufrufen
do DatenEinlesen.do
do Aufbereitung.do
do DeskriptiveAuswertungen.do
```

Dadurch wird die Syntax schlanker und die do-Files werden garantiert in der richtigen Reihenfolge aufgerufen.

Übersicht

Übersicht der eingelesene Daten

```
describe
codebook
list
browse
```

Nach `help describe` kennt man

```
describe, simple
```

In der Hilfe steht,
ob einem Befehl noch
einzelne Variablen
oder Variablenlisten
übergeben werden können.

Deskriptive Auswertungen

Deskriptive Auswertungen von Variablen[1]

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

Nach `help summarize` kennt man

```
summarize var1, detail
```

[1] `var1` und `var2` sind hier Platzhalter.

Variablen

Variablen Inhalt

Grundlagen

- Generieren
- Umbenennen
- Labeln

Fortgeschritten

- Generieren mit Mathe
- Rekodieren in Variable
- Klassifizieren
- Fehlende Werte
- Variablen Handhabung

Speichern und Laden

- Speichern
- Laden

Generieren

Hin und wieder ist es notwendig Variablen zu erstellen.

```
** Variable klonen
clonevar sex = v298
** Variable generieren
generate geschlecht = .
recode geschlecht . = 1 if sex==1
recode geschlecht . = 2 if sex==2

tab sex
tab geschlecht
```

Umbenennen

Nach einigem hin und her gefällt uns der Variablenname `geschlecht` nicht mehr.

```
drop sex // die Variable werfen wir weg
** geschlecht heißt jetzt sex
rename geschlecht sex
```

Moment generate ... = . ?

Durch den Punkt, wird in Stata ein fehlender Wert gekennzeichnet.

```
recode sex 9 = .
** Verschiedene fehlende Werte
recode sex 8 = .a
recode sex 7 = .b
...
recode sex -3 = .z
```

Der Punkt dient darüberhinaus auch als Zeichen für $+\infty$ und ist die größte Stata bekannte Zahl.

Moment ... if ... == 1 ?

Bedingungen

```
** Haushaltseink ostdeutscher Haushalte mit Bildungsabschluss 1 oder 2
  tab hhinc if east == 1 & ///
    (education == 1 | education == 2)
```

Nach if kommt die Bedingung, verschiedene Bedingungen können mit & und | verknüpft werden. Bei Verkettungen muss trotzdem immer der Variablenname angegeben werden.

Moment tab ?

Kurzschreibweisen

```
help tabulate twoway
```

```
tab ulate varname1 varname2 [if] [ ...
```

Der unterstrichene Teil steht für die minimal notwendige Anzahl an Buchstaben, die gebraucht werden, damit Stata den Befehl eindeutig erkennt.

Label

Label erst definieren, dann zuweisen

```
** Variable labeln
label variable geschlecht "Geschlecht"
** Label für Ausprägungen definieren
label define labgeschlecht 1 "Männlich" 2 "Weiblich"
** Label für Ausprägungen der Variable zuschreiben
label values geschlecht labgeschlecht
```

Jetzt können die Label angezeigt werden

```
tab geschlecht
tab geschlecht, nolabel
```

Generieren II

Variablen können auch durch mathematische Operationen erzeugt werden.

```
** Neu Variable "alter" generieren
gen alter = 2011-gebjahr
sum alter

** Haushaltseinkommen
gen hhinc = inc_female + inc_male
** Haushaltsäquivalenzeinkommen
gen equiv = hhinc * equiscale
```

Generieren III

Eine neue Variable wird aus rekodierten Werten erzeugt, dabei bleibt die Originalvariable unberührt.

```
recode var1 ///
  (-1=.) ///
  (1=1 "Vollzeitbeschäftigung") ///
  (2=2 "Teilzeitbeschäftigung") ///
  (3 5 6 7 8 = 99 "Sonstige Beschäftigung") ///
  (4=3 "geringfügige Beschäftigung") ///
  (9=5 "Keine Beschäftigung"), gen (Beschaeftigung)
```

Variablennamen ohne Sonderzeichen und mit Buchstabe am Anfang.

Moment /// ?

Die drei aufeinanderfolgenden *Slashs* sagen Stata, dass der Befehl sich über die nächste Zeile erstreckt.

```
** alles in einer Zeile
display 1 + 1
** jetzt mit (unnötigem) Zeilenumbruch
display 1 + ///
  1
```

So können zwei und mehr Zeilen verbunden werden. Dadurch wird die Syntax lesbarer. Alternativ: help delimit

Rekodieren

Manchmal bietet es sich an metrische Variablen zu klassifizieren:

```
clonevar alter_kl = alter
recode alter_kl ///
  (18/20=1) (21/30=2) ///
  (31/40=3) (41/50=4) ///
  (51/60=5) (61/110=6)
tab alter_kl
```

18/20 meint hier von 18 bis 20.
21/30 von 21 bis 30. Achten Sie
auf die Klassenbreite!

Daten aufbereiten

Ihr Datensatz erhält, für das Einkommen Angaben von -3, -2 und -1. Dem Codebuch entnehmen Sie, dass diese für „keine Angabe“, „Antwort verweigert“ und „weiß nicht“ stehen. Wenn Sie sich den Mittelwert über das Einkommen angeben lassen, wird dieser – durch die Angaben die kleiner 0 sind – verzehrt.

```
mean hhinc
```

Deshalb kodieren Sie die fehlenden Werte als Missings.

```
mvdecode hhinc, mv(-3=.c\-2=.b\-1=.a)
mean hhinc
```

Fehler

Sie haben Variablen erstellt und kodiert. Sie möchten die soeben erstelle Variable mit den klassifizieren Altern noch einmal anpassen. Was ist zu tun?

- Glücklicherweise haben Sie die Originalvariablen nicht angerührt. Sie können nun einfach die Syntax nochmals von Anfang an durchlaufen lassen.
- Sie kennen die Stelle mit dem Fehler und wollen die falsch erstellte Variable löschen

```
drop alter_kl
```

Daten sichern

Speichern Sie ihr Do-File in einen Ordner *do-files* und ihre Daten in einen Ordner *data*.

```
save "D:/Daten/data/testdaten_rekodiert.dta"
```

Zum erneuten Speichern muss ein anderer Dateiname angegeben werden oder die Option `replace` genutzt werden.

```
save "D:/Daten/data/testdaten_rekodiert.dta", replace
```

Einmal ersetzte Datensätze sind unwiderruflich überschrieben. Überschreiben Sie niemals die Originaldaten!

Daten wieder einlesen

Mit der Option `clear` wird der aktuelle Datensatz ersetzt.

```
use "D:/Daten/data/testdaten.dta", clear
```

Mögliche Veränderungen an vorhandenen eingelesenen Datensätzen werden ignoriert. Sie zwingen Stata hier ein Verhalten auf, vor dem Stata Sie sonst normal warnt.

Grafiken

Grafiken Inhalt

Grafiken

Folgend ein Überblick über verschiedene ausgewählte Grafiktypen. Für Details wie Achsenbeschriftungen, Grafik Titel, mehrere Grafiken und Kombinationen von Grafiken siehe Kohler und Kreuter (2012, S. Kap. 6) und ausführlich und umfassend bebildert Mitchell (2012). Datenbasis der Abbildungen ist jeweils der Allbus Compact 2010.

Vorab ...

Achtung!

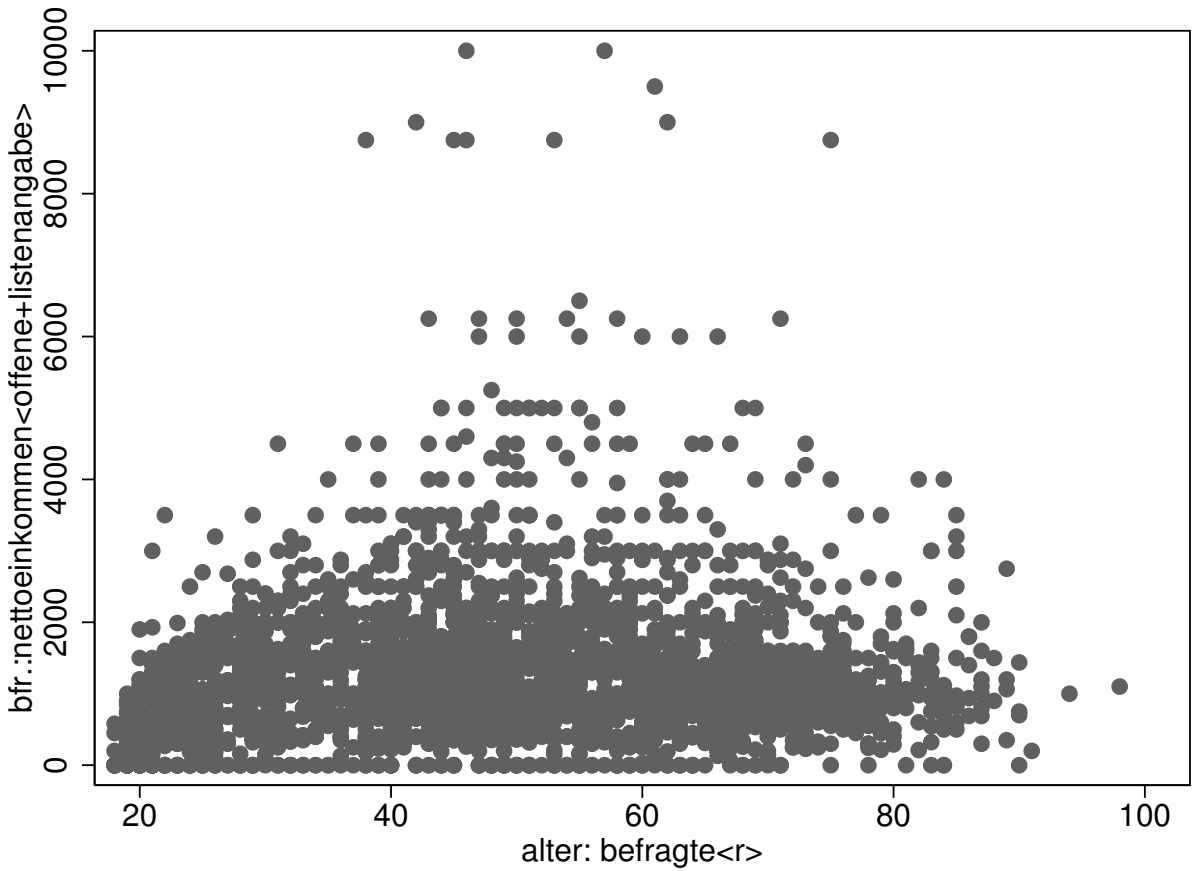
Machen Sie keine Kreis-/ Torten-/ Kuchen-/ Pizza- oder wie sie sonst heißen mögen Diagramme. Machen Sie es einfach nicht!

Außerdem

```
** Sonst wird alles bunt
set scheme s1mono
```

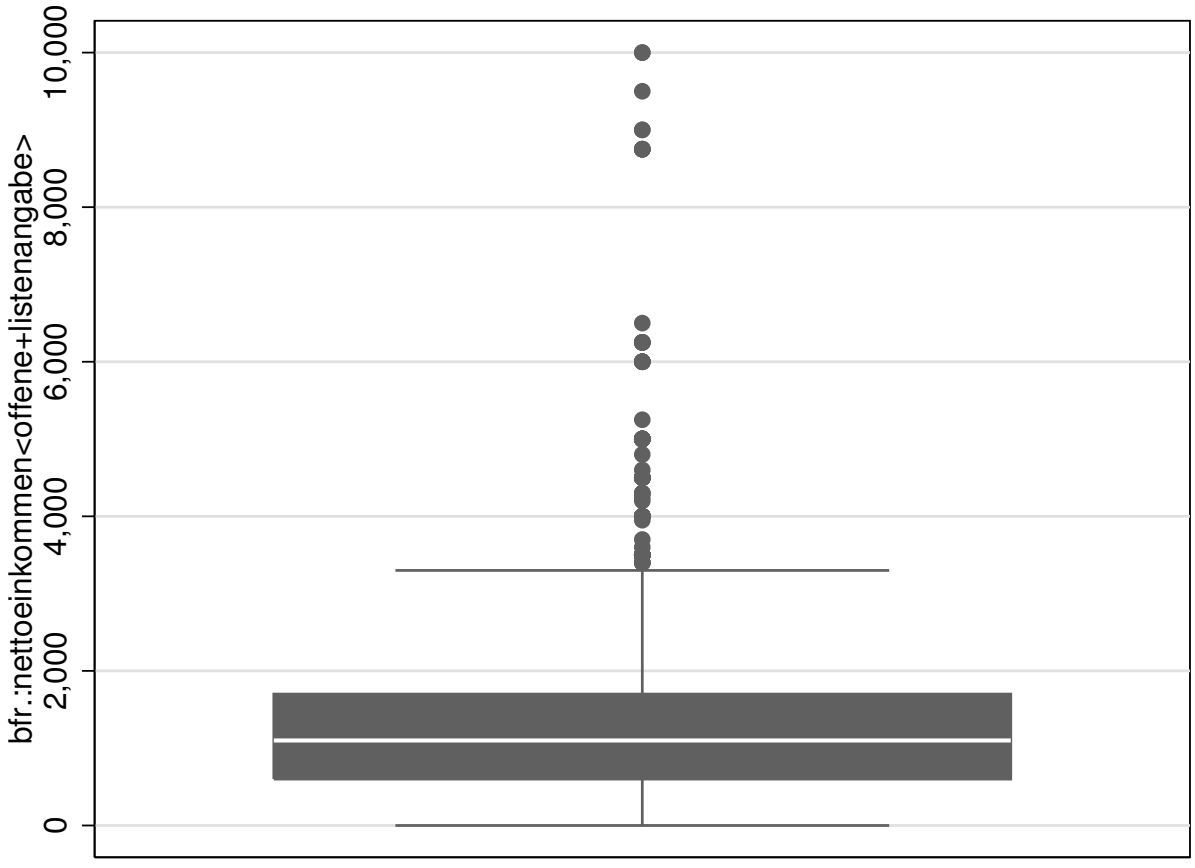
Scatterplots

```
scatter hhinc age
```



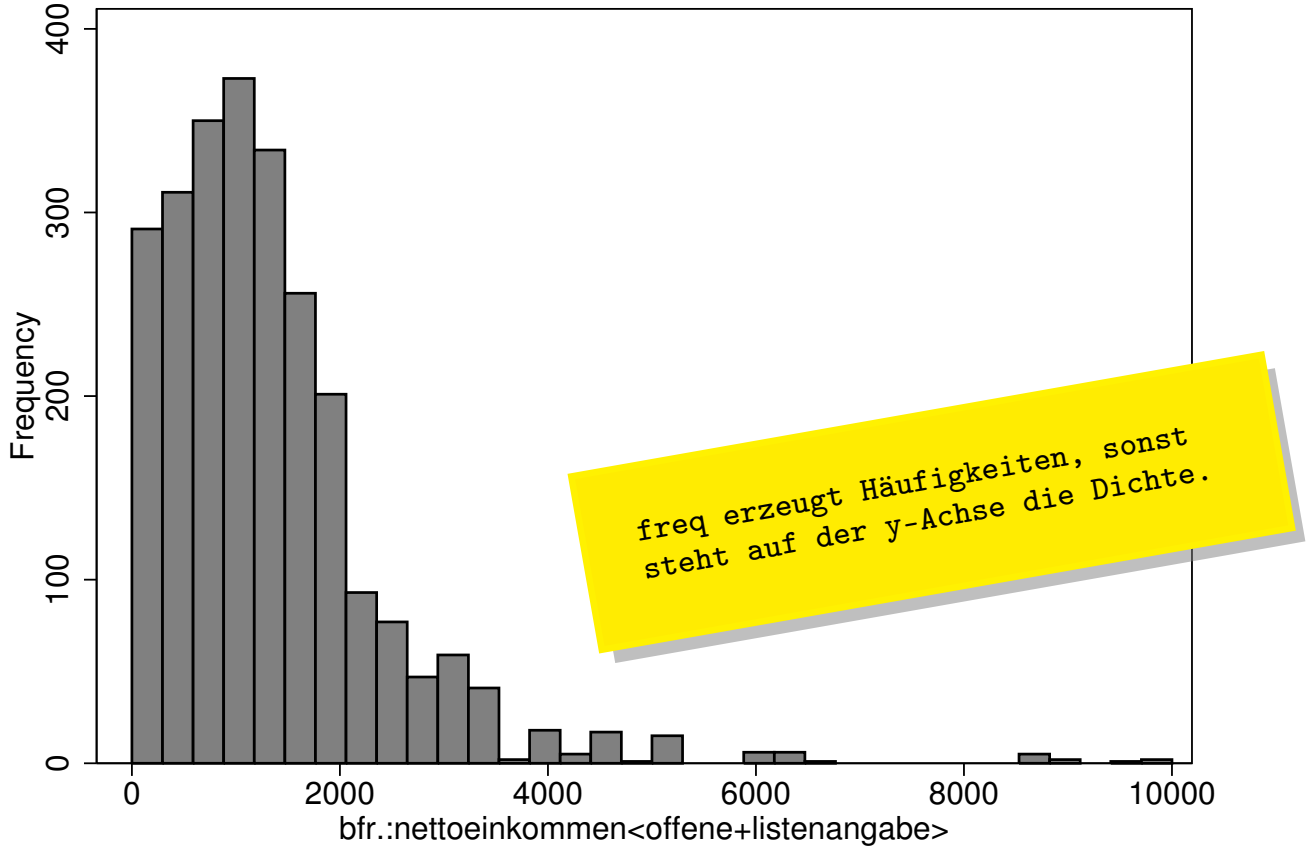
Boxplots

```
graph box hhinc
```



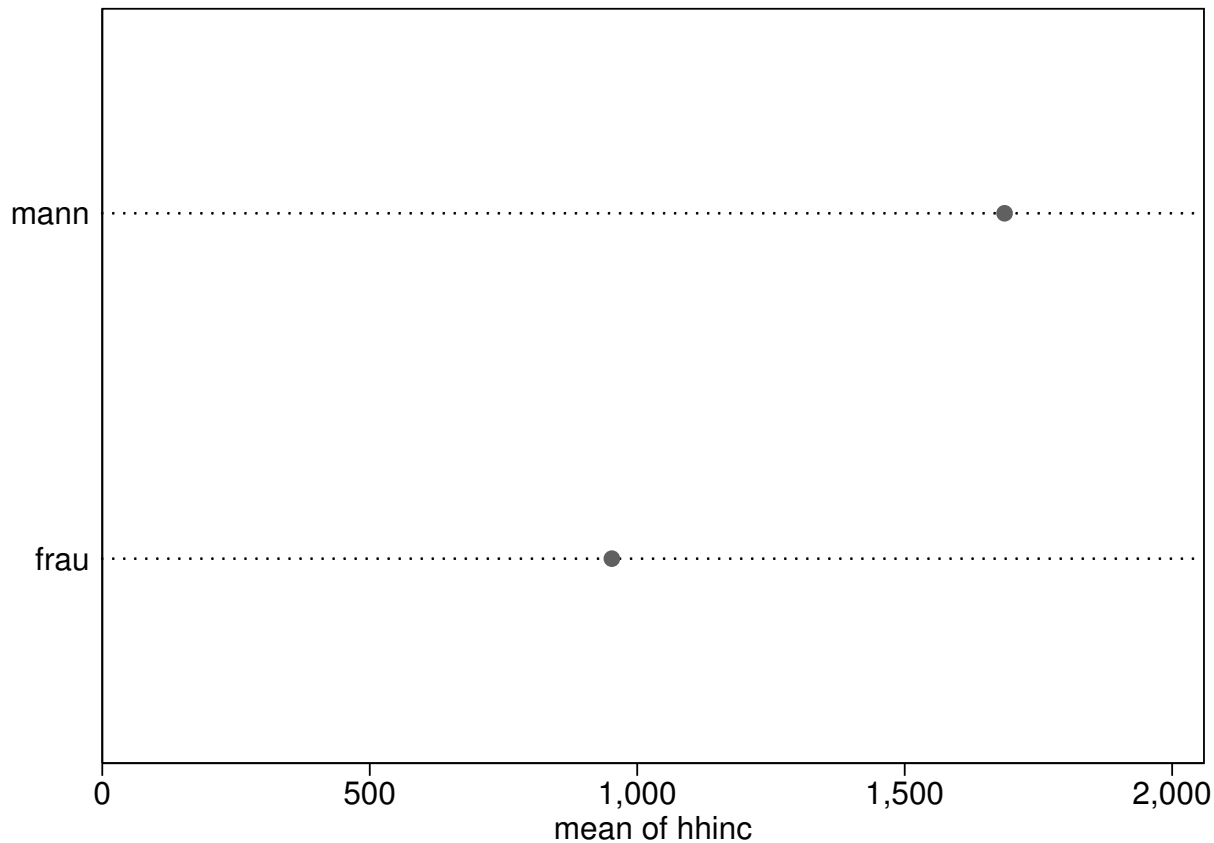
Histogramme

```
hist hhinc, freq
```



Dot-Charts

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



Export von Grafiken

```
graph export "${OUTPUT}\graph1.pdf", replace
```

- Zum Export von Grafiken stehen mehrere Formate zur Verfügung
- .ps .pes .wmf .emf .pdf .png .tif
- Zur Weiterverarbeitung in MS-Office Programmen empfiehlt sich .png
- Zur Weiterverarbeitung in TeX empfiehlt sich .pdf oder .eps

Fortgeschrittenes Datenhandling

Fortgeschrittenes Datenhandling Inhalt

by I

Einzelne Ausprägungen ansprechen

```
summarize hhinc if edu == 1
summarize hhinc if edu == 2
```

⋮

```
summarize hhinc if edu == 7
```

Alle gleichzeitig ansprechen

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

by funktioniert nicht mit jedem Befehl, im Zweifel in die Hilfe gucken. Für by müssen die Fälle sortiert werden. Alternativ: bysort

by II

Es können auch mehrere Variablen genommen werden

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

in

Wenn nur ausgewählte Fälle interessieren

```
** Nur die erste Beobachtung
list sex in 1
** Beobachtungen 1 bis 10
list sex in 1/10
```

Im Regelfall interessieren immer alle Fälle, wenn nicht, dann ist irgendetwas seltsames los.

Schleifen I

Schleifen in Stata. Beispiel aus Kohler und Kreuter (2012, S. 69f.)

```
** Variablen r1 bis r10 erstellen
foreach var of newlist r1-r10 {
  gen `var' = runiform()
}

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

Schleifen II

```
** Mehrere Ausdrücke in einer Schleifen
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

- Auswertungen werden in log-Files gespeichert.
- In die log-Files werden die Befehle und der Output geschrieben.
- Das loggen erfolgt nach

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

- Stata Logfiles können mit jedem Texteditor geöffnet werden.

LogfilesII

- log-File einstellen. Dieses sollte aus Gründen der Übersicht heißen, wie das do-File, welches das log-File erstellt.

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

- Am Dateiende das Log-File schließen.

```
** Befehle nach log close werden nicht mehr geloggt
log close
```

- Bei Fehlern im do-File kann es nötig sein, dass log-File manuell zu schließen

```
log close
```

Macros

Hin und wieder empfiehlt es sich, sogenannte Macros zu verwenden. Ein solches Macro kann z. B. global definiert werden.

```
** global Macroname Pfadname
global DATA "D:/Daten/data/original/"
global OUT "D:/Daten/data/bearbeitet/"
```

Die Macros enthalten jetzt die Pfadangabe

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

Macros II

Diese können nun aus dem do-File aufgerufen werden.

```
use "${DATA}testdaten.dta", clear
save "${OUT}testdaten_rekodiert.dta", replace
```

Dadurch wird die Syntax wieder lesbarer.

Sie sparen sich Tippzeit. Lange Pfade müssen nur einmalig eingegeben werden, dadurch sinkt die Fehleranfälligkeit. Ein Macro was einmal richtig ist, ist das ganze Dokument über richtig. Macros übernehmen aber nicht das Denken.

using

Stata kann nur eine begrenzte Anzahl Variablen handeln, daher sollte man diese ein wenig im Blick behalten.[2] Deshalb nicht immer alle Variablen einlesen

```
** Sehr schmalen ppfad einlesen
** behält nur hhnr persnr sex gebjahr
use hhnr persnr sex gebjahr using "${DATA}ppfad.dta"
** Anschließend Fälle sortieren
sort persnr gebjahr
```

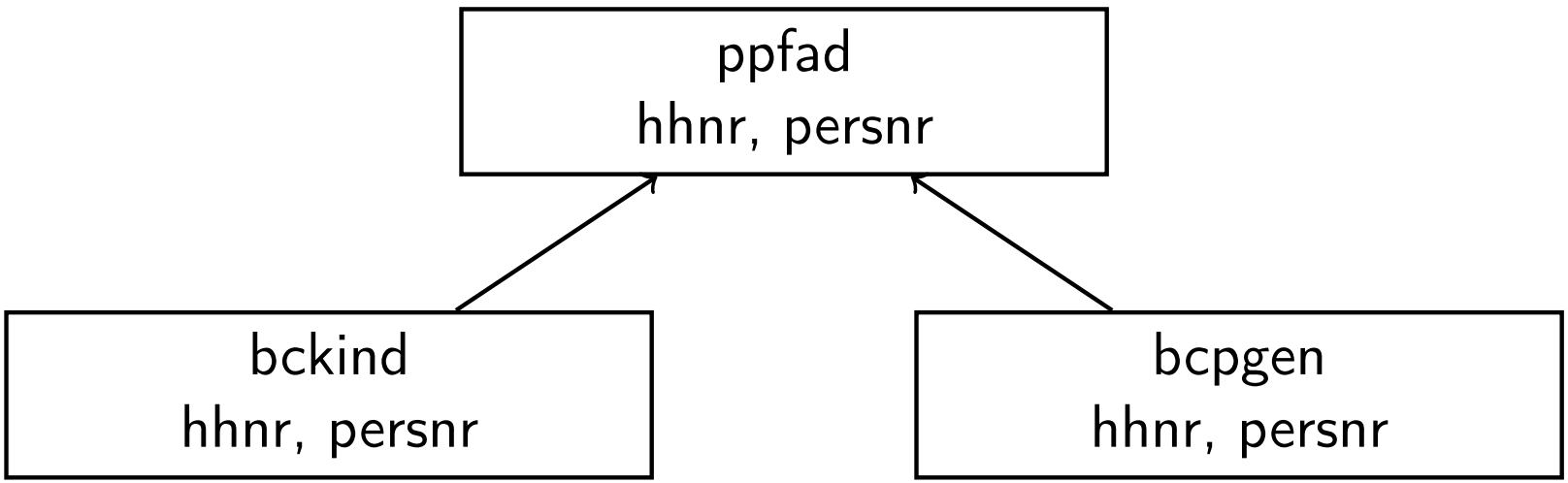
[2] Je nach Version hat Stata ein unterschiedliches Limit an Variablen: IC (die Version aus dem CIP-Raum) 2,047, ab SE 32,767. Mehr Variablen lässt sich Stata aber auch gut bezahlen. Preise 2013: $189 und $395 für Studenten. Nicht akademische Einzelplatzlizenzen $1545 und $2090.

merge

Im hier verwendeten Datensatz, dem Sozio-oekonomischen Panel (SOEP) (s. Wagner et al., 2007), sind die Datensätze eines jeden Jahres in verschiedene Datenfiles aufgeteilt. Das hat zum Teil historische, möglicherweise auch praktische Gründe.
Ausgehend von einem gemeinsamen Datenfile, können die anderen Datenfiles hinzugespielt werden. Dieses hinzuspielen wird *mergen* genannt.

merge II

Wir müssen also unter Umständen aus anderen Datenfiles zum Beispiel aus **bckind** und **bcpgen** Informationen an unseren Datensatz ppfad anspielen. Damit den richtigen Personen die richtigen Informationen zugespielt werden, werden die Haushaltsnummer (hhnr) und die Personennummer (persnr) als Referenz genommen.



merge III

Der `merge`-Befehl wird aufgerufen.

```
** merge ppfad mit 2011 bbp
merge 1:1 persnr using "${DATA}bbp.dta"
```

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

merge IV

Es gibt nun verschiedene Möglichkeiten. Im Folgenden werden alle _merge==2 Fälle gelöscht. Beobachtungen also, die nicht in **ppfad** vorkommen, wohl aber im hinzugespielten Datenfile. _merge==1 würde die Fälle löschen, bei denen ein Eintrag aus **ppfad** nicht im hinzugespielten Datenfile ist, _merge==3, wenn Beobachtungen übereinstimmen.

```
** Drop wenn using nicht in master ist
drop if _merge==2
** _merge droppen
drop _merge
```

Wann welcher merge gewählt wird, hängt nicht unerheblich von der dahinterliegenden Fragestellung ab.

merge V

Beispiel: merge mit der generierten Variable des höchsten Bildungsabschlusses aus **bbpgen**.

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

In bbpgen sind 59 Variablen, wir brauchen aber nur den Schulabschluss.

Operatoren

Stata listet folgende Operatoren, die im Regelfall auch mit Variablen funktionieren. Bei `var1^2` wird jede Beobachtung hoch zwei genommen.

```
== // gleich
!= // ungleich alternativ ~=
& // und
| // oder
! // nicht
< // kleiner
> // größer
<= // kleiner gleich
>= // größer gleich
+ // plus
- // minus
/ // geteilt
* // mal
^ // hoch
```

Funktionen

Darüberhinaus verfügt Stata über eine Reihe von eingebauten Funktionen (`help functions`). Eine Auswahl:

```
abs() // Absolutwert/Betrag |-2| == 2
max() // Größter Wert
min() // Kleinster Wert
exp() // e-Funktion
ln() // Logarithmus
round() // Runden
sin() // Sinus
sqrt() // Wurzel
runiform() // Zufallszahlen
```

Statistik

Statistik Inhalt

Maßzahlen

Tabellen

- Kreuztabellen
- Chi, V und Phi

Inferenz

Maßzahlen I

Arithmetisches Mittel und ein paar Maßzahlen

```
mean sex
summarize sex
summarize sex, detail
```

mean gibt den Standardfehler aus: $s/\sqrt{n}$. summarize gibt die Standardabweichung aus: $\sqrt{\sum(x-\bar{x})^2)/n}$.

Maßzahlen II

Minimum, Maximum, arithmetisches Mittel, Median, Anzahl und Quartile.

```
tabstat age, statistic(min max mean median p50)
tabstat age, statistic(min max range mean count q) by(sex)
```

```
range = max - min
```

Maßzahlen III

Standardabweichung, Standardfehler, Varianz und Interquartilsabstand.

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

Schiefe und Wölbung

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

```
iqr = q3 - q1
```

Tabellen

Wir hatten bereits `summarize` und `tab`, jetzt kombinieren wir

```
tab sex, summarize(age)
```

Kreuztabellen

Kreuztabellen erzeugen mit

```
tab sex east
```

Weitere Tabellen

```
** Tabellen von jeder der drei Variablen
tab1 sex age edu
** Kreuztabellen ab bc ac
tab2 sex age edu
```

Kreuztabellen II

Die Statistik I Vorlesung rekapitulieren, wir machen ein paar Tests [3]

```
** chi-quadrat
tab sex east, chi
** cramers v
tab sex east, V
** wer per Hand nachrechnen will
tab sex east, exp col row
** phi manuell ausrechnen
di (934*426-441*1026) / (1375*1452*1960*867)^(1/2)
di 2827 *(-.02963311)^2 // == chi^2
```

[3] Oder nachschlagen z. B. bei Kühnel und Krebs (2010) oder Agresti und Finlay (2009).

Numlabel

Falls man die numerischen Werte auch im Label haben möchte

```
** numerische Label an für alle Variablen
numlabel _all, add
** numerische Label aus
numlabel _all, remove
```

T-Test

Teststatistik

```
** Vergleich Männer Frauen in West Ost
tab sex, sum(east)
ttest sex, by(east)
```

Und noch viel mehr ...

Regression

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

Fortsetzung folgt.

Index
und Literatur

Index

Literatur 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.

Literatur 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).