

Erste Schritte mit Stata

ein permanent provisorisches Handbuch für EinsteigerInnen

Timotheos Frey*
<mailto:frey@ipz.uzh.ch>

Version vom 29. September 2008
(erste Version: 18.5.2004)

Zusammenfassung

Stata ist ein mächtiges Statistikprogramm, das sich sowohl für die Datenanalyse wie auch für die 2D-Datenvisualisierung eignet. Mit Hilfe dieses Handbüchleins können Stata-Neulinge rasch und ohne grosse Vorkenntnisse¹ erste Arbeitsschritte mit Stata auszuführen. Dieses Handbüchlein ist *permanent provisorisch*, was bedeutet, dass es keinen Anspruch auf Vollständigkeit erhebt, jedoch ständig erweitert wird und in der vorliegenden Form die Stata-Versionen 8 bis 10 abdeckt.

Kommentare und Ergänzungen, insbesondere in Form bereits beschriebener Befehle und Syntaxe, sind jederzeit willkommen. Ich werde diese gerne berücksichtigen, vorausgesetzt, sie entsprechen den Erwartungen des Zielpublikums *Einsteigerinnen und Einsteiger*. Die aktuellste Version ist auf meinem Blog unter nachstehendem URL zu finden:

<http://timfrey.wordpress.com/teaching-stuff/statamanual/>

Weiterführende Literaturangaben und Links zu Hilfeseiten sind sich ganz am Schluss dieses Dokuments zu finden.

*Ich danke an dieser Stelle herzlich Romain Lachat vom Institut für Politikwissenschaft der Universität Zürich für seine Inputs, Christian Brzinsky vom Wissenschaftszentrum Berlin für Sozialforschung für seine Folien, welche er netterweise aufs Netz gestellt hat sowie der Statistics and Empirical Economics Research Group der Uni Zürich für ihre Stata-Hilfsseite auf dem Web, die mir ebenfalls eine grosse Hilfe bei der Erstellung dieses Handbüchleins waren.

¹Eine Ahnung von Statistik und in der Bedienung von Computern wird vorausgesetzt.

Inhaltsverzeichnis

1	Einleitende Bemerkungen	5
1.1	Stata-Versionen	5
1.2	Die Benutzeroberfläche von Stata	5
1.3	Allgemeine Informationen	8
1.4	Synopsis	9
1.5	Bedingungen für Befehle	9
1.6	Reproduzierbarkeit von Analysen: .do-files	9
1.7	Übersicht in grossen .do-files und bei langen Befehlen	10
1.8	Lückenlose Dokumentation der Arbeit	12
1.9	Abbruch eines Befehls	12
2	Einen Datensatz erstellen	12
2.1	Dateneingabe per Editor	12
2.2	Dateneingabe per Syntaxdatei	13
2.3	Einlesen bestehender Datenbanken	13
3	Daten aufbereiten und modifizieren	14
3.1	Der erste Überblick	14
3.2	Anzeigen einer Observation	14
3.3	Nicht-aggregierte Anzeige aller Werte einer Variable	15
3.4	Observationen zählen	15
3.5	Weitere Möglichkeiten zur Fallselektion	16
3.6	Einfache Tabellen (Häufigkeiten)	16
3.7	Erstellen einer neuen Variable	17
3.8	Änderung eines Variablennamens	17
3.9	Variablenlabels	17
3.10	Wertelabels	18
3.11	Verkürzung einer Stringvariable	18
3.12	Rekodierung	19
3.13	Auswahlprobleme mit gestapelten Daten	19
3.14	Zusammenfügen von mehreren Variablen in einer Variable	21
3.15	Generierung einer Datumsvariable aus den drei numerischen Variablen Tag, Monat, Jahr	21
3.16	Mittelwerte in separater Datenbasis sichern	22
3.17	Mittelwerte über verschiedene Variablen	22
3.18	Löschen von Fällen und Variablen	23
3.19	Temporäre Transformationen	23
4	Erste Analysen mit Stata	24
4.1	Bivariate Verteilungen	24
4.2	Drittvariablenkontrolle	24
4.3	Regression	24

5	Schlusswort	25
	Literaturverzeichnis	26

Abbildungsverzeichnis

1	Die Stata-Benutzeroberfläche	6
2	Der Stata-Grafikeditor	7

Vorwort

Es gibt viele Wege, ein Statistikprogramm kennen zu lernen: Manche besuchen Kurse, andere lesen Bücher, wiederum andere lernen den Umgang mit neuen Programmen über den Kontakt mit Kolleginnen und Kollegen.

Ich persönlich bevorzuge die Variante „learning by doing mit Hilfsdokumenten“, denn sie erscheint mir als die effizienteste Methode. Der Besuch von Kursen ist sicher hilfreich, Kurse sind aber oft teuer und berücksichtigen die individuellen Vorkenntnisse und Bedürfnisse nur schwach. Bücher sind dick. Sie sind ein gutes Hilfsmittel für fortgeschrittene Anwender, überfordern aber Einsteigerinnen und Einsteiger in vielen Fällen durch die Fülle der in ihnen enthaltenen Informationen. Kolleginnen und Kollegen sind nett, aber nur solange man nicht mehr als drei Fragen pro Tag stellt. Mit „learning by doing“ dauert es vielleicht etwas länger, bis man einen umfassenden Überblick über die Kapazitäten eines Programms erhält, doch erlaubt diese Variante, rasch und ohne grosse Umschweife mit der Arbeit zu beginnen und spezifisches Wissen gezielt in den Bereichen anzueignen, wo es auch benötigt wird.

Stata ist ausgezeichnet dokumentiert, doch will der Umgang mit der Dokumentation und den Hilfs- und Suchfunktionen erlernt sein. Und dafür gibt es dieses Handbüchlein. Es soll Einsteigerinnen und Einsteiger bis an den Punkt begleiten, wo sie mit der Dokumentation von Stata klarkommen.

Aus diesem Grund ist die allgemeine Einleitung so knapp wie möglich gehalten, dafür sind die Dinge ausführlich beschrieben, die man gleich zu Beginn der Arbeit braucht, so zum Beispiel die Dateneingabe und -modifikation, deskriptive Statistiken oder die Fallselektion. Dieses Papier ersetzt also nicht die Stata-Hilfsfunktion, Lehrbücher oder die Stata-Handbücher. Es erlaubt aber, rasch und ohne grosse Vorkenntnisse erste Arbeitsschritte mit Stata auszuführen.

1 Einleitende Bemerkungen

1.1 Stata-Versionen

Stata ist ein Programm, das bereits auf eine lange Geschichte zurückblicken kann. Obwohl Stata auf den ersten Blick immer noch gleich wie in den 1980er Jahren aussieht, hat sich im Verlauf der Jahre „unter der Haube“ mächtig viel getan. Seit der Version 8 verfügt Stata über eine graphische Benutzeroberfläche, auf die im nächsten Abschnitt eingegangen wird. Mit Stata 9 wurde Stata um Mata, eine Matrix-Programmiersprache, und um zahlreiche multivariate Methoden, zum Beispiel multidimensionale Skalierungen oder Korrespondenzanalysen, ergänzt. Die heute aktuelle, 10. Version brachte neben vielen weiteren Neuerungen einen dynamischen Grafikeditor mit. Für Anfängerinnen und Anfänger sind nur zwei Punkte wichtig zu wissen: Erstens speichert Stata 10 Daten in einem anderen, xml-ähnlichen Datenformat. Will man also Datenbanken mit älteren Stata-Versionen austauschen, müssen diese in Version 9 gesichert werden. Zweitens gibt es überholte Befehle, was dazu führen kann, dass eine neue Stata-Version alte Syntaxdateien nicht mehr ausführen mag. Man kann jedoch mit der Eingabe des Befehls `version` Stata anweisen, Syntaxdateien entsprechend der angegebenen Version zu lesen – und dann funktioniert wieder alles wie gewohnt.

Von allen Stata-Versionen gibt es verschiedene Varianten, die sich in erster Linie durch die maximale Datenkapazität unterscheiden. Für sozial- und wirtschaftswissenschaftliche Anwendungen reichen „Intercooled Stata“ oder „Stata/IC“ in den meisten Fällen bei Weitem aus.²

1.2 Die Benutzeroberfläche von Stata

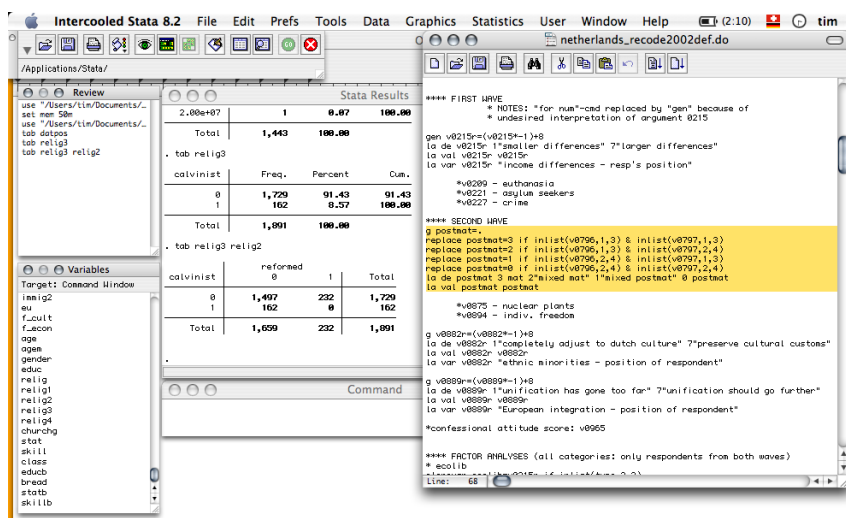
Egal ob es um Programme, Restaurants, Bücher oder auch Menschen geht, der erste Kontakt ist meist visuell. Stata kommt mit einer sehr einfachen und aufgeräumten Oberfläche daher. Wenig Schnickschnack, aber auch wenig Knöpfe zum Drücken, was zu Beginn vielleicht verunsichern mag.

Bis zur Version 7 konnte Stata praktisch nur über die Kommandozeile oder aber über Syntax-Dateien gesteuert zu werden. Diese Vergangenheit prägt das Erscheinungsbild bis heute, sie soll aber nicht darüber hinwegtäuschen, dass Stata seit der Version 8 auch über eine umfassende grafische Oberfläche (GUI³) verfügt. Diese wird vor allem über Pulldown-Menüs aus der Befehlsleiste (die Leiste am oberen Bildschirmrand; Bei Windows der obere Fensterrand) bedient. Auch wenn Befehle über das GUI eingegeben werden, erscheinen sie im Resultatsfenster. Das ist zum Beispiel bei der Erstellung von Syntax-Dateien extrem hilfreich, denn die korrekte Syntax

²Überblick zu den aktuellen Varianten: <http://www.stata.com/products/whichstata.html>

³GUI: *Graphic User Interface*

Abbildung 1: Die Stata-Benutzeroberfläche unter Mac OS X (unter anderen Betriebssystemen ist sie praktisch identisch). Links oben [Review]: Die letzten Befehle; links unten [Variables]: Die Liste aller Variablen der Datenbank; Mitte oben [Stata Results]: Das Resultatsfenster, welches alle Resultate, aber auch die Befehle anzeigt; Mitte unten [Command]: Hier können Befehle direkt eingegeben werden; Rechts im Vordergrund: Syntax-Editor mit offener Syntax-Datei.



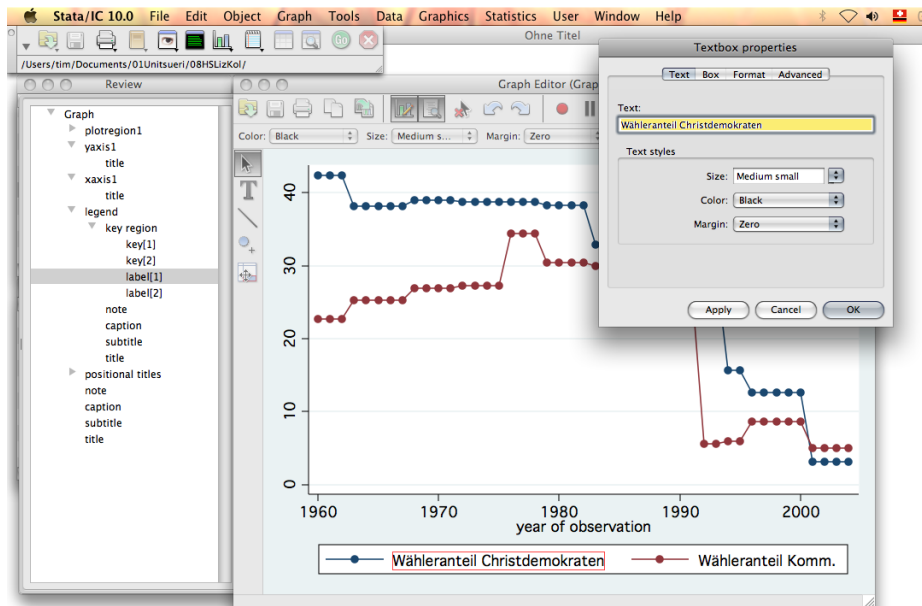
wird immer gleich mitgeliefert. Die wichtigsten Funktionen können direkt über die Knöpfe oben links oder am oberen Rand des Syntax-Editors aufgerufen werden. Die Funktion der einzelnen Knöpfe wird angezeigt, wenn man den Cursor einen kurzen Moment auf dem Knopf lässt, ohne diesen zu drücken.

Mit der Version 10 erhielt Stata einen dynamischen Grafikeditor, siehe dazu die Abbildung 2 auf Seite 7. Durch simples Doppelklicken können die einzelnen graphischen Elemente wie Linienstärken und Boxen, aber auch Titelinhalt Achsenbeschriftungen direkt manuell bearbeitet werden. So ist es auch für ungeübte Benutzerinnen und Benutzer sehr rasch möglich, ansprechende Diagramme zu erstellen.

Die Stata-Hilfsfunktionen

Viele Grundfunktionen lassen sich heute also auch über ein GUI steuern. Dies gilt natürlich auch für die umfassenden Hilfs- und Suchfunktionen, welche direkt über das Menü ‚Help‘ in der Befehlsleiste aufgerufen werden können.

Abbildung 2: Der dynamische Stata-Grafikeditor ab Version 10 unter Mac OS X: Praktisch alle graphischen Elemente lassen sich auch nach der Erstellung eines Diagramms noch manuell ändern.



Dieses Handbüchlein beschränkt sich aber ab hier auf über die Kommandozeile einzugebenden oder in einem Syntax-Dokument verwendbaren Befehle, da ich davon ausgehe, dass jemand, der mit Stata arbeitet, grundsätzlich mit komplexen Daten arbeitet. Und in solchen Fällen ist die Verwendung von Syntax-Dokumenten (bei Stata sogenannte .do-files) unumgänglich.

Genaue Angaben über den Funktionsumfang einzelner Befehle erhält man durch die Eingabe einer Hilfe-Abfrage mit dem Befehl `help`. Beispiel: Hilfe und Informationen zu Rekodierungen erhält man durch den Befehl `help recode`. Das zweite Kommando der interaktiven Stata-Hilfe lautet `search`, welches eine Stichwortsuche erlaubt und im Gegensatz zum Befehl `help` keine Kenntnis des genauen Befehls voraussetzt. Eine Übersicht zu den verschiedenen Hilfsdokumenten erhält man durch die Eingabe des Befehls `search help`.

1.3 Allgemeine Informationen

Es gibt ein paar Dinge, die man unbedingt wissen sollte, bevor man sich in die Arbeit mit Stata stürzt:

- Alle hier erwähnten Befehle funktionieren in allen Stata-Versionen ab Stata 7, lassen sich direkt in der Kommandozeile eingeben, über ein .do-file ausführen und verfügen über einen Eintrag im Manual, welches, wie bereits im letzten Abschnitt erwähnt, mit dem Befehl `help` aufgerufen werden kann.
- Stata ist *case-sensitive*. Das bedeutet beispielsweise, dass Stata immer und überall zwischen Gross- und Kleinschreibung unterscheidet. Eine Variable `Relig12` ist also nicht die Gleiche wie die Variable `relig12`.
- *Speicherzuteilung*: Standardmässig startet Intercooled Stata mit einer Arbeitsspeicherbelegung von 1 Megabyte. Dies ist relativ knapp bemessen und reicht schon bei einer mittleren Datenbasis nicht aus, um die Daten in den Arbeitsspeicher zu laden. Mit dem Befehl `set memory` kann Stata mehr Arbeitsspeicher zugeordnet werden. Beispiel: Der Befehl `set mem 50m` teilt Stata 50 Megabyte Arbeitsspeicher für die Daten zu.
- *Struktur der Daten*: Im Prinzip ist eine Stata-Datenbank (oder -basis) nichts anderes als eine grosse Tabelle. Linien stehen für die einzelnen Observationen, Spalten für die Variablen. Mit dem Befehl `browse` kann man sich die Daten anzeigen lassen, mit dem Befehl `edit` diese auch direkt bearbeiten. Stata verwendet für Datenbanken ein eigenes, proprietäres Datenformat, erkennbar an dem Suffix `.dta`.
- *Abkürzung der Befehle*: Die meisten Stata-Befehle lassen sich auf die minimale Anzahl Stellen abkürzen, die für Stata noch einen eindeutigen Befehl darstellen. Anstelle von `generate` kann man beispielsweise auch nur `gen` schreiben. In den Hilfsdokumenten sind viele zulässige Abkürzungen durch unterstrichene Befehlsfragmente angegeben.
- *Programmversionen und Befehle*: Jede Stata-Version kennt im Vergleich zu früheren Versionen neue Befehle. Das Gegenteil ist aber auch möglich: Einige Befehle sind in neuen Versionen nicht mehr gültig oder bedeuten etwas anderes. Um solche Probleme zu vermeiden, kann man Stata mit Hilfe des Befehls `version` „in vergangene Zeiten zurückkehren“. Egal ob man nun unter Stata 8 oder Stata 9 arbeitet: Eine Syntax, die zum Beispiel 1999 auf Stata 7 geschrieben wurde, wird korrekt interpretiert, wenn zuerst der Befehl `version 7` eingegeben wird.
- *Portabilität*: Stata ist für die meisten Betriebssysteme, also für verschiedene Linux-Distributionen genau so wie für Windows oder Mac

verfügbar; die Dokumente (Syntaxen, Datenbanken) können ohne Einschränkungen untereinander ausgetauscht werden. Ebenso lassen sich alle hier beschriebenen Befehle auf allen Plattformen verwenden.

Die meisten Stata-Befehle gelten plattformübergreifend, doch können Ausnahmen vorkommen. Stata versteht auf UNIX-Plattformen auch viele UNIX-Befehle, obwohl ein spezifischer Stata-Befehl existiert. So löscht der Stata-Befehl `erase` unwiederruflich ein auf der Festplatte gesichertes Dokument. Auf einem UNIX-Rechner kann zum gleichen Zweck aber auch der UNIX-Befehl `rm` eingegeben werden).

1.4 Synopsis

Die Struktur von Stata-Befehlen ist eigentlich immer die gleiche: Zuerst steht der Befehl, anschliessend die betreffenden Variablen⁴ und am Schluss folgen allfällige Optionen. Die Optionen werden durch ein Komma vom Rest des Befehls abgetrennt. Der Befehl wird normalerweise mit einem Zeilenschlag (Return-Taste) abgeschlossen (normalerweise, denn es gibt auch andere Möglichkeiten. Siehe hierzu den Abschnitt 1.7 „Übersicht in grossen do-files und bei langen Befehlen“).

1.5 Bedingungen für Befehle

Viele Befehle lassen sich durch Bedingungen, sogenannte *expressions*, eingrenzen und auf eine bestimmte Auswahl der Fälle beschränkt anwenden. *Expressions* werden durch Befehle wie `if` eingeleitet. Logische Argumente werden mit logischen Operatoren wie zum Beispiel dem Zeichen `==` für ‚gleich‘ eingebracht. Verschiedene Beispiele folgen unten, genauere Informationen gibt es mit dem Befehl `help operator`. Strings (Text) müssen bei Abfragen oder sobald mehr als ein Wort vorkommt, in Anführungs- und Schlusszeichen gesetzt werden. Numerische Werte können direkt und ohne weitere Definition verwendet werden, egal ob es sich um Eingaben oder um einen Teil eines Arguments handelt.

1.6 Reproduzierbarkeit von Analysen: .do-files

.do-files sind einfache Textdateien mit Stata-Befehlen, welche hintereinander ausgeführt werden. Einerseits erleichtern sie die eigene Arbeit, denn sie erlauben jederzeit die Korrektur und Aktualisierung von Rekodierungen, Variablen oder Analysemodellen. Andererseits erlauben sie die intersubjektive Nachvollziehbarkeit: dank .do-files wird anderen Forschenden ermöglicht, die Forschungsergebnisse mit den gleichen Methoden zu bestätigen oder zu kritisieren. Do-files können mit Hilfe von Stata aber auch mit jedem Texteditor

⁴Stata nennt das *varlist* in den Hilfsdokumenten

erstellt werden. Damit do-Files von Stata als solche erkannt werden, müssen sie mit dem Suffix `.do` versehen sein.

1.7 Übersicht in grossen `.do`-files und bei langen Befehlen

Mit etwas Disziplin und den folgenden drei kleinen Massnahmen gerät man auch bei der Arbeit mit umfangreichen do-Files nicht in Gefahr, den Überblick zu verlieren.

1. Befehlsmarkierung: Im Gegensatz zu SPSS, wo Befehle in einer Syntax mit einem Punkt abgeschlossen werden, ist die Eingabe eines bestimmten Zeichens zu diesem Zweck bei Stata nicht nötig. Eine neue Zeile bedeutet ein neuer Befehl. Bei sehr langen Befehlen, beispielsweise bei Wertelabels, kann dies aber zu einer sehr unübersichtlichen Darstellung führen. Mit dem Befehl `#delimit` kann das Steuerzeichen für einen nächsten Befehl geändert werden. Im folgenden Beispiel hebt der Befehl `#delimit` den Zeilenumbruch als Befehlstrenner auf und ersetzt ihn durch das Semikolon. Ab nun behandelt Stata alle Informationen zwischen den Semikolon als ob sie auf einer Linie geschrieben wären. Mit dem Befehl `#delimit cr` wird wieder zum üblichen Modus zurückgekehrt. Nachstehend ein Beispiel für die Verwendung dieses Befehls: Definition von Wertelabels für Akteure. Beachte das Semikolon auf der zweitletzten Zeile, welches den Abschluss des Befehls markiert.

```
#delimit ;
la de akt3
    2460 "Schoch, Hans (FDP)"
    2462 "Roselli, Maria (SP)"
    2463 "Gallade, Chantal (SP)"
    2465 "Heer, Alfred (BDS)"
;
#delimit cr
```

2. Dokumentation der Arbeiten an einem `.do`-file: Damit nie Zweifel über die Version oder über die zuletzt gemachten Änderungen aufkommen, empfiehlt es sich, das `.do`-file genau zu dokumentieren. Am einfachsten ist ein einfaches Log, eine Art Tagebuch, versehen mit Datum, Name und einem kurzen Beschrieb gleich zu Beginn des `.do`-files. Es ist auch möglich, Kommentare direkt im Syntax anzubringen. Damit Stata diese Textfragmente nicht zu interpretieren versucht, müssen sie auskommentiert werden. Lange Rede, kurzer Sinn: ein Stern `*` zu Beginn der Zeile, und Stata kümmert sich nicht mehr um das, was auf der entsprechend markierten Zeile noch folgt.

3. Organisation und funktionale Unterscheidung von .do-files: Gerade bei grösseren Arbeiten ist eine Unterscheidung von *Create-do-files*, mit denen Datensätze erzeugt werden, und *Analyzing-do-files*, mit welchen Datensätze untersucht werden, sinnvoll. Damit der Überblick gewahrt bleibt und die verschiedenen .do-files nicht jedes Mal einzeln von Hand gestartet werden müssen, kann ein Master-do-file angelegt werden, welches die einzelnen .do-files in der richtigen Reihenfolge startet. Ein Beispiel eines kompletten Master-do-files gibt es auf der nächsten Seite. Die Dateipfade sind entsprechend dem Betriebssystem und der Datenstruktur auf dem Rechner zu setzen.

Beispiel eines Master-do-files: Zuerst wird Stata genügend Arbeitsspeicher zugewiesen, anschliessend eine Datenbank geöffnet und dann werden die verschiedenen .do-files aufgerufen.

```
** Master Test-Analyse **
** LOG: created 20.5.2004 - tim

set mem 40m
use "/Users/hobbes/files/test.dta"
do "/Users/hobbes/files/cr1test.do"
do "/Users/hobbes/files/cr2test.do"
do "/Users/hobbes/files/an1test.do"
```

1.8 Lückenlose Dokumentation der Arbeit

Gerade bei einer *explorativen* Arbeitsweise ist es sinnvoll, sämtliche Arbeitsschritte genau dokumentiert zu haben. Mit dem Befehl `log using` weist man Stata an, eine Log-Datei anzulegen, welche sämtliche Inputs und Outputbefehle automatisch notiert. Eine solche Logdatei ist im Prinzip nichts anderes wie eine Blackbox eines Flugzeugs oder eine Überwachungskamera: man kann immer noch Mist bauen, weiss aber nachher, wo der Fehler passiert ist. Zum Befehl `log using` gehört immer die Angabe eines Dateinamens, der komplette Befehl lautet beispielsweise `log using log0504.log`. Mit dem Befehl `print log` wird dann dieses Log gedruckt, der Befehl `view log` zeigt alle Einträge am Bildschirm an. Es ist auch möglich, nur den Befehlsinput aufzunehmen. In diesem Fall verwendet man den Befehl `cmdlog` anstelle des Befehls `log`. Mit dem Befehl `log close` beziehungsweise `cmdlog close` wird die Logdatei wieder geschlossen.

1.9 Abbruch eines Befehls

Durch Drücken der Tasten `ctrl + c` kann jeder Befehl abgebrochen und wieder zur Kommandozeile zurückgekehrt werden.

2 Einen Datensatz erstellen

2.1 Dateneingabe per Editor

Mit dem Befehl `edit` wird der Dateneditor von Stata aufgerufen, der auf den ersten Blick der Oberfläche von Tabellenkalkulationsprogrammen ähnelt. Er erlaubt die direkte Dateneingabe per Tastatur. Im Gegensatz zu Tabellenkalkulationsprogrammen ist die Datenstruktur hier aber nicht frei. Zeilen stehen für Beobachtungen (Fälle), Spalten für Variablen.

2.2 Dateneingabe per Syntaxdatei

Tippfehler passieren rasch, nachvollziehbar sind sie selten. Deshalb ist es besser, Daten nicht von Hand, sondern über eine Syntaxdatei einzugeben. Nur so lassen sich allfällige Fehler entdecken, nachvollziehen und auch zu einem späteren Zeitpunkt noch korrigieren. Für die Dateneingabe steht der Befehl `input` zur Verfügung. Nach dem Befehl `input` setzt man die Variablenamen der neuen Variablen. Anschliessend – und das ist eine der grossen Ausnahmen, was den Befehlsabschluss betrifft – drückt man die Enter-Taste, um die Observationen eingeben zu können. Die Dateieingabe wird mit `end` abgeschlossen. Hier ein Beispiel mit rein numerischen Variablen:

```
input var_a var_b var_c
      12.1  13   14.6
      22   28.8   .
end
```

Fehlende Werte werden mit `.` eingegeben, so wie in diesem Beispiel der Wert der Variable `var_c` der zweiten Observation.

Variablen, die Text enthalten [*Strings*] müssen als solche definiert werden, wie das nachfolgende Beispiel zeigt. `str16` ist in diesem Fall keine Variable, sondern definiert die Variable `name` als String mit maximal 16 Zeichen. Bei der Eingabe der Observationen sind Texteingaben in Anführungszeichen zu setzen.

```
input str16 name alter str16 material
      "calvin"  8  "biomasse"
      "hobbes"  5  "pluesch, gestopft"
      "charlie brown"  8  "biomasse"
end
```

2.3 Einlesen bestehender Datenbanken

Stata unterstützt Import und Export von Daten in ASCII, CVS, FDA (SAS) und XML (ab Stata 9). ASCII- und aus Excel exportierte CSV-Daten liest man mit dem Befehl `insheet` ein. Wie man dabei am Besten vorgeht, steht in der Dokumentation, ganz genau beschrieben, einfach `help insheet` eingeben.

Datenaustausch zwischen SPSS und Stata ist dank dem Programm Stat-Transfer⁵ problemlos möglich.

⁵Siehe <http://www.stattransfer.com/>

3 Daten aufbereiten und modifizieren

3.1 Der erste Überblick

Mit dem Befehl `ds` erhält man eine Liste sämtlicher Variablennamen der Datenbasis, die im Arbeitsspeicher geladen ist. Der Befehl `ds` lässt sich durch verschiedene Optionen einschränken, siehe dazu die Hilfsdokumente. Wünscht man einen ausführlicheren Beschrieb, dann wählt man den Befehl `describe`. Mit seiner Hilfe zeigt Stata eine Liste aller Variablen unter Angabe des Variablentyps, des Formats, allfälliger Wertelisten und des Variablenlabels an. Auch dieser Befehl lässt sich durch verschiedene Optionen einschränken, beispielsweise durch eine nachgestellte Auswahl von Variablen. Dabei können auch durch Wildcards⁶ ergänzte Variablennamen verwendet werden.

Beispiel: `describe *su*`

Darauf wird eine Liste angezeigt, die ungefähr so aussieht:

variable name	storage type	display format	value label	variable label
sub1	str72	%72s		SUB1
sub2	str22	%22s		SUB2
sub3	str30	%30s		SUB3
issue1	int	%8.0g	iss1	Themenbereich
issue2	int	%8.0g	iss2	Thema

Aufgrund der Einschränkung `*su*` werden nur die Variablen angezeigt, in deren Namen irgendwo die Buchstabenkombination `su` vorkommt. `storage type str72` bedeutet, dass es sich bei der Variable `sub1` um eine Stringvariable mit 72 Stellen handelt. Der Eintrag `iss1` zeigt an, dass diese Variable `issue1` mit einer Liste von Wertelabels (siehe dazu 3.10) ausgerüstet ist, die unter dem Namen `iss1` definiert wurde.

Der Befehl `summarize` zeigt die wichtigsten univariaten Statistiken⁷ aller numerischen Variablen an.

3.2 Anzeigen einer Observation

Dies geschieht mit dem Befehl `list`. Diesen Befehl ist eigentlich immer nur mit Einschränkungen zu verwenden, das heisst, man wählt einen oder mehrere Datensätze, die bestimmten Suchkriterien entsprechen.

⁶eine Wildcart ist ein Platzhalter für „hier sollte noch etwas anderes stehen“. Stata verwendet das Zeichen `*` als Wildcart.

⁷Min, Max, Mean, Standardabweichung

Beispiel: `list if issue2==1040`

`if` schränkt die anzuzeigenden Observationen ein. Es werden nur die Observationen angezeigt, die dem folgenden Argument (in diesem Fall der Wert 1040 der Variable `issue2`) entsprechen.

3.3 Nicht-aggregierte Anzeige aller Werte einer Variable

Dies geschieht ebenfalls mit dem Befehl `list`, man ergänzt ihn einfach mit der oder den gewünschten Variablen. Beispiel: `list var1 var2` zeigt die Werte dieser beiden Variablen über sämtliche Observationen an.

3.4 Observationen zählen

Der Befehl `count` zählt die Anzahl Observationen, die eine bestimmte Bedingung erfüllen. Will man einfach die gesamte Anzahl Observationen einer Datenbank zählen, dann gibt man einfach `count` ein.

Beispiel: `count in 12/22`

Die Einschränkung `in 12/22` bedeutet, dass Stata nur die Observationen zwischen der 12. und der 22. Stelle in der Datenbank zählt. In diesem Fall wird folgendes Resultat angezeigt: 11. Etwas schwieriger ist das Zählen von „unterschiedlichen Vorkommen“. Beispiel: In einer Variable sind Werte enthalten, die unterschiedlich oft vorkommen. Um nun herauszufinden, wie viele unterschiedliche Werte es in dieser Variable gibt, kommt man um einen tiefen Griff in die Trickkiste nicht herum⁸: Man muss mit Makros arbeiten. Weiter auf Makros einzugehen würde den Rahmen dieses Handbuchs sprengen, deshalb hier einfach die Syntax dazu:

```
quietly levels var_name, local(irgendein_name)
    local i = 1
    foreach x of local irgendein_name {
        local count='i++'
    }
display " 'count' "
```

Der Ausdruck `var_name` wird einfach durch den Namen der Variable ersetzt, in der die einzelnen Vorkommen gezählt werden sollen. `irgendein_name` kann durch irgendwas ersetzt werden, es sollte einfach zwei Mal das gleiche stehen, sonst weiss das gute Stata nicht, wo es die zwischengespeicherten Informationen herholen soll. Mit dem abschliessenden Befehl `display` wird das Resultat angezeigt. Will man diese Syntax über die Kommandozeile

⁸*thanks to Romain*

eingeben, so muss man nach jeder Zeile die Enter-Taste drücken, ein kleines .do-File zu schreiben ist in diesem Fall sicher die bequemere Lösung.

3.5 Weitere Möglichkeiten zur Fallselektion

`if` lässt sich mit vielen Befehlen kombinieren, so zum Beispiel mit dem oben erwähnten Befehl `list` oder mit den noch folgenden Befehlen `tab`, `gen` oder `replace`. Die entsprechenden Hilfsdokumente geben zu jedem Befehl genaue Auskunft über die Einsetzbarkeit über die verschiedenen `if`. An dieser Stelle das vermutlich wichtigste Beispiel neben der oben erwähnten Anwendung: die Bedingung `inrange`. Mit `inrange` wird eine fortlaufende Wertgruppe durch die Nennung ihres Minimal- und Maximalwertes aufgerufen.

Syntaxbeispiel: `replace varxy=345 if inrange(obs, 1240, 1250)`

Dieser Befehl schreibt den Wert 345 in die Variable `varxy`, sofern sie eine Observationsnummer `obs` zwischen 1240 und 1250 tragen. Zum gleichen Resultat führt auch folgende Schreibweise:

```
replace varxy=345 if obs>=1240 & obs<=1250
```

Mehr Infos zum hier erwähnten Befehl `replace` gibt es im Abschnitt 3.12 *Rekodierung*.

3.6 Einfache Tabellen (Häufigkeiten)

Mit dem Befehl `tabulate` erhält man einen Überblick über den Inhalt einer Variable. `tab` kennt viele Optionen und Erweiterungen, siehe dazu die Hilfe.

Beispiel: `tab zeitung` zeigt folgende Tabelle an:

Zeitung	Freq.	Percent	Cum.
NZZ	6,290	42.01	42.01
Blick	835	5.58	47.58
Tagesanzeiger	7,849	52.42	100.00
Total	14,974	100.00	

Die Variable `zeitung` enthält numerische Werte; angezeigt werden jedoch die Wertelabels (in diesem Fall die Zeitungsnamen) sofern diese definiert sind. Um die numerischen Werte der Variable `zeitung` zu sehen -oder, mit anderen Worten ausgedrückt- um die Anzeige der Wertelabels zu unterdrücken, muss der Befehl `tab` mit der Option `nolabel` ergänzt werden. Der komplette

Syntax würde dann folgendermassen geschrieben:

```
Syntax: tab zeitung, nol
```

Der Befehl `nolabel`, hier auf seine drei ersten Stellen `nol` abgekürzt, ist eine Option des Befehls `tabulate`. Aus diesem Grund folgt er nach dem Komma.

3.7 Erstellen einer neuen Variable

Hierfür gibt es mehrere Möglichkeiten. Natürlich kann man einfach den Data-Editor öffnen (entweder über das Menü oder mittels dem Befehl `edit`) und dort von Hand eine weitere Spalte einfügen. Eleganter geht es aber über den Befehl `generate`. Dies ist insbesondere bei einer grossen Anzahl von Observationen die einzig schlaue Möglichkeit. `generate` ist ein sehr mächtiger und umfassender Befehl. Details und viele Beispiele dazu findet man mit dem Befehl `help generate`. Hier nur ein einfaches Beispiel:

```
Syntax: gen temp_c=37
```

Mit diesem Syntax wird eine Variable namens `temp_c` kreiert. Alle Observationen erhalten den Wert 37.

3.8 Änderung eines Variablennamens

Ganz einfach: den Befehl `rename`, gefolgt vom alten und anschliessend vom neuen Variablennamen eingeben. Beispiel Syntax: `rename oldn newn`

3.9 Variablenlabels

Unter `newn` mag man sich vielleicht noch etwas vorstellen können, sicher ist aber das Wort `newn` in einer Grafik, welche einem grösseren Publikum präsentiert wird, etwas unverständlich. Da Variablennamen nie mehr als acht Zeichen haben sollten, muss man auf Variablenlabels ausweichen, sofern man verständliche Beschriftungen wünscht. Diese Labels lassen sich ganz einfach definieren:

```
Syntax: lab var newn "neuer Name"
```

`lab var` ist der Befehl für `label variable`⁹, gefolgt vom Namen der *zu labelnden* Variable `newn` und zum Schluss das Label für diese Variable. Sobald dieser Befehl ausgeführt ist, wird die Variable `newn` in Tabellen oder wo auch immer als Neuer Name angezeigt.

⁹der Befehl kann auch ausgeschrieben werden

3.10 Wertelabels

Damit die einzelnen, numerischen Werte einer Variable ebenfalls mit einem aussagekräftigen Text verknüpft werden können, definiert man eine Liste, in welcher jedem Wert einem Label gleichgestellt wird. Vorgehen: zuerst wird eine Werteliste erstellt, anschliessend wird diese mit einer bestimmten Variable verknüpft. Beispiel: Die Variable `newn` beinhaltet drei Kategorien: 1, 2 und 9. 1 steht für *neu*, 2 steht für *sehr neu* und 9 für *keine Angabe*.

```
Syntax: la de nlist 1"neu" 2"sehr neu" 3"n/a"
```

`la de` steht für `label define` und kreiert die Werteliste `nlist`, welche die anschliessenden Zahl/Aussagenpaare beinhaltet. Nun muss nur noch diese Werteliste mit der Variable `newn` verknüpft werden. Dies geschieht mit dem Befehl `label value`.

```
Syntax: la val newn nlist
```

Auf den Befehl folgt zuerst der Name der Variable und anschliessend der Name der Werteliste.

3.11 Verkürzung einer Stringvariable

Stata erlaubt innerhalb von Stringvariablen leider keine Wildcards. Somit müssen für Abfragen immer genaue Werte eingegeben werden. Solange dies mit numerischen Werten geschieht, ist das noch einigermaßen machbar, bei langen Stringvariablen wird es aber dann sehr mühsam. Eine Möglichkeit ist hier die Verkürzung der Strings auf eine bestimmte Anzahl Stellen. Beispiel: Eine Datenbank enthält Personennamen (Variable `perso`) in Textform. Damit nicht jedesmal der ganze Namen ausgeschrieben werden muss, wird eine neue Variable `pers5` kreiert, die die ersten fünf Buchstaben der Variable `perso` beinhaltet.

```
Syntax: gen str5 pers5=substr(perso,1,5)
```

`gen str5 pers5` kreiert eine neue, 5-stellige Stringvariable namens `pers5`, welche einen *Substring* `substr(perso,...)` der Variable `perso` darstellt. Die beiden Zahlen `,1,5)` am Schluss bedeuten, dass ab dem ersten Buchstaben gezählt, fünf Buchstaben der Variable `perso` übernommen werden. Stand also in der Variable `perso` noch „Hugentobler“, so lautet der entsprechende Wert der Variable `pers5` „Hugen“.

3.12 Rekodierung

Der Befehl `recode` erlaubt das Zusammenfassen und Ummodellieren von Kategorien und Werten. Bedingung ist, dass der Startwert aber auch der Zielwert numerisch sind. Ist dies nicht der Fall, muss mit dem Befehl `replace` gearbeitet werden. Der Befehl `replace` erlaubt die Änderung eines Werts einer Variable unter Berücksichtigung von Bedingungen, welche durch eine andere Variable gegeben sind. Hier einige Syntaxbeispiele:

- `recode varxy varyy (100=120)`

In diesem Fall werden alle Werte der beiden Variablen `varxy` und `varyy` auf 120 gesetzt, sofern sie zuvor den Wert 100 hatten. Alle anderen Werte werden nicht tangiert. Die Werte werden direkt in den Variablen `varxy` und `varyy` überschrieben.

- `recode varxy (100/230=1) (240/400=2), gen(varzz)`

In diesem Fall werden die Werte von 100 bis 230 zu 1 und die Werte von 240 bis 400 zu 2 rekodiert. Die Ausgangsvariable `varxy` wird nicht verändert, die rekodierten Werte werden in die neue Variable `varzz` geschrieben. Die Kreierung der neuen Variable ist eine Option des Befehls `recode`. Deshalb folgt der Befehl `gen(varzz)` nach einem Komma.

- `replace p_typ=1 if perso=="Hugentobler"`

Hier erhält die Variable `p_typ` den Wert 1, und zwar unabhängig davon, welchen Wert sie vorher hatte. Das Argument ist durch eine andere Variable¹⁰ definiert.

3.13 Auswahlprobleme mit gestapelten Daten

Viele Datenbanken haben eine ziemlich komplexe Struktur. Ein grosses Problem stellen beispielsweise *gestapelte* Daten dar. Dabei handelt es sich um Datenbanken, die in einander verschachtelt sind wie zum Beispiel mehrere Observationen zu ein und derselben Person, mehrere Observationen zu einem Land oder mehrere Sätze aus einem bestimmten Artikel in Datenbanken zu Medieninhalten. Manchmal verfügen solche Daten bereits über eine Variable, die die einzelnen Beobachtungen innerhalb eines Landes oder zu einer Person eindeutig identifizieren. Oft ist das aber nicht der Fall, und so muss man sich diese Variable zuerst erstellen. Ein Beispiel anhand folgender Ausgangsdaten:

¹⁰hier `perso`, vorausgesetzt der Wert dieser Variable ist „Hugentobler“

Tabelle 1: Beispieldaten

no_i	typ	var3	var4	...
22	1	12	blau	
22	3	12	blau	
22	1	12	blau	
22	1	12	blau	
23	3	12	blau	
23	1	12	blau	
24	1	20	rot	
...				

Um nun zum Beispiel einen Überblick über die Variable `typ` pro Identifikationsnummer `no_i` zu erhalten, muss man die jeweils erste Observation einer Observationsgruppe definieren.

```

sort no_i
gen id=0
  replace id=1 in 1
  replace id=1 if no_i !=no_i[_n-1]

```

In einem ersten Schritt sortiert man alle Beobachtungen nach der Identifikationsnummer `no_i`. Anschliessend wird eine neue Dummyvariable `id` mit dem Wert 0 erstellt. Die erste Beobachtung in der Datenbank wird manuell auf 1 gesetzt, alle weiteren werden dann unter der Bedingung, dass `no_i` ungleich (`!=`) dem vorangegangenen Wert ist, automatisch auf 1 gesetzt. Die Datenbank sieht dann so wie in Tabelle 2 aus. Eine neue Variable, die die ersten Beobachtungen einer Gruppe identifiziert, wurde hinzugefügt.

Tabelle 2: Beispieldaten

no_i	typ	var3	var4	id	...
22	1	12	blau	1	
22	3	12	blau	0	
22	1	12	blau	0	
22	1	12	blau	0	
23	3	12	blau	1	
23	1	12	blau	0	
24	1	20	rot	1	
...					

3.14 Zusammenfügen von mehreren Variablen in einer Variable

Dies ist dann hilfreich, wenn zusammenhängende, in verschiedenen Variablen erfasste Kodierungen überprüft werden müssen. Beispiele: Datumsangaben oder auf mehreren Ebenen kodierte Elemente. Vorgehen: Zuerst wird eine neue Variable kreiert, deren einzige Funktion die Trennung der verschiedenen Elemente durch ein bestimmtes Zeichen (keine mathematischen Operatoren oder das Semikolon verwenden, denn diese können von Stata missverstanden werden) oder einen Leerschlag ist.

Syntax: `gen str1 leer=" "`

`gen` steht für ‚generate new variable‘, `str1` definiert den Variablentyp¹¹, `leer` ist der Name der Variable, das Zeichen `=` leitet die Definition des Werts ein. Bei diesem Beispiel besteht der Wert beziehungsweise der Inhalt aus einem Leerschlag. Da es sich um Text handelt, muss der Leerschlag in Anführungs- und Schlusszeichen gesetzt werden. Anschliessend wird die zusammengefügte Variable generiert:

Syntax: `egen vx=concat(va leer vb leer vc)`

`egen` bedeutet, dass eine neue, der folgenden Funktion entsprechende Variable kreiert wird. `concat` heisst nichts anderes als verbinden oder verknüpfen, und das tut dieser Befehl dann auch mit den Variablen, die in der Klammer folgen. Der obenstehende Syntax zeigt auch die Anwendung der TrennungsvARIABLE `leer`. Kurz: mit diesem Syntax wird eine Variable namens `vx` kreiert, deren Inhalt aus den Variablen `va`, `vb` und `vc` besteht. Damit klar ist, welcher Teil aus welcher Variable stammt, wurden die drei Einträge durch einen Leerschlag, genauer gesagt durch Einfügen des Werts der Variable `leer` getrennt.

3.15 Generierung einer Datumsvariable aus den drei numerischen Variablen Tag, Monat, Jahr

Oftmals wird das Datum in drei einzelnen Variablen erfasst. Sortierfunktionen oder Ausschlusskriterien lassen sich aber erheblich einfacher definieren, wenn sich komplette Datumsangaben in einer Variable befinden und Stata weiss, dass es sich bei den numerischen Werten um ein Datum handelt. Hier der Syntax, mit dem Wasser zu Wein verwandelt werden kann:

```
generate date = mdy(month, day, year)
format dat %d
```

¹¹hier 1-stelliger String; bei numerischen Variablen kann dies weggelassen werden

Der erste Befehl `generate` baut eine Datumsvariable namens `date` durch die Kombination der drei Variablen `month`, `day` und `year`. Der zweite Befehl (kann in einem Syntax nicht auf der gleichen Linie stehen) `format` definiert das Datumsformat, mit anderen Worten die Art und Weise, wie dieses Datum schlussendlich angezeigt wird.

3.16 Mittelwerte in separater Datenbasis sichern

Mit dem Befehl `means` kann man sich die Mittelwerte anzeigen lassen. Möchte man aber weiterführende Analysen mit Mittelwerten einer oder mehrerer Variablen über alle oder ausgesuchte Observationen durchführen, so ist es am einfachsten, diese zuerst in einer separaten Datenbasis zu sichern. Dafür steht der Befehl `collapse` zur Verfügung. Hier ein Anwendungsbeispiel:

```
#delimit ;
      collapse (mean) target_var=right_left (median)
      target_var2=right_left, by (ptype)
;
#delimit cr
```

In diesem Beispiel beinhaltet die Ausgangsvariable `right_left` die Links-Rechts-Positionierungen von politischen Akteuren. Mit diesem Syntax erhält man eine neue Datenbasis, die die Mittelwerte und die Mediane nach Parteytyp (durch die Variable `ptype` definiert, die in diesem Beispiel als Option verwendet wird) enthält.

3.17 Mittelwerte über verschiedene Variablen

*thanks to Silja*¹² Die folgende Syntax bildet eine neue Variable, welche pro Observation (Linie) den Mittelwert mehrerer anderer – bereits vorhandenen – Variablen enthält:

```
gen prox1_4 = (pos_int + pos_lev + pos_sco +pos_comp)/4
```

Die neue Variable `prox1_4` enthält als Werte die arithmetischen Mittel der vier Variablen `pos_int`, `pos_lev`, `pos_sco` und `pos_comp`. Sobald aber eine Observation auf einer dieser vier Variablen fehlt (missing value), kriegt auch die neue Variable `prox1_4` den Wert `missing`. Wenn das nicht erwünscht ist, wenn also die neue Variable den Mittelwert der vorhandenen Werte erhalten soll, ist folgende Syntax nötig:

```
egen prox1_4 = rmean (pos_int pos_sco pos_lev pos_comp)
```

`rmean` ist eine erweiterte Funktion von `egen` (mehr zu diesem Befehl auf Seite 21). Wenn also zum Beispiel eine Observation auf der Variable `pos_comp` fehlt, berechnet `rmean` einfach den Mittelwert aus den andern drei Variablen.

¹²<mailto:silja.haeusermann@pwi.unizh.ch>

3.18 Löschen von Fällen und Variablen

Dies geschieht mittels den Befehlen `drop` und `keep`. Mit `drop [expression]` wird entsprechend der Bedingung gelöscht und mit `keep [expression]` wird alles gelöscht, ausser den Elementen, die der Bedingung entsprechen. Beispiele:

- Syntax: `drop var1`
Löscht die Variable `var1`.
- Syntax: `drop if var1==12`
Löscht alle Observationen, die die Bedingung `var1==12` erfüllen.
- Syntax: `keep var1 var2 var3 var4 if var1==6`.
Wenn die Variable `var1` den Wert 6 besitzt werden nur die Variablen `var2 var3 var4` der betreffenden Fälle behalten. Alles anderen Variablen und/oder Observationen werden gelöscht.

3.19 Temporäre Transformationen

Mit dem Befehlspar `preserve [...] restore` können Transformationen oder Modifikationen temporär angewendet werden. In folgendem Beispiel werden für eine Tabelle sämtliche Observationen vor 1980 gelöscht. Nach Anzeige der Tabelle wird mit dem Befehl `restore` die Datenbank wieder in ihren ursprünglichen Zustand versetzt:

```
preserve
  drop if year<1980
  tab varx
restore
```

4 Erste Analysen mit Stata

4.1 Bivariate Verteilungen

Der Befehl `tabulate` wurde bereits vorgestellt. Mit seiner Hilfe lassen sich Kreuztabellen erzeugen und man erhält einen raschen Überblick über die Kennziffern. Für komplexere Kreuztabellen steht der Befehl `table` zur Verfügung. Wünscht man Korrelationskoeffizienten, so kann man zwischen verschiedenen Möglichkeiten auswählen. Hier einige Beispiele:

Tabelle 3: Befehle für Korrelationskoeffizienten

Koeffizient	Befehl
Pearsonscher Korrelationskoeffizient	<code>correlate</code>
Paarweise Anzeige der Korrelationskoeffizienten der <i>varlist</i> oder – falls keine <i>varlist</i> angegeben – sämtlicher Variablen	<code>pwcorr</code>
Spearman'scher Korrelationskoeffizient (Rangkoeffizient)	<code>spearman</code>
Kendalls Rangkoeffizient	<code>ktau</code>

Syntax: Auf die Eingabe des Befehls folgen die beiden Variablen beziehungsweise der *varlist*, anschliessend gefolgt von allfälligen Optionen.

4.2 Drittvariablenkontrolle

Die meisten Analysebefehle lassen sich durch `if` einschränken, wie dies bereits besprochen wurde. Eine andere Möglichkeit bietet der Befehl `by`. Mit seiner Hilfe lassen sich Befehle auf bestimmte Subgruppen anwenden.

Syntaxbeispiel: `by sex: tab var1 var2`

`sex` ist in diesem Beispiel eine Dummyvariable; mit diesem Syntax werden zwei Kreuztabellen generiert, eine für Frauen, eine zweite für Männer. Der Doppelpunkt nach der sich auf den Befehl `by` beziehenden Variable ist zwingend zu setzen. Damit der Befehl `by` funktioniert, müssen die Observationen zuerst nach der Drittvariable sortiert (Befehl: `sort varlist`) werden. Die Sortierung kann aber auch direkt mit dem Befehl `by` kombiniert werden, man schreibt einfach `bys` anstelle von `by`.

4.3 Regression

Stata kennt eine Vielzahl von verschiedenen Regressionsmodellen. Eine einfache OLS-Regression erhält man durch den Befehl `regress`.

Syntax: `regress depvar var1 var2 var3`

Auf den Befehl `regress` folgt zuerst die abhängige Variable `depvar`, anschliessend folgen die unabhängigen Variablen.

5 Schlusswort

Wie bereits einleitend erwähnt, stellt dieses Handbüchlein keinen Anspruch auf Vollständigkeit, sondern bietet Hilfe bei ersten Schritten. Nun, eine Einleitung hat per Definition ein Ende, und das ist im Moment hier. Stata ist ein sehr umfassendes Statistikprogramm, welches neben seinen zahlreichen statistischen Modellen auch über ausgezeichnete graphische Fähigkeiten verfügt, deren Entdeckung sich lohnt. Aber dafür gibt's nun die Stata-Hilfe, die offiziellen Handbücher oder auch weitere Literatur, zum Beispiel Kuhn und Ruf (2006), deren Einführung wesentlich weiter geht als die vorliegende – netterweise orientiert sich ihre Einleitung und teilweise auch die Struktur offenbar am diesem Handbüchlein, was die weitere Arbeit mit ihrer Einführung sehr einfach macht... ;-)

Literatur

- Acock, Alan C. 2008. *A Gentle Introduction to Stata*. 2. Auflage. College Station, TX: Stata Press.
- Juul, Svend. 2004. *Introduction to Stata 8*. Universität Aarhus, [8.11.2006].
http://www.soziologie.uni-halle.de/langer/logitreg/books/juul/juul_stata8.pdf
- Kohler, U. und F. Kreuter. 2001. *Datenanalyse mit Stata*. München: Oldenbourg.
- Kuhn, Andreas und Oliver Ruf. 2006. *Einführung in die Statistiksoftware STATA*. Zürich: Institute for Empirical Research in Economics, Working Paper Nr. 277.
<http://www.iew.unizh.ch/wp/iewwp277.pdf>
- Pfeffer, Fabian, Stephan Lindner und Bernd Weiss. 2004. *Erste Schritte mit Stata*. AG Statistische Methoden der Sozialwissenschaften, Köln, [8.11.2006].
<http://www.metaanalyse.de/material/master.pdf>
- Resources to help you learn and use Stata*. 2006. UCLA [5.6.2004].
<http://www.ats.ucla.edu/STAT/stata/>