GESStabs

Daten aus CSV-Dateien

Gesellschaft für Software
in der Sozialforschung mbH

Waterloohain 6 - 8
22769 Hamburg
Tel.: 040 - 853 753 - 0
Fax: 040 - 853 753 - 33
www.gessgroup.de

Highlights:

GESStabs verarbeitet in einem Lauf beliebig viele CSV-Files.
Deren Struktur muss nicht identisch sein.
Namenskonventionen erlauben einfache Erzeugung von Mehrfachnennungs- und Alpha-Variablen.

Die Felder der ersten Zeile der CSV-Datei werden als Variablennamen interpretiert.

Die folgende Datei (null.csv),

	A	B	C	D	E
1	@testq1	@testq2	testq4	testq5	
2	aa	bb	2	1	
3	aa	bb	3	2	
4	aa	bb	2	1	
5	aa	bb	2	2	
6	aa	bb	3	1	
7	aa	bb	2	2	
8					
9					
10					

abgespeichert als CSV-File:

```
@testq1; @testq2; testq4; testq5;
"aa";"bb";2; 1;
"aa";"bb";3; 2;
"aa";"bb";2; 1;
"aa";"bb";2; 2;
"aa";"bb";3; 1;
"aa";"bb";2; 2;
```

erzeugt, wenn sie als CSVINFILE eingelesen wird, 4 Variablen, testq1, testq2, testq4 und testq5. GESStabs hat dem @ vor den beiden ersten Variablennamen entnommen, dass es sich bei diesen Variablen um Alpha-Variablen handeln soll. Sonst ist über diese Variablen nichts bekannt, VARTITLE, Fragentexte, Labels etc. sind zu diesem Zeitpunkt undefiniert.

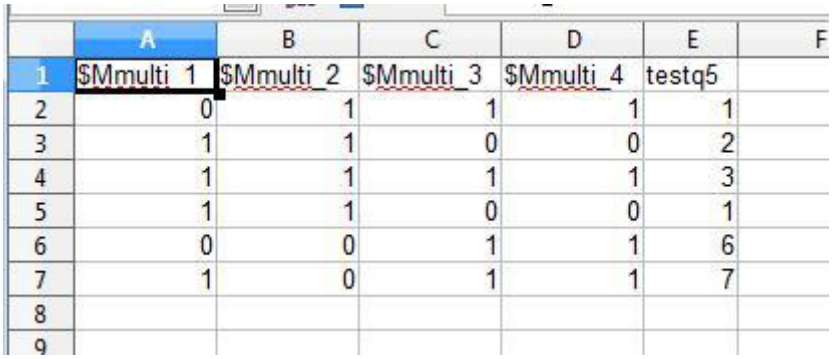
In einem GESStabs-Lauf können beliebig viele solcher Dateien verarbeitet werden. Würde anschließend eine zweite Datei (null_2.csv)

	A	B	C	D	E	F
1	@testq2	testq4	testq5	@testq1		
2	bb	2	1	aa		
3	bb	3	2	aa		
4	bb	2	1	aa		
5	bb	2	2	aa		
6	bb	3	1	aa		
7	bb	2	2	aa		
8						
9						

eingelesen werden, so erkennt die Software, dass hier dieselben Variablen eingelesen werden sollen. Die Reihenfolge der Spalten ist unerheblich. Die Variablen haben GESS-intern eine Reihenfolge, die sich daraus bestimmt, in welcher Reihenfolge sie erzeugt wurden. Die Reihenfolge bleibt also testq1, testq2, testq4,

testq5, da das Einlesen der zweiten Datei keine neuen Variablen hinzufügt.

In weiteren Files können aber durchaus neue Variablen hinzugefügt oder Variablen ausgelassen werden. Es wird noch eine dritte Datei (null0_m.csv) eingelesen:

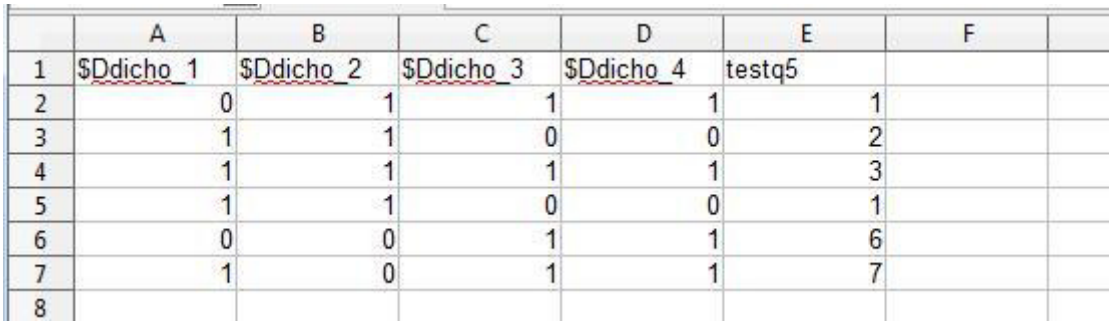


	A	B	C	D	E	F
1	$Mmulti_1	$Mmulti_2	$Mmulti_3	$Mmulti_4	testq5	
2	0	1	1	1	1	
3	1	1	0	0	2	
4	1	1	1	1	3	
5	1	1	0	0	1	
6	0	0	1	1	6	
7	1	0	1	1	7	
8						
9						

Diese Datei enthält 5 Spalten. Daraus erzeugt GESStabs folgerichtig 5 atomare Variablen. Die ersten 4 Spalten enthalten aber für GESStabs zusätzliche Informationen zur Interpretation: Das $ bedeutet, dass dies Variablen zu einer Mehrfachnennungsvariablen gehören. Dem $M entnimmt GESStabs, dass es sich um eine VARFAMILY bzw. MULTIQ handelt; also eine Variable, die eine 1., 2., 3. Nennung kennt. GESStabs erkennt auch, dass die 3 Spalten D bis F zusammengehören. Wenn eine $M-Variable erkannt ist, wird der „Namensstamm“ isoliert, Am Beginn des Wortes wird die Typkennnzeichnung ($M), und am Ende die Serienkennung (_1, _2, _3, _4) entfernt. Es bleibt der Stammname „multi“, und unter diesem Namen ist diese Mehrfachnennungsvariable anschließend bekannt[1].

Außerdem sind die Spalten @testq1, @testq2 und testq4 nicht vorhanden. Das bewirkt, dass diese Variablen in den Fällen, die aus dieser Datei gelesen werden, auf MISSING gesetzt werden.

Nachzutragen ist nur noch, wie es mit DICHOQ funktioniert, den 0/1 Variablen. Eigentlich genauso, nur dass die Kennzeichnung hier „$D“ lautet statt „$M“.



	A	B	C	D	E	F
1	$Ddicho_1	$Ddicho_2	$Ddicho_3	$Ddicho_4	testq5	
2	0	1	1	1	1	
3	1	1	0	0	2	
4	1	1	1	1	3	
5	1	1	0	0	1	
6	0	0	1	1	6	
7	1	0	1	1	7	
8						

In den 4 Spalten A .. D der vierten Datei (null0.csv) findet GESStabs die 4 atomaren Variablen der DICHOQ dicho. In Spalte E ist noch die Variable testq5 gespeichert. testq1, testq2 und testq4 sind in diesen sechs Fällen MISSING.

Insgesamt ist dies ein einfaches und robustes Verfahren, Variablen und deren Inhalte in CSV zu speichern und aus CSV zu lesen, und dabei die GESS-Variablentypen ALPHA, MULTIQ und DICHOQ zu erhalten. Die Modifikationen der Variablennamen bestehen lediglich aus

[1] GESStabs schreibt CSV-Dateien mit entsprechend anhand der Struktur modifizierten Variablennamen, wenn man dies mit dem Statement MODIFYCSVNAMES = YES; verlangt.

@ für Alpha-Variablen
$M für MULTIQ
$D für DICHOQ.

Wenn man mehrere Dateien einliest, kann man überwachen, welche Variablen aus welcher Datei gelesen werden. Die Anweisung lautet:

```
CSVINPROTOCOL = csvdoku.txt;
CSVINFILE = csv.csv;
CSVINFILE = csv2.csv;
```

Die ausgegebene Diagnose in csvdoku.txt lautet:

```
CSVINFILE = null.csv                          (n=6)

List of new variables:
testq1                            Atom        ALPHA
testq2                            Atom        ALPHA
testq4                            Atom
testq5                            Atom

List of known variables not in null.csv:


CSVINFILE = null_2.csv                        (n=6)

List of new variables:

List of known variables not in null_2.csv:


CSVINFILE = null0_m.csv                       (n=6)

List of new variables:
multi                             MULTIQ

List of known variables not in null0_m.csv:
testq1
testq2
testq4


CSVINFILE = null0.csv                         (n=6)

List of new variables:
dicho                             DICHOQ

List of known variables not in null0.csv:
testq1
testq2
testq4
multi
```

Die aus CSV-Dateien erzeugten Variablen sind „nackt“. Sie haben einen Namen, Strukturinformationen (ALPHA, Mehrfachnennungen) und Werte. Sie haben vieles nicht: keinen VARTITLE, keinen TEXT, keine LABELS, keine Filter. Grundsätzlich kann man diese Informationen mit den Standardstatements VARTITLE, TEXT, LABELS hinzufügen. Vielfach liegt diese Information aber als GESS Script vor, z.B. in der Form von VAR.INC Dateien, wie sie von Q.dot erzeugt werden.

Wenn Dateninput via CSVINFILE verlangt ist, schaltet GESStabs in einen speziellen Interpretationsmodus bei VARIABLE, VARIABLES, MULTIQ und DICHOQ-Statements. Im normalen Interpretationsmodus erzeugen diese Statements neue Variablen, und es dürfen noch keine Variablen dieses Namens existieren. Im CSVINFILE-spezifischen Interpretationsmodus kehrt sich diese Prüfung um: es muss bereits eine Variable dieses Namens existieren, und der Typ muss dem entsprechen, was aus dem CSV-File entnommen wurde.

Wandlung von Spaltenfixierten Datensätzen zu CSV.

Diese Form von CSV-Files in Kombination mit einem beschreibenden Script wie VAR.INC ist praktisch, es bietet folgende Vorteile:

1. Mehrere Wellen mit unterschiedlichen Variablensets bedürfen keiner speziellen Aufmerksamkeit.
2. Der Datensatz kann in UTF8 kodiert sein.
3. Einfaches Anschauen und Korrigieren in EXCEL oder OpenOffice. Variablennamen statt Spalten.
4. Datenlieferungen in CSV- und EXCEL-Format ohne Aufwand.

Die ankommenden Datenlieferungen sind allerdings oft spaltenfixierte Datensätze. Wie kann man diese einfach wandeln? Am einfachsten schreibt man am Ende des Scripts, das den spaltenfixierten Datensatz verarbeitet:

```
MODIFYCSVNAMES = YES;
ASCIIOUTFILE DELIMITED = csv.csv;
ASCIIOUT ALL;
```

Für alle aus dem Standardinputfile gelesenen Variablen ist damit alles erledigt. Alle Daten aus dem DATAFILE werden auf diese Weise in den CSV-Datensatz übertragen: GESStabs interpretiert ALL so, dass alle Variablen aus dem Standardinput übertragen werden. Variablen, die mit COMPUTE o.ä. berechnet werden, und OPENQ bleiben hierbei außen vor.

Die erste Gruppe kann man durch ein ergänzendes ASCIIOUT-Statement in den Datensatz transferieren. Sei z.B. die Variable Status berechnet worden, dann kann man so deren Inhalt zusätzlich übertragen:

```
MODIFYCSVNAMES = YES;
ASCIIOUTFILE DELIMITED = csv.csv;
ASCIIOUT ALL;
ASCIIOUT Status = 1;
```

Der Standardanfang eines Script, in dem man diese CSV-Datei verarbeitet wird, würde so lauten:

```
CSVINFILE = csv.csv;
INCLUDE = var.inc;
CODEBOOK ;
```

Besonderes Augenmerk muss man den OPENQ widmen, da sie viele unterschiedliche Funktionen erfüllen können. Aus der jeweiligen Funktion ergibt sich, ob, und in welcher Form, man sie im CSV-Datensatz abgelegt sehen möchte.

Ein paar typische Szenarien:

1. Für die offenen Fragen soll ein Codeplan erstellt werden, und zu einem späteren Zeitpunkt sollen diese Codes dem Datensatz und der Auswertung hinzugefügt werden.

2. Der Codeplan liegt bereits vor, und die OPENQ-Files enthalten die Kodierung.

3. Eine Kodierung ist nicht geplant; die Antworten auf die offenen Fragen sollen verbatim tabelliert und in den Datensatz eingefügt werden.

4. Der Kunde weiß es noch nicht.

Ein paar typische Lösungen:

1. Die Struktur, dass die Informationen zu den offenen Antworten in einem separaten File liegen, ist sinnvollerweise beizubehalten. Die Originaltexte sollen im Datensatz gar nicht auftauchen. Bei der weiteren Arbeit mit dem CSV-File soll die Benutzung der OPENQ-Datei genau so funktionieren wie beim spaltenfixierten Datensatz, d.h. die CSV-Datei enthält die Schlüsselinformation zum Zugriff auf die OpenQ-Datei.

2. Diametral anders: die Ergebnisse der Kodierung sollen als MULTIQ im CSV-File enthalten sein. Das OpenQ-File wird in der Folge nicht mehr benötigt.

3. Die wörtlichen Antworten sollen als ALPHA-Variablen im CSV-File enthalten sein. Die OpenQ-Datei hat ihren Zweck erfüllt und spielt in der Auswertung keine Rolle mehr.

4. Wie 1. Die OpenQ-Datei wird beibehalten. Man sieht den später kommenden Wünschen des Kunden gefasst entgegen.

Ein paar Scriptschnipsel:

Zu 1: Struktur beibehalten

Speichern:

```
MODIFYCSVNAMES = YES;
ASCIIOUTFILE DELIMITED = csv.csv;
ASCIIOUT ALL;
```

Lesen:

```
CSVINFILE = csv.csv;
KEY OPENQFILE = "$SYSCASE $1";
NEWOPENFORMAT = YES;
OPENQFILE = "<filename>";
INCLUDE = var.inc;
```

Wie funktioniert das? Das Speichern kennen wir schon. Alle Variablen, die aus dem spaltenfixierten Datensatz gelesen wurden, werden in die CSV-Datei übertragen. Die Variablen zu den offenen Fragen sind nicht enthalten.

Im einlesenden Script wird erst das CSV-File gelesen. Nachdem der Header gelesen ist, sind die dort benannten Variablen bekannt. Für das OPENQFILE muss die Schlüsselvariable bekannt gemacht werden. Bei spaltenfixierten Datensätzen stellt die CASENUMBER-Anweisung die benötigte Information bereit. Diese Information wird intern in einer Variable $SYSCASE $1" verwaltet. Da sie aus dem ASCII-Datensatz gelesen wurde, wurde sie von „ASCIIOUT ALL;“ mit übertragen, sie steht als Schlüsselvariable zur

Verfügung. Daher werden die Inhalte im OpenQ-File richtig zugeordnet. Die Inhalte dieser Datei können nun sowohl verbatim oder auch als Codes verwendet werden. Unten steht die Auswertung einer OPEN-Variablen mit GLOBALOPENASALPHA = YES;

Table 92:

Planen Sie den Kauf eines PKW in den nächsten 1-2 Jahren?

f23_1_97_open	Abs.	Col %
SUV	18	62
MINI	1	3
Kleinwagen	1	3
Coupé	5	17
kompakt	1	3
Geländewagen	1	3
Kult	1	3
Bus	1	3
N =	29	29

GESS mbH *

Da die Auswertung der OpenQ-Datei erst in dem Script passiert, das die CSV-Datei auswertet, gibt es bei dieser Konstruktion die volle Freiheit, ob man Codes oder wörtliche Zitate auswerten möchte.

Tabelliert man die Datei ohne GLOBALOPENASALPHA, und wenn die OpenQ-Datei (hoffentlich) richtig kodiert wurde, sieht das Ergebnis so aus:

Table 92:

Planen Sie den Kauf eines PKW in den nächsten 1-2 Jahren?

f23_1_97_open	Abs.	Col %
SUV	18	62
Coupé	5	17
Sonstiges	6	21
N =	29	29

GESS mbH *

Zu 2: Kodierung als MULTIQ im CSV-File

Schreiben:

```
NEWOPENFORMAT = YES;
OPENQFILE = "<filename>";
MODIFYCSVNAMES = YES;
ASCIIOUTFILE DELIMITED = csv.csv;
ASCIIOUT ALL;
M_OPEN 5 = f23_1_97_open;
```

Um die kodierten Werte ins CSV-File zu transportieren, muss man die OPEN-Variable in eine Standard-Variable speichern. Das geht am komfortabelsten mit dem M_OPEN-Statement. Es erzeugt eine MULTIQ-Variable mit einem modifizierten Variablennamen, indem „M_“ vorangestellt wird. Die Codes der OPEN-Variable werden dann mit LOAD in diese Multi übertragen. So erzeugte Variablen (hier f23_1_97_open) bekommen intern eine Property, die die automatische Übertragung in den CSV-Datensatz bewirkt.

Lesen:

```
CSVINFILE = csv.csv;
INCLUDE = var.inc;
LABELS m_f23_1_97_open =
1 "SUV"
```

```
2 "Coupé"
3 "Sonstiges"
;
```

Das CSV-File enthält jetzt die MULTIQ, die die Codes aus der OPEN-Variablen aufgenommen hat. Diese MULTIQ muss zur Darstellung mit dem Codeplan versehen werden. Dann kann tabelliert werden.

Table 1:

m_f23_1_97_open	Abs.	Col %
SUV	18	62
Coupé	5	17
Sonstiges	6	21
N =	29	29

GESS mbH

Zu 3. OpenQ als ALPHA-Variablen im CSV-File

Schreiben:

```
GLOBALOPENASALPHA = YES;
NEWOPENFORMAT = YES;
OPENQFILE = "<filename>";
MODIFYCSVNAMES = YES;
ASCIIOUTFILE DELIMITED = csv.csv;
ASCIIOUT ALL;
```

Das OpenQ-File wird wörtlich gelesen, wegen GLOBALOPENASALPHA. Variablen mit der Eigenschaft OPENASALPHA werden automatisch in CSV-File übertragen.

Lesen:

```
CSVINFILE = csv.csv;
INCLUDE = var.inc;
```

Table 42:

f23_1_97_open	Abs.	Col %	Cum. %
MISSING	313	92	92
SUV	18	5	97
MINI	1	0	97
Kleinwagen	1	0	97
Coupé	5	1	99
kompakt	1	0	99
Geländewagen	1	0	99
Kult	1	0	100
Bus	1	0	100
N =	342	342	342

GESS mbH

Zu 4.Nichts Genaues weiß man nicht

Genauso behandeln wie oben unter 1. Man adressiert aus dem SCV-File heraus das OpenQ-File und hat bei allen Variablen die Freiheit, wie sie zu behandeln sind.