Erweiterung der Ausgabe von PROC GMAP mit dem Data Step Graphics Interface (DSGI)

Grischa Pfister
Abteilung für Wirtschafts- und Sozialgeographie, Computerkartographie und GIS
Geographisches Institut der Ruprecht-Karls-Universität Heidelberg

Abstract

SAS/GRAPH bietet mit GMAP eine Prozedur, die kartographische Darstellungen ermöglicht, allerdings ist das Ergebnis nicht unbedingt zufriedenstellend. Es kann lediglich eine Legende erzeugt werden, Information über die Verteilung der Klassen fehlt. Auch die Möglichkeiten, ein ansprechenderes Layout zu entwerfen, sind beschränkt. Abhilfe schafft hier nur die Verwendung von ANNOTATE-Data Sets oder eben – was in diesem Beitrag vorgestellt werden soll – das Data Step Graphics Interface (DSGI).

DSGI besteht aus einer Reihe von Funktionen und Routinen mit denen Graphikelemente gezeichnet und Eigenschaften dieser Graphikelemente festgelegt bzw. abgefragt werden. Ergebnis ist eine reguläre SAS-Vektorgraphik vom Typ GRSEG.

Zum Zwecke der Automatisierung wurde das DSGI-Layout in einem Makro zusammengefaßt.

Einleitung

Zwar bietet SAS/GRAPH mit der Prozedur GMAP die Möglichkeit, in SAS relativ einfach Karten zu erzeugen, doch leider läßt die Darstellung zu wünschen übrig. Es kann nur eine Legende gezeichnet werden, die die Einteilung der Klassen darstellt. Wieviele der Gebietseinheiten auf die einzelnen Klassen entfallen, kann der Betrachter der Karte nicht entnehmen. Dabei ist aber neben der Wahl der Klassenbreiten die Verteilung auf die einzelnen Klassen maßgeblich für den Aussagegehalt der Karte.
Auch die Layout-Optionen sind unbefriedigend. Über die globalen Graphikoptionen (GOPTIONS) kann zwar eine Hintergrundfarbe (cback=farbe) festgelegt werden, damit wären die Möglichkeiten aber auch schon ausgereizt.
Ist ein Mehr an Gestaltung gefragt, muß entweder auf die ANNOTATE-Facility - eventuell in Kombination mit PROC GREPLAY - zurückgegriffen werden, um z.B. eine Karte mit einem Schatteneffekt zu versehen. Oder es wird das Data Step Graphics Interface verwendet, das die benötigten Funktionen von ANNOTATE und GREPLAY in einer konsistenten Programmiersprache vereint. Hier können sowohl einzelne Graphikelemente wie Rechtecke, Linien, Texte gezeichnet, als auch bereits erzeugte SAS-Vektorgraphiken (Entrytyp GRSEG) in definierte Bildausschnitte hineingesetzt werden.
In diesem Beitrag werden zunächst die Grundprinzipien von DSGI erläutert, bevor auf die Konzeption des Kartenlayouts und die Umsetzung in DSGI eingegangen wird.

Das Data Step Graphics Interface (DSGI)

Das Data Step Graphics Interface ist eine Graphikprogrammiersprache, mit der in Datenschritten oder in der Screen Control Language (SCL) SAS-Graphiken erzeugt werden können. Anders als bei der ANNOTATE-Facility, die einen Data Set erzeugt, der erst von einer Prozedur interpretiert werden muß, wird die Graphik in DSGI direkt erstellt, der Data Set selbst enthält keine verwendbaren Informationen.
DSGI besteht aus einer Reihe von Funktionen und Routinen, mit denen Parameter gesetzt oder abgefragt und Graphikelemente gezeichnet werden können, oder zwischen verschiedenen „Stationen“ hin- und hergewechselt werden kann. Diese „Stationen“ stellen verschiedene Schritte dar, die für die Graphikerstellung notwendig sind und nacheinander durchlaufen werden müssen. Sie heißen:

Tabelle 1: DSGI-Stationen

Station	Aufgabe
GKCL *graphic kernel closed*	Einstellungen, die die gesamte Graphik betreffen wie z.B. globale Graphikoptionen; Zielkatalog
WSAC *workstation open*	Initialisierung von DSGI; Abfrage von Systemeinstellungen; Setzen von Attributen und Aufteilung der Bildschirmbereiche
SGOP *segment open*	Zeichnen von Graphikelementen; Wechsel zwischen Bildschirmbereichen; Setzen von Attributen

Von jeder Station aus kann immer nur um einen Schritt weitergegangen werden, deshalb läuft der gesamte Prozeß der Erstellung einer DSGI-Graphik in fünf Schritten ab.

Tabelle 2: Die fünf Schritte der Graphikerstellung mit DSGI

1.	`DATA _null_ ;`	GKCL	Zu Beginn des Datenschrittes ist die graphic kernel geschlossen. Hier können der Zielkatalog eingestellt oder globale Graphikoptionen gesetzt werden
2.	`rc = ginit();`	GKCL ⇨ WSAC	Mit diesem Schritt werden die DSGI-Funktionalitäten geladen. Jetzt können Bildschirmbereiche und Attribute definiert werden.
3.	`rc = graph('clear', 'Name', 'Beschrei-bung');`	WSAC ⇨ SGOP	Die Graphik wird geöffnet und es können verschiedenste Graphikelemente gezeichnet werden.
4.	`rc = graph( 'update')`	SGOP ⇨ WSAC	Die erstellte Graphik wird geschlossen und gespeichert.
5.	`rc = gterm();` `run;`	WSAC ⇨ GKCL	DSGI wird beendet.

Die wichtigsten DSGI-Konzepte

CALL-Routinen und Funktionen

DSGI besteht aus CALL-Routinen und Funktionen. CALL-Routinen werden benutzt, um Einstellungen abzufragen, im vorliegenden Makro finden sie keine Verwendung.

Syntax:

```
call gask('Einstellung', parameter1 <, ...,parametern>, return-code);
```

Funktionen dienen dem Setzen von Parametern und Zeichnen von Graphikelementen, außerdem schalten sie zwischen den verschiedenen DSGI-Stationen um. Das Funktionspaar ginit() und gterm() beispielsweise initialisiert bzw. beendet das Data Step Graphics Interface. Beide Funktionen werden ohne Parameter aufgerufen.

Syntax:

```
rc = ginit();
rc = gterm();
```

Sämtliche Einstellungen, angefangen vom Arbeitskatalog bis hin zu den Attributen für die einzelnen Graphikelemente, werden mit Hilfe der Funktion gset() vorgenommen. Diese Einstellungen sind solange gültig, bis sie mit anderen Werten überschrieben werden.

Syntax:
`rc = gset('Einstellung', parameter1 <, ..., parametern>);`

Alle Aufgaben, die mit der Graphik an sich zu tun haben, werden über die Funktion graph() abgewickelt. Dazu gehört das Öffnen und Schließen (vergl. Tabelle 2, Punkt 3 u. 4), aber auch das Einfügen bereits bestehender (und im gleichen Katalog befindlicher) Graphiken.

Syntax:
`rc = graph('aufgabe', parameter1 <,...,parametern>);`

Für das Zeichnen von Graphikelementen ist die Funktion gdraw() zuständig, sie kann Linien, Rechtecke, Kreisbögen u.v.m. darstellen. Wichtig ist, daß jedes der Graphikelemente spezifische Eigenschaften hat, die mit Hilfe von gset() eingestellt werden, z.B. Farbe, Füllmuster, Schriftart, Linientyp etc.

Syntax:
`rc = gdraw('Graphikelement', parameter1 <, ..., parametern>);`

Viewports und Windows

Mit Hilfe von Viewports kann der Graphikbereich in verschiedene Abschnitte unterteilt werden. Die Angabe erfolgt in Prozent, definiert wird das untere linke und das obere rechte Eck des Viewports. Der Standard-Viewport umfaßt den gesamten Graphikbereich, seine Koordinaten lauten (0/0) und (1/1), d.h. bei der Definition von Viewports wird die 1 gleich 100% gesetzt.
Das „window" legt das „Koordinatensystem" für einen Viewport fest. Auch hier werden wieder die untere linke und die obere rechte Ecke definiert. Die Werte stellen Prozent der Viewportgröße dar, d.h. ein Window mit den Ecken (0,0) und (100,100) entspricht genau dem Viewport. Hier müssen als Werte „normale" Prozente verwendet werden.
Werte größer 100 sind möglich, denn Windows können benutzt werden, um in Graphiken zu zoomen, auf diese Möglichkeit kann hier aber nicht eingegangen werden.
Zusammengehörige Viewports und Windows haben eine gleichlautende „transformation number", mit deren Hilfe sie aktiviert werden. Standard-Viewport und -Window haben die Nummer 0 und können nicht verändert werden, es dürfen bis zu 20 eigene Viewports und Windows definiert werden.

Syntax:

`rc = gset('viewport',n, llx, lly, urx, ury);`	n ist die „transformation number", sie kann zwischen 1 und 20 liegen; llx, lly, urx, ury $\in$ [0,1]
`rc = gset('window', n, llx, lly, urx, ury);`	n $\in$ [1,20]; llx, lly, urx, ury $\in$ N
`rc = gset('transno',n);`	aktiviert die „transformation number" n Default ist 0

Die verwendeten Graphikelemente und ihre Eigenschaften

Im Makro werden nur wenige DSGI-Graphikelemente benutzt, die hier kurz erläutert werden sollen, dabei wird auch nicht auf alle, sondern nur auf die verwendeten Eigenschaften einge-

gangen. Zunächst muß allerdings noch eine Vorbemerkung zu der Farbzuordnung in DSGI gemacht werden. Anders als in ANNOTATE kann die Farbe für Text, Linien, Füllungen etc. hier nicht direkt angeben werden. Stattdessen wird eine Farbliste definiert, und dann die Farbeigenschaft des Graphikelements auf den numerischen Index der Farbliste eingestellt. Es müssen also alle Farben registriert werden, bevor sie verwendet werden können.

Syntax:

`rc = gset('colrep', n, 'Farbe');`	n: Index, der mit der Farbe verbunden wird Farbe: jede gültige SAS-Farbbezeichnung; RGB- oder HLS-Definition

Rechtecke

Rechtecke werden über zwei Ecken, (x_1 ,y_1) und (x_2 ,y_2) definiert. Daneben können sie verschiedene Eigenschaften haben, eine Füllart, ein Füllmuster und eine Füllfarbe.

`rc = gdraw('bar', x1, y1, x2, y2);`	x_1, y_1: Anfangspunkt y_2, y_2: Endpunkt
`rc = gset('filtype', 'Füllart');`	Füllart kann sein*: hatch, hollow, pattern, solid*; nur im Falle von *hatch* und *pattern* wird das unter ‚filstyle' eingestellte Füllmuster verwendet; Default ist 1
`rc = gset('filstyle', 'Füllmuster');`	vgl. Tabelle 21.1 in SAS/GRAPH: Reference Vol. 1, S. 709; Default ist 1 (Pattern: X1; Hatch: M1X)
`rc = gset('filcolor', Index);`	Index ist die Nummer, die unter gset(‚colrep', index, ‚Farbe'); für die gewünschte Farbe vergeben wurde; Default ist 1

Linien

Linien bestehen aus einer Reihe von XY-Paaren. In DSGI werden diese Paare aber nicht zusammen, sondern getrennt angegeben. Der erste Parameter gibt an, aus wievielen XY-Paaren die Linie besteht, dann folgen erst alle X-, dann alle Y-Werte. Linien haben die Eigenschaften Linienfarbe, -typ und -breite.

`rc = draw ('line', n, x1, ..., xn, y1, ...,yn);`	n = Anzahl der XY-Paare x_1, ..., x_n: Liste der X-Werte y_1, ..., y_n: Liste der Y-Werte
`rc = gset('lincolor', Index);`	Index ist wieder die Nummer, die unter gset(‚colrep', index, ‚Farbe') festgelegt wurde; Default ist 1
`rc = gset('lintype',Typ)`	Typ $\in$ [1,46]; vgl. Tabelle 16.5 in SAS/GRAPH: Reference Vol. 1, S. 429; Default ist 1
`rc=gset('linwidth',Stärke);`	Stärke der Linie in Pixel; muß größer 0 sein, Default ist 1

Text

Um Text auszugeben wird ein XY-Paar benötigt, das die Koordinaten bezeichnet, an denen der Text ausgerichtet werden soll. Wie genau er ausgerichtet wird – zentriert über diesem Punkt, rechts davon, links davon – wird mit Hilfe der Eigenschaft „texalign“ festgelegt. Daneben besitzt Text die Eigenschaften Schriftart, -farbe, und -größe.

```
rc = gdraw ('text', x, y, 'Text');
```
x_1, y_1: Textposition

```
rc = gset('texalign', 'horizontal',
     'vertikal');
```
Ausrichtung des Textes in Bezug auf x,y
horizontal: *center, left, right, normal*
vertikal: bottom, *base, half, top, normal*
Default ist horizontal = 'normal', vertikal = ‚normal'
vgl. SAS/GRAPH: Reference Vol. 1, S. 721

```
rc = gset('texcolor', Index);
```
Farbindex

```
rc = gset('texfont','Schriftart');
```
Entweder ein SAS- oder ein device-abhängiger Font

```
rc = gset('texheight', Schriftgröße);
```
Default ist der Wert der GOPTION htext= oder 1

Speichern und Benennen von DSGI-Graphiken

Die Auswahl des Arbeitskatalogs muß erfolgen, bevor DSGI durch den Aufruf von ginit() initialisiert wird. Auch hierfür wird die Funktion gset() verwendet.

Syntax:

```
rc = gset('catalog', 'libref', 'catalog');
```
libref: Name einer gültigen Library
catalog: Name eines bestehenden oder zu erstellenden Kataloges
Default-Arbeitskatalog ist WORK.GSEG

Für die DSGI-Graphik können ein Name und eine Beschreibung angegeben werden, die Benennung erfolgt mit dem Wechsel von Station WSAC zu GSOP.

Syntax:

```
rc = graph('clear', 'Name', 'Beschreibung');
```
Name: gültiger SAS-Name (8 Zeichen)
Beschreibung: bis zu 40 Zeichen
Default-Name ist DSGIn

Einfügen von GRSEG-Einträgen in DSGI-Graphiken

SAS-Vektorgraphiken werden als Katalogeinträge vom Typ GRSEG gespeichert. Um einen GRSEG-Eintrag in eine DSGI-Graphik einfügen zu können, muß er sich im gleichen Arbeitskatalog befinden. Zum Kopieren kann auf PROC CATALOG oder das Katalogfenster zurückgegriffen werden (oder SCL-Funktionen). DSGI selbst bietet hierfür leider keine Möglichkeit.
Für das Einfügen in die DSGI-Graphik ist wieder die Funktion graph() zuständig, der GRSEG-Eintrag erscheint im aktiven Viewport.

Syntax:

```
rc = graph('insert','Name');
```
Name: Name des GRSEG-Eintrages

Festlegen des Layouts und Konzeption des Programmes

Es sind mehrere Arbeitsschritte für die Erstellung der DSGI-Karte notwendig, die auch in der Konzeption voneinander getrennt werden. Die gesamte Graphik soll aus einer Karte, einer Legende, einem Verteilungsdiagramm und einer Kompaßrose bestehen.
Zunächst muß die Karte selbst berechnet und in einem Katalog abgespeichert werden. In diesen Katalog wird auch die bereits vorhandene Graphik mit der Kompaßrose kopiert. Die verwendete Kompaßrose ist ein Clipart aus Winword, das als CGM-Datei in SAS importiert wurde (sie könnte aber natürlich auch in DSGI gezeichnet werden).
Aufgabe des DSGI-Schrittes ist es, Karte und Kompaßrose in vordefinierte Bereiche einzusetzen und dann das Verteilungsdiagramm und die Legende zu zeichnen.
[Anmerkung: Selbstverständlich könnte das Makro auch so konzipiert werden, daß die Kartenerstellung ebenfalls innerhalb des Makros stattfindet. Da aber häufig die Grundkarte nur einmal erstellt werden muß, während dann das weitere Layout mehrfach angepaßt wird bis die Graphik fertig ist – Beschriftung, Schriftart etc. – habe ich diesen Weg gewählt. Es steht natürlich jedem frei, daß Makro an seine Bedürfnisse anzupassen]

Aufteilung des Graphikbereiches

Hauptelement der Graphik ist die Karte, sie nimmt den größten Teil des Graphikbereiches ein (70%). Rechts davon sollen Kompaßrose, Verteilungsdiagramm und Legende plaziert werden. An Beschriftung sind ein oder zwei Titel, sowie zwei Fußzeilen vorgesehen.

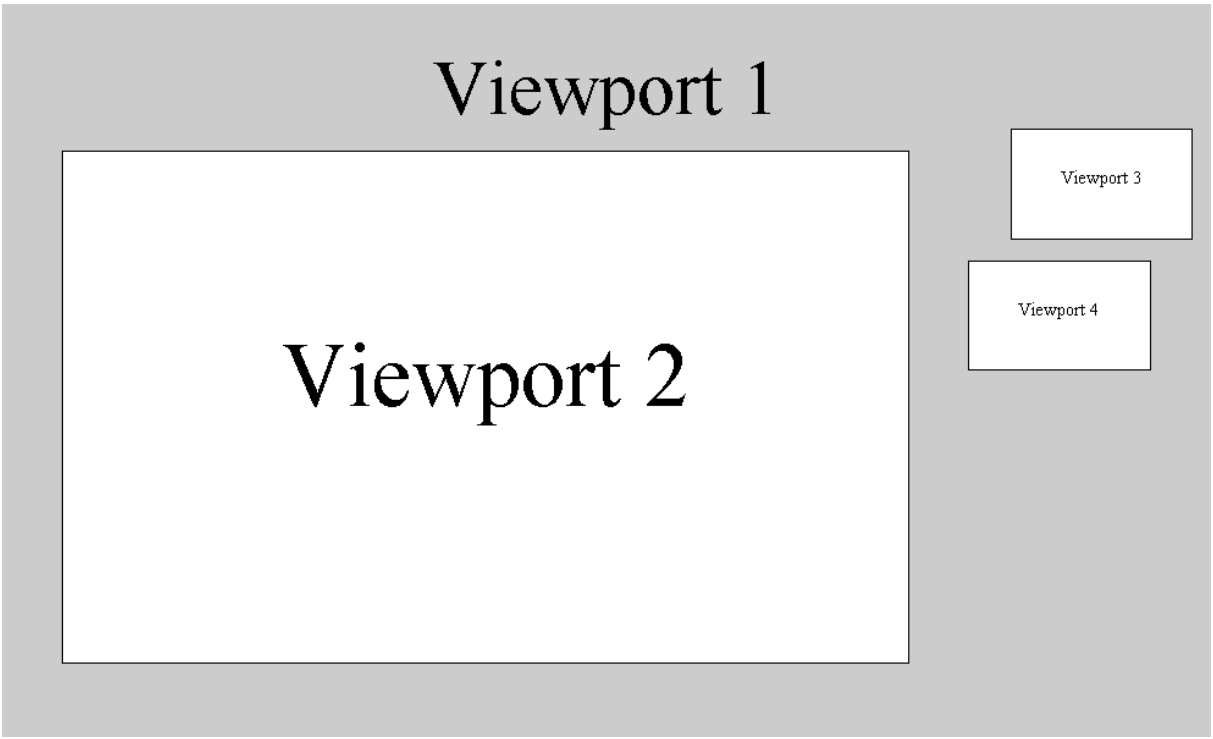

Abbildung 1: Aufteilung des Graphikbereiches

Es werden insgesamt vier Viewports definiert. Viewport 1 ist der gesamte Graphikbereich, hier wird das Window (wie übrigens alle verwendeten Windows) explizit auf den Bereich 0 bis 100 gestellt. In diesem Viewport wird der Schatten erzeugt und die Graphik beschriftet. Auch das Zeichnen der Legende und die Beschriftung des Verteilungsdiagramms wird hier vorgenommen. Viewport 2 definiert den Bereich für die Karte, er läßt 5% Abstand zum linken Rand und nimmt in Breite und Höhe 70% der gesamten Fläche ein. Viewport 3 ist so positioniert, daß die Kompaßrose zentriert über dem Verteilungsdiagramm zu liegen kommt, das in

Viewport 4 gezeichnet wird. Viewport 4 wiederum hat ebenfalls 5% Abstand zum Rand und zur Karte (ohne Schatten).

Beziehungen zwischen den einzelnen Viewports

Viewport 1 nimmt den gesamten Graphikbereich ein, d.h. die Größenangaben aller anderen Viewports beziehen sich automatisch auf ihn. Da die Position dieser Viewports bekannt ist, können alle Graphikelemente in Viewport 1 genau auf sie ausgerichtet werden. In Viewport 1 werden der Schatten für Viewport 2, die Beschriftung des Verteilungsdiagramms und die Legende gezeichnet. Dabei sollen Legende und die Beschriftung linksbündig zum Verteilungsdiagramm ausgerichtet sein. Der Schatten entspricht in der Größe genau Viewport 2, nur daß er um einen bestimmten Versatz nach rechts unten verschoben wird, die gedachte Lichtquelle befindet sich links oben.
Viewport 4 kann komplett für das Verteilungsdiagramm benutzt werden, was die Berechnung von Höhe und Breite der einzelnen Balken wesentlich erleichtert.

Konzeption des Programmes

Im Makro erfolgen die Beschriftung der Legende und das Zeichnen des Verteilungsdiagramms. Deshalb werden hier auch Informationen über den zu verwendenden Text und die entsprechenden Werte benötigt. Um den Aufwand für die Beschriftung der Legende gering zu halten, wird für die Berechnung der Ausgangskarte ein Format verwendet, dessen Name dann auch an das Makro übergeben wird. Das hat den Vorteil, daß die Beschriftung für die einzelnen Legendenitems nicht einzeln übergeben werden muß, die Anzahl der Klassen könnte ja auch variieren.
Für das Verteilungsdiagramm werden Absolutwerte und Prozent-Anteile der einzelnen Klassen benötigt, deshalb wird vor den eigentlichen DSGI-Schritt ein PROC FREQ geschaltet, das die Werte berechnet und in einen temporären Data Set schreibt. Dieser Data Set wird anschließend eingelesen und die enthaltene Information in Makrovariablen gespeichert. Im einzelnen enthält der Data Set:

1. Die Anzahl der Klassen (= Anzahl der Observations)
2. Die Beschriftung für die Legende (= formatierte Werte der Analysevariablen)
3. Die absoluten Anteile der Klassen (= Variable count)
4. Die prozentualen Anteile der Klassen (=Variable percent)

Das DSGI-Makro

Das DSGI-Makro hat eine Reihe von Parametern, ein Teil davon wird als „positional parameter“ übergeben, d.h. die Werte müssen im Aufruf in der korrekten Reihenfolge erscheinen. Die optionalen Angaben werden als „keyword parameter“ übergeben, dem eigentlichen Wert werden der Name der Makrovariable und ein „=“-Zeichen vorangestellt (s.u.).

```
/*************************************************************

 Aufgabe:
               DSGI-Layout für mit GMAP erzeugte Karten

 Autor:
               Grischa Pfister
               Abteilung für Wirtschafts- und Sozialgeographie,
               Computerkartographie und GIS
               Geographisches Institut der Ruprecht-Karls-Universität
               Heidelberg
```

```
 Anschrift:
                Im Neuenheimer Feld 348
                69120 Heidelberg
                gpfister@geog.uni-heidelberg.de
                www.rzuser.uni-heidelberg.de/~fg2

 Variablen:

    DATA:       Data Set
    VAR:        Variable mit darzustellenden Anteilen
    FORMAT:     Format, das für die Klassifizierung verwendet wird
    LIBCAT:     Library & Katalog in dem sich die Graphiken befinden;
                Dieser Katalog wird von DSGI als Arbeitskatalog benutzt,
                d.h. die Kompaßrose und Ausgangskarte müssen hier gespeichert
                sein (Angabe MUSS zweigliedrig sein in der Form LIB.CAT)
    MAP:        Ausgangskarte, die eingefügt werden soll
    TITLE1:     erste Titelzeile
    TITLE2:     zweite Titelzeile

*****************************************************************/


%MACRO dsgimap(data, var, format, libcat, map, title1, title2=none, foot1=Quelle:
Amt für Stadtentwicklung und Statistik, foot2=Bearbeitung: Geographisches Institut
Heidelberg);


/*****************************************************************
 BERECHNUNG DER KENNWERTE
*****************************************************************/


/* Berechnung mit PROC FREQ */

PROC FREQ data=&data noprint;
   FORMAT &var &format;
   TABLE  &var / out=tmp;
run;


/* Speicherung in Makro-Variablen */

DATA _null_;

   SET tmp (where = (percent ne .)) end = eof;

   call symput('a'||trim(left(putn(_n_,'best.'))),
               putn(count,'best.'));

   call symput('p'||trim(left(putn(_n_,'best.'))),
                putn(percent,'best.'));

   call symput('class'||trim(left(putn(_n_,'best.'))),
               trim(left(putn(&var,"&format"))));
```

```
   total + count;

   if eof then do;

      call symput('total', putn(total,'best.'));
      call symput('nclass',putn(_n_,  'best.'));
   end;

run;


/*****************************************************************
 BEGINN DES DSGI-DATA STEPS
*****************************************************************/

DATA _null_;

/* STANDARDS */

   bs   = 1.3;                  * Breite für den Schatten ;
   font = 'hwdmx020';           * Schriftart ;
   h_t1 = 5;                    * Schriftgröße 1. Titel ;
   h_t2 = 4;                    * Schriftgröße 2. Titel ;
   h_f  = 3;                    * Schriftgröße Fußnoten ;
   h_l  = 2.7;                  * Schriftgröße für Legende u. Verteilung ;



/* Hintergrundfarbe */

   rc = gset('cback','cxCCCCCC');


/* Festlegen des Arbeitskataloges */

   lib = scan("&libcat",1,'.');
   cat = scan("&libcat",2,'.');

   rc  = gset('catalog',lib,cat);


/*****************************************************************
 GKCL -> WSAC
 DEFINITION DER GRAPHIKAUSSCHNITTE
*****************************************************************/

   rc = ginit();


/* VP1 - Viewport für die gesamte Graphik */

      rc = gset('viewport',1, 0,0,1,1);
      rc = gset('window',  1, 0,0,100,100);
```

```
/* VP2 - Viewport für die Ausgangskarte */

      vp2_x1 = .05;
      vp2_x2 = .75;

      if upcase("&title2") ne 'NONE' and "&title2" ne '' then vp2_y1 = .1;
      else vp2_y1 = .15;

      vp2_y2 = vp2_y1 + .7;

      rc = gset('viewport',2, vp2_x1,vp2_y1,vp2_x2,vp2_y2);
      rc = gset('window',  2, 0,0,100,100);


/* VP3 - Viewport für die Kompaßrose
         Kompaßrose zentriert über Verteilungsdiagramm */

      rc = gset('viewport',3, .835,.68,.985,.83);
      rc = gset('window',  3, 0,0,100,100);


/* VP4 - Viewport für das Verteilungsdiagramm */

      rc = gset('viewport',4, .8,.5,.95,.65);
      rc = gset('window',  4, 0,0,100,100);


/*****************************************************************
 WSAC -> GSOP
 WEITERE DEFINITIONEN UND GRAPHIKERSTELLUNG
*****************************************************************/

      rc = graph('clear',"&map.1","&title1");   * Benennung folgt den SAS-
                                                  Konventionen ;


/* Erstellung der Farbliste
   WICHTIG: die hier verwendeten Farben müssen mit denen der Karte
            übereinstimmen. Werden mehr als 4 Klassen verwendet, muß
            die Liste entsprechend erweitert werden !!! */

         rc = gset('colrep', 1,'black');
         rc = gset('colrep', 2,'white');
         rc = gset('colrep', 3,'cxCCCCCC'); * Graustufe Hintergrund;
         rc = gset('colrep', 4,'cx999999'); * Graustufe Schatten;
         rc = gset('colrep',11,'cxffe100'); * Klasse 1;
         rc = gset('colrep',12,'cxff9b00'); * Klasse 2;
         rc = gset('colrep',13,'cxcd3737'); * Klasse 3;
         rc = gset('colrep',14,'cx9b0032'); * Klasse 4;
*         rc = gset('colrep',15,'');        * Klasse ...;



         rc = gset('transno',1);
```

```
/* Hintergrund:
   ein graues Rechteck bildet den Hintergrund für die Gesamt-
   graphik */

         rc = gset('filtype','solid');
         rc = gset('filcolor',3);
         rc = gdraw('bar',0,0,100,100);

/* Schatten:
   ein um "bs" nach links und unten versetztes Rechteck gleicher
   Größe hinter VP 2 */

         rc = gset('filcolor',4);
         rc = gdraw('bar',vp2_x1 * 100 + bs,       vp2_y1 * 100 - bs,
                          vp2_x2 * 100 + bs * 3/4, vp2_y2 * 100 - bs);

/* ANMERKUNG:
   Hier wird von einer Graphik im Querformat ausgegangen, deshalb wird
   der Versatz in X-Richtung mit dem Faktor 3/4 multipliziert. Die tat-
   sächliche Breite des Versatzes ist von der Seitengröße abhängig. Da
   X- und Y-Größe in der Regel nicht gleich sind (z.B. DIN A 4 29.7 cm *
   21 cm) würde auch der Schatten unterschiedlich breit sein - ein Effekt
   der nicht gewünscht ist, deshalb diese Korrektur. Für Graphiken im
   Hochformat müßte stattdessen y1_vp2 korrigiert werden (- bs * 3/4) */


/*****************************************************************
 EINSETZEN DER KARTE IN VP2
*****************************************************************/

        rc = gset('transno',2);


/* Weißes Rechteck als Hintergrund */

        rc = gset('filtype','solid');
        rc = gset('filcolor',2);
        rc = gdraw('bar',0,0,100,100);


/* Rahmenlinie */

        rc = gset('lincolor',1);
        rc = gset('linwidth',1.5);
        rc = gdraw('line',5,
                    0,100,100,  0,0,
                    0,  0,100,100,0);


/* Einsetzen der Karte */

        rc = graph('insert',"&map");
```

```
/*****************************************************************
 EINSETZEN DER KOMPASSROSE IN VP3
*****************************************************************/

        rc = gset('transno',3);

        if cexist(trim(left(lib))||'.'||trim(left(cat))||".compass.grseg") then
        rc = graph('insert','compass');

/* Die SCL-Funktion cexist('lib.cat.entry.type') testet, ob ein Katalogeintrag
   existiert und gibt im Erfolgsfalle einen Wert > 0 zurück */

/*****************************************************************
 ZEICHNEN DES VERTEILUNGSDIAGRAMMS IN VP4
*****************************************************************/

/* Die Zeichnung des Verteilungsdiagramms erfolgt in VP4, der zu 100%
   für die Balken benutzt werden kann. Die Höhe der Balken entspricht dem
   prozentualen Anteil, die Breite ist gleich 100 geteilt durch die
   Anzahl der Klassen.
   Die Beschriftung erfolgt dann in VP 1 */

        rc = gset('transno',4);

        rc = gset('linwidth',1);

        breite = 100 / &nclass;

        do i = 1 to &nclass;

           x1 = breite * (i - 1);
           x2 = breite * i;
           y1 = 0;
           y2 = input(symget('p'||trim(left(putn(i,'best.')))),best.);

           rc = gset('filcolor',i + 10);

           rc = gdraw('bar',x1,y1,x2,y2);  * Balken ;

           rc = gdraw('line',5,
                    x1,x2,x2,x1,x1,
                    y1,y1,y2,y2,y1);       * schwarze Außenlinie ;
        end;


/* Beschriftung des Verteilungsdiagramms in VP1 */

        rc = gset('transno',1);

        rc = gset('texcolor',1);
        rc = gset('texfont',font);
        rc = gset('texheight',h_1);
        rc = gset('texalign','center','normal');
```

```
        xpos  = 15 / &nclass;     * 15 ist die Breite von Viewport 4 ;
        x1pos = 80 + xpos / 2;    * 80 ist X1 von Viewport 4;

        do i = 1 to &nclass;

           x1 = x1pos + xpos * (i - 1);  * X-Wert für zentrierte Beschriftung ;
           y1 = 47;                      * 50 ist Y1 von Viewport 4 ;

           txt = trim(left(symget("a"||trim(left(putn(i,'best.'))))));

           rc  = gdraw('text',x1,y1,txt);
        end;

        rc = gset('texalign','left','normal');

        rc = gdraw('text',80,43,'Fälle je Klasse');
        rc = gdraw('text',80,40,'Insgesamt: '||left(trim("&total")));


/*****************************************************************
 ZEICHNEN DER LEGENDE
*****************************************************************/

/* Koordinaten für das erste Kästchen der Legende:
   Breite = Höhe = 2%;
   Der Ausgangswert für y1 wird an vp2_y1 ausgerichtet, x1 ist wieder 80,
   d.h. Legende und Verteilungsdiagramm werden linksbündig ausgerichtet */

        x1 = 80;
        y1 = vp2_y1 * 100 + 18;
        x2 = x1 + 2;
        y2 = y1 + 2;


/* Legendentext hat 1% Abstand zu Kästchen */

        txt_x1 = x2 + 1;

        rc = gset('texalign','left','base');

        do i = 1 to &nclass;

           txt = trim(left(symget("class"||trim(left(putn(i,'best.'))))));

           rc  = gset('filcolor',i + 10);    * Farbe für Klasse i ;

           rc  = gdraw('bar',x1,y1,x2,y2);
           rc  = gdraw('line',5,
                     x1,x2,x2,x1,x1,
                     y1,y1,y2,y2,y1);

           rc  = gdraw('text',txt_x1,y1,txt);

           y1 + (-3);   * Y-Werte werden für nächsten Legendenitem ;
           y2 + (-3);   * eingestellt: 1% Abstand + 2% Höhe = 3%   ;
```

```
        end;


/*****************************************************************
 BESCHRIFTUNG
*****************************************************************/

        rc = gset('transno',1);

        rc = gset('texalign','center','normal');

/* Titel 1 */

        rc = gset('texheight',h_t1);
        rc = gdraw('text',50,90,trim(left("&title1")));


/* Titel 2 */

        if upcase("&title2") ne 'NONE' and "&title2" ne '' then do;
           rc = gset('texheight',h_t2);
           rc = gdraw('text',50,85,trim(left("&title2")));
        end;


/* Fußnoten */

        rc = gset('texheight',h_f);
        rc = gdraw('text',50,vp2_y1 * 20 + .5 + 3,"&foot1");
        rc = gdraw('text',50,vp2_y1 * 20 + .5,    "&foot2");



/*****************************************************************
 GSOP -> WSAC
 Speichern und Anzeige der Graphik
*****************************************************************/

        rc = graph('update');


/*****************************************************************
 WSAC -> GKCL
 Beenden von DSGI
*****************************************************************/

        rc = gterm();

run;

/*****************************************************************
 LÖSCHEN DER MAKROVARIABLEN
*****************************************************************/
```

```
DATA _null_;

   do i = 1 to &nclass;

      call symput('a'    ||trim(left(putn(i,'best.'))),'');
      call symput('class'||trim(left(putn(i,'best.'))),'');
      call symput('p'    ||trim(left(putn(i,'best.'))),'');
   end;

   call symput('nclass','');
   call symput('total','');

run;

%mend dsgimap;
```

Beispiel – Eine Weltkarte mit dem Makro „dsgimap“

1. Schritt - Erstellen der Karte

Aus Gründen der Nachvollziehbarkeit habe ich hier ein anderes Beispiel gewählt, als auf dem KSFE-Plakat dargestellt. Statt den Wahlergebnissen für die Bundestagswahl 1998 wird eine Übersichtkarte für die Welt dargestellt, der Koordinaten-Data Set ist im Lieferumfang von SAS/GRAPH enthalten, so daß jeder das Beispiel nachrechnen können sollte.
Zunächst werden alle bereits geänderten Graphikoptionen zurückgesetzt. Dann wird ein „Dummy“-Data Set erzeugt, er soll die IDs der einzelnen Länder, sowie eine Variable mit den Kontinent-Nummern enthalten. Zu diesem Zweck wird der Data Set, der die Koordinatengrundlage enthält, eingelesen, wobei nur die beiden benötigten Variablen behalten werden. Die Variable „CONT“ wird in „VAR01“ umbenannt, um beim Makroaufruf eine Verwechslung mit dem Format „CONT.“ zu vermeiden. Die Kontinentnummer wird dann mit Hilfe eines Formates klassifiziert, und es wird eine Beschriftung festgelegt, die in der Legende Verwendung finden wird.

```
GOPTIONS
reset = global
reset = all
;

PATTERN1 v=s c=cxffcd00;
PATTERN2 v=s c=cxff9b00;
PATTERN3 v=s c=cxcd3737;
PATTERN4 v=s c=cx9b0032;

DATA dummy;
   SET maps.worldprj (keep=cont id);
   rename cont = var01;

PROC SORT nodups data=dummy;
   BY var01;

PROC FORMAT;
   VALUE cont
   91,92 = 'Amerika'
   93,95 = 'Europa & Asien'
```

```
   94    = 'Afrika'
   96    = 'Australien'
;

PROC GMAP data = dummy map = maps.worldprj gout=work.test;
   FORMAT var01 cont.;
   ID id;
   CHORO var01 /
           discrete
           coutline=black
           nolegend
           name='world'
   ;
run;
quit;
```

Die Graphik wird im temporären Katalog WORK.TEST als „WORLD.GRSEG“ gespeichert. Dieser Katalog wird als Arbeitskatalog für DSGI definiert und hierhin muß auch die Graphik mit der Kompaßrose kopiert werden.

2. Schritt - Aufruf des Makros

Jetzt kann das DSGI-Makro aufgerufen werden (nachdem es kompiliert worden ist). Die benötigten Parameter sind:

- Name des Data Sets
- Name der dargestellten Variablen
- Name des Formates für die Klassenbildung
- Name des Arbeitskataloges
- Name der Ausgangskarte, die eingesetzt werden soll
- Titel
- Titel (wenn gewünscht)

```
%dsgimap(dummy,var01,cont.,work.test,world,Weltkarte mit Makro ""dsgimap"",
        foot1=Quelle: SAS-Institute,foot2=Kartographie: Grischa Pfister)
```

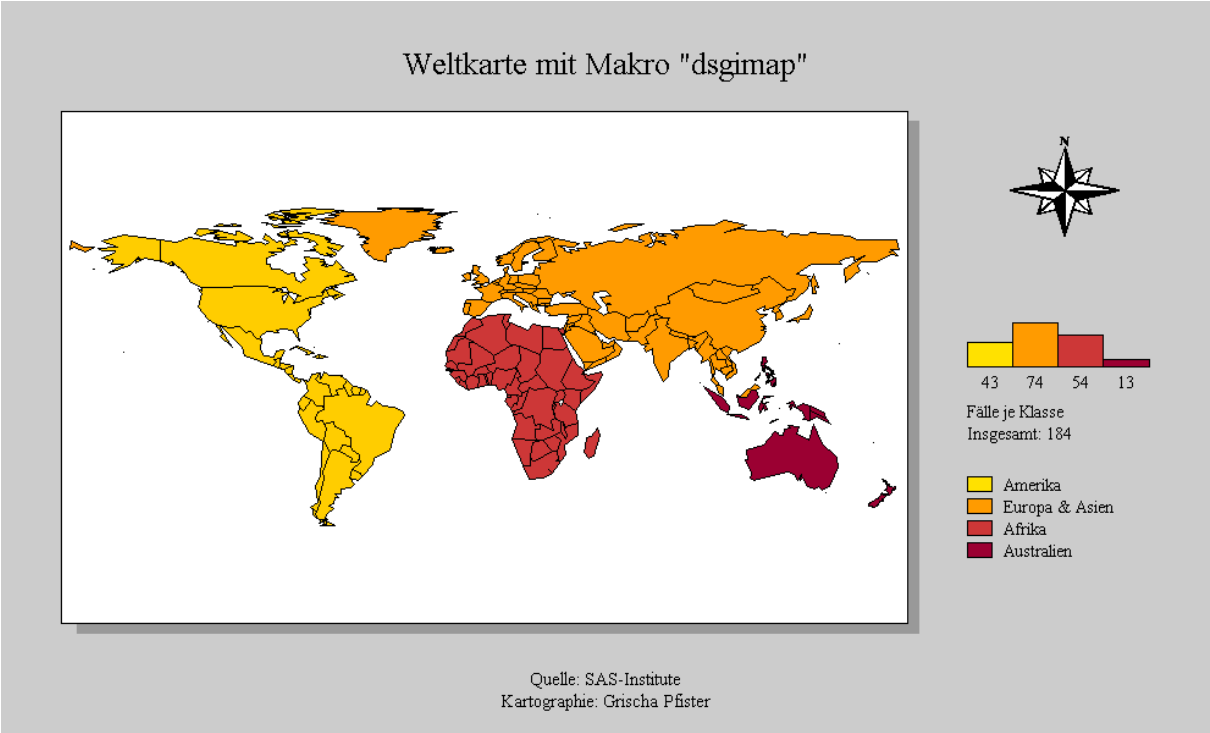

Abbildung 2: Weltkarte mit Makro „dsgimap“

Für das Ausstellungsplakat wurden nacheinander sechs unterschiedliche Karten erstellt und dann in eine Gesamtgraphik zusammengefaßt. Auch hierfür wurde DSGI verwendet, möglich ist das aber auch mit Hilfe von PROC GREPLAY und entsprechenden Templates. Der Vorteil von DSGI ist jedoch, daß die Gesamtgraphik direkt beschriftet werden kann.

Probleme bei der Erstellung von Karten mit %dsgimap

- Soll in den Titeln Text in Anführungszeichen gesetzt werden, müssen entweder einfache Anführungszeichen verwendet werden, oder jeweils zwei doppelte (vgl. Beispiel).
- Es kann vorkommen, daß Text, der in SAS dargestellt wird, in der Druckausgabe fehlt – das ist besonders dann ärgerlich, wenn die Ausgabe in großen Formaten und mit entsprechenden Kosten geschieht. Es ist deshalb auf alle Fälle sinnvoll, die Ausgabe in eine Postscript-Datei umzuleiten und in einem entsprechenden Viewer (z.B. Ghostview) anzusehen, bevor die Datei an einen Drucker geschickt wird. Meiner Erfahrung nach ist die Fehlerquelle meist die Schriftgröße, d.h. der Text wird – aus welchen Gründen auch immer – zu lang und fehlt deshalb in der Druckausgabe.
- Das Makro verwendet einen Windows True Type Font statt einer SAS-Schriftart – unter UNIX sind diese Schriftarten nicht verfügbar, dann sollte Zapf o.ä. verwendet werden.

- Probleme können unter Windows auch dann auftauchen, wenn die Devices WIN und WINPRTC unterschiedliche Schriftartenlisten haben. Dies sollte im Zweifelsfalle mit Hilfe von PROC GDEVICE überprüft werden:

```
PROC GDEVICE c=gdevice0.devices nofs;
   LIST win;
   LIST winprtc;
run; quit;
```

- Ausgangskarte und DSGI-Karte sollten mit gleichen Einstellungen für HSIZE/VSIZE erstellt werden, um Verzerrungen zu vermeiden, die entstehen, wenn die Graphiken mit unterschiedlichen Seitenverhältnissen erzeugt werden (z.B. Bildschirmgröße im Vergleich zu druckbarem Bereich des Druckers). Am besten werden sowohl Ausgangskarte als auch die engültigen Graphiken gleich mit den korrekten Einstellungen des Ziel-Devices erzeugt. Der für den Device verfügbare Bereich kann mit Hilfe der Prozedur GTESTIT abgefragt werden.
- Soll gedruckt werden, muß zunächst über das Pulldown-Menü der entsprechende Drucker gewählt und in einem GOPTIONS-Statement WINPRTC als Target-Device angegeben werden, dann kann die Karte erzeugt, und das Makro aufgerufen werden.
- Das Makro ist auf vier Klassen ausgelegt, sollen mehr verwendet werden, muß die Farbliste entsprechend angepaßt werden. Bei Verwendung von mehr als sechs Klassen wird die Legende über Viewport 2 hinausragen, dann sollten die Ausgangskoordinaten für Viewport 4, Beschriftung der Verteilung und Legende entsprechend nach oben verschoben werden.

Ich wünsche allen, die sich auf DSGI einlassen, viel Spaß beim Ausprobieren. Sicherlich ist es zunächst einmal etwas komplizierter als ANNOTATE, aber ich denke, die Ergebnisse zeigen, daß sich die Einarbeitung lohnt.
Das Makro wird im "SAS-Anwenderhandbuch im Netz" (http://www.rzuser.uni-heidelberg.de/~x16/sas-ah.html) veröffentlicht werden.

Für Fragen und Anregungen stehe ich gerne zur Verfügung unter
gpfister@geog.uni-heidelberg.de

Literatur

SAS Institute Inc. (1990): SAS/GRAPH Software: Reference, Version 6, First Edition, Volume 1, Cary, NC.

SAS Institute Inc. (1993): SAS/GRAPH Software: Examples, Version 6, First Edition, Cary, NC.