Programmierung anpassungsfähiger Makros durch Datensatzzerlegung am Beispiel eines erweiterten Bubble-Plots

Andreas Deckert
Institute of Public Health
INF 324
Heidelberg
a.deckert@uni-heidelberg.de

Zusammenfassung

Anhand eines konkreten Beispiels wird die Zerlegung und Aufbereitung eines Datensatzes für die Erstellung flexibler Makros aufgezeigt. SAS 9.2 bietet die Möglichkeit, Bubble-Plots über die BUBBLE-Anweisung in der Prozedur GPLOT zu realisieren, unterteilt in höchstens 2 Untergruppen. Verwendet man hier die FILL-Option, kann es passieren, dass ein Bubble der einen Gruppe ein kleineres Bubble in derselben Kategorien-Kombination der anderen Gruppe vollkommen überdeckt. Ausgehend von diesen Beschränkungen wurde vom Autor ein Makro entworfen, das beliebig viele Untergruppen zulässt und verschiedene Optionen für die Darstellung bei der Besetzung einer Kreuzkategorie durch mehrere Bubbles bietet. Dazu wird der Datensatz über Arrays in Diagonal-Matrizen überführt womit dann jeder Datenpunkt über singuläre SCATTER-Anweisungen (bzw. BUBBLE-Anweisungen in SAS 9.3) innerhalb PROC SGPLOT darstellbar ist. Um auch die in der Regel Datensatz-übergreifend festgelegten Optionen der Anweisungen innerhalb von PROC SGPLOT individuellen Bedingungen anzupassen, werden diese ebenfalls durch die Einträge der Datenmatrix gesteuert, was allerdings nur durch zusätzliche Makro-Variablen möglich ist. Hinweis: SAS 9.3 bietet nun erstmals die Möglichkeit, die BUBBLE-Anweisung auch innerhalb der Prozedur SGPLOT zu verwenden. Damit können einige aber nicht alle Teile der von dem Makro abgedeckten Funktionalitäten realisiert werden.

Schlüsselwörter: Makro-Programmierung, Makro-Variable, Bubble-Plot, SGPLOT-Prozedur, SCATTER-Anweisung, BUBBLE-Anweisung, Array

1 Einführung: Bubble Plots

Mit Bubble-Plots kann man (aggregierte) dreidimensionale Daten grafisch in zweidimensionaler Form darstellen. Die dritte Dimension beeinflusst dabei die Größe von Kreisen (daher der Name Bubble-Plot) die an den zweidimensionalen Koordinaten der ersten beiden Variablen erscheinen. Hierbei ist dann die Fläche oder der Radius des Kreises (A) proportional zur Häufigkeit einer aufgetretenen zweidimensionalen Kategorien-Kombination oder einfach (B) proportional zum Wert der dritten Variablen. Als Beispiel für (A) sei ein Fragebogen genannt. Dieser besteht aus mehreren kurzen Fragen, zu denen es immer die gleichen Antwort-Kategorien zum Ankreuzen gibt (siehe Abbildung 1). Der Fragebogen wird nun ausgewertet, indem über alle befragten Personen die Häufigkeit von Kombinationen der jeweiligen Frage mit den Antwort-Katego-

rien ermittelt wird. Diese Häufigkeiten lassen sich dann mithilfe eines Bubble-Plots darstellen (siehe Abbildung 2).

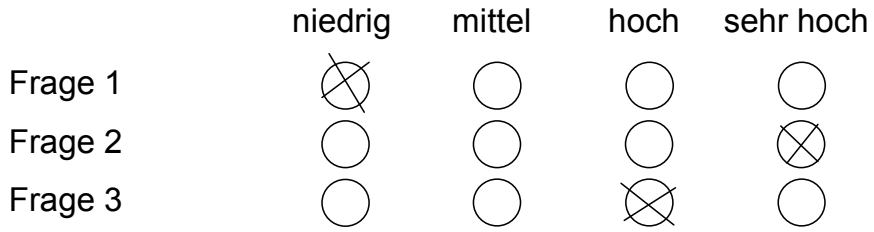

Abbildung 1: Aufbau eines Fragebogens

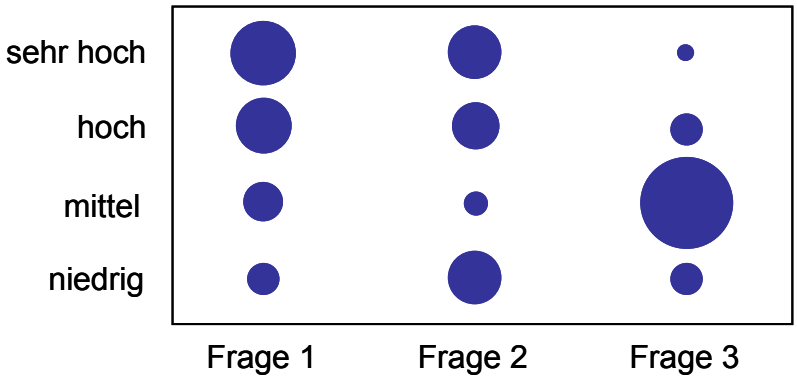

Abbildung 2: Grafische Darstellung der Auswertung des Fragebogens

Ein Beispiel für (B) ist die Darstellung der Lebenserwartung und der Kosten des Gesundheitswesens mit gleichzeitiger Visualisierung der Bevölkerungsgröße. Hier wird nicht die Häufigkeit von besetzten Kategorien-Kombinationen zweier Variablen dargestellt, sondern der unabhängige Wert der dritten Variable bestimmt die Größe des Bubbles.

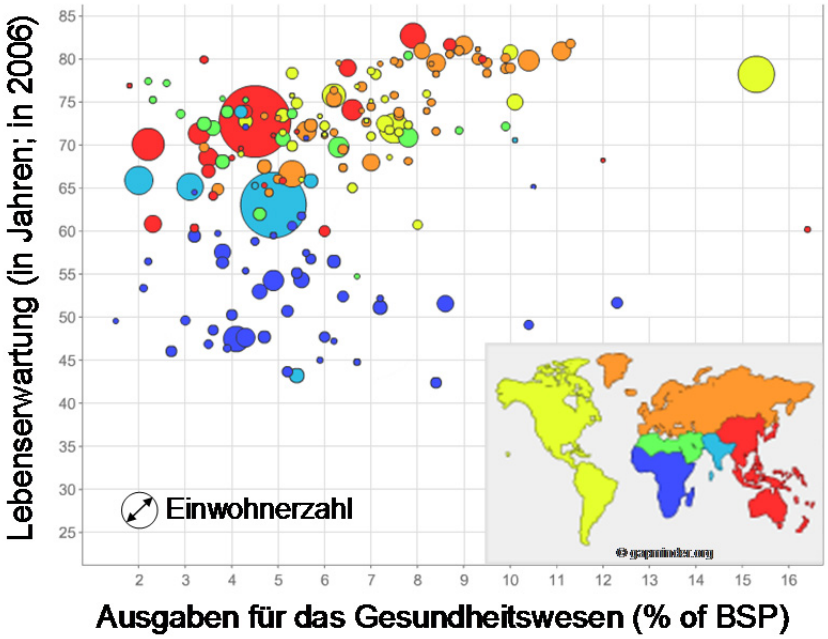

Abbildung 3: Bubble-Plot-Darstellung von Lebenserwartung und Kosten des Gesundheitswesens unter Berücksichtigung der Bevölkerungszahl

1.1 Standardlösung in SAS 9.2

In SAS 9.2 kann man einfache Bubbles-Plots mit dem **BUBBLE**-Statement in **PROC GPLOT** erzeugen.

```
proc gplot data=final;
    bubble cat2*cat1 = count /  bcolor=blue bscale=area
                                bfill=solid bsize=10 ...;
run;
```

Das **BUBBLE2**-Statement erlaubt die Einbeziehung von Untergruppen (weitere Aufteilung der Daten z.B. nach Geschlecht), die Anzahl der Untergruppen ist jedoch auf zwei begrenzt. Ein weiterer Nachteil ist, dass bei gefüllten Bubbles ein kleineres Bubble einer Untergruppe durch ein größeres Bubble komplett verdeckt werden kann (siehe Abbildung 4). Zudem müssen Daten, die dem ersten Beispiel (Fragebogen) entsprechen, vorher durch **PROC FREQ** aufbereitet werden.

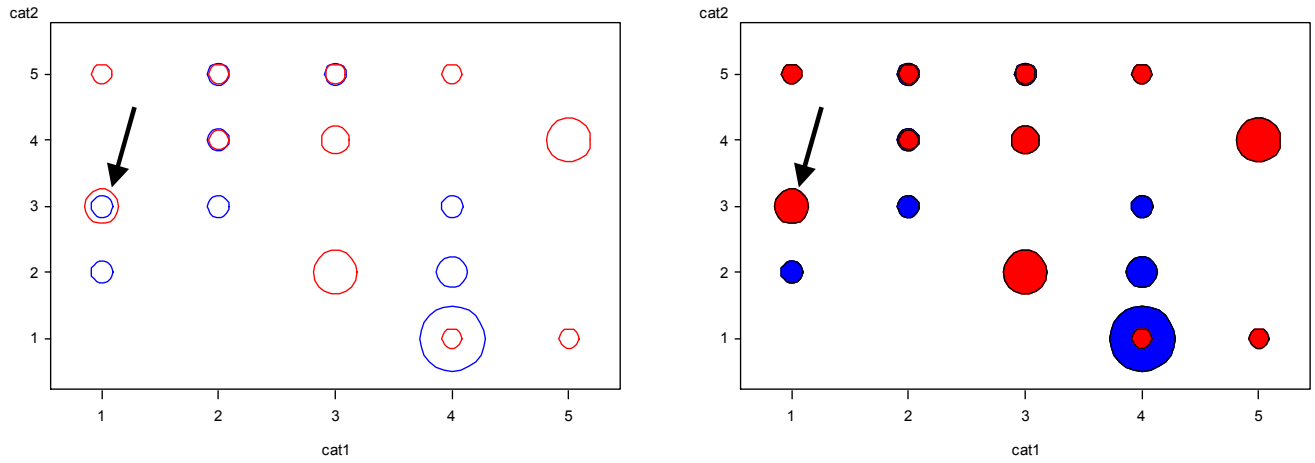

Abbildung 4: Ergebnis von **PROC GPLOT** mit **BUBBLE2**-Statement, einmal ohne und einmal mit Füllung

1.2 Lösung in SAS 9.3

In SAS 9.3 besteht die Möglichkeit, in **PROC SGPLOT** ein **BUBBLE**-Statement zu verwenden:

```
proc sgplot data=final;
    bubble x=var1 y=var2 size=count/ group=group_var  fill ...;
run;
```

Hier kann nun in den Attributen des Statements eine Gruppierungsvariable übergeben werden, die beliebig viele Untergruppen zulässt. Außerdem stehen viele weitere Optionen zur Gestaltung der Bubbles zur Verfügung. Aber auch hier muss der Datensatz im Vorfeld aufbereitet werden (falls er Beispiel (A) entspricht). Außerdem lassen sich gleich große Bubbles in derselben Kategorien-Kombination nur schwer unterscheiden.

1.3 Alternativen

Bei **PROC GPLOT** besteht noch die Möglichkeit, anstatt des **BUBBLE**-Statements ein **SCATTER**-Statement zu verwenden und den Kreis über ein vordefiniertes **SYMBOL** zuzuweisen. Hier müsste allerdings für jedes Bubble ein eigenes **SYMBOL** definiert werden, da die Kreisgröße pro **SYMBOL** für die damit abzubildenden Daten festgelegt ist.
Eine weitere Möglichkeit besteht darin, ein eigenes Makro zu entwickeln. Obwohl SAS 9.3 mit dem **BUBBLE**-Statement in **PROC SGPLOT** für die meisten Anwendungen mittlerweile genügend Funktionsumfang bietet, soll das Makro hier verwendet werden, um die Zerlegung von Datensätzen für flexible Makro-Programmierung beispielhaft aufzuzeigen.

2 Ideen und Bausteine

Das Makro soll zunächst folgende Funktionen erfüllen:

- Darstellung von Bubble-Plots mit beliebig vielen Untergruppen
- Bei Überlagerungen sollen kleine Bubbles immer im Vordergrund dargestellt werden
- Es soll eine Lösung für die Überlagerung gleich großer Bubbles bieten
- Es soll **PROC SGPLOT** für hochauflösende Grafiken verwendet werden
- In SAS 9.2 ist das **BUBBLE**-Statement in **PROC SGPLOT** noch nicht verfügbar, daher soll stattdessen das **SCATTER**-Statement verwendet werden. Bubbles können mit dem **SCATTER**-Statement wie folgt erzeugt werden:

```
proc sgplot data=_plot_;
    scatter x=xvar y=yvar / markerattrs=(size=count
                            color=blue symbol=circle);
run;
```

- Die Daten sollen vorher nicht mit **PROC FREQ** aufbereitet werden müssen.
- Die Kategorien sollen nicht auf ganze Zahlen beschränkt sein.
- Es sollen verschiedene Formen von Datensätzen verarbeitet werden können.

2.1 Verarbeitbare Datensätze

(A) Datensatz mit dreidimensionalen Einzelbeobachtungen:

ID	Cat1	Cat2	Level (group)
1	2	4	Angestellter
2	2	4	Angestellter
3	3	1	Arbeitslos
4	2	4	Selbstständig

Abbildung 5: dreidimensionale Einzelbeobachtungen

(B) Eine Untergruppe pro Zeile (aggregiert)

Country	Var1	Var2	Size
Australia	2.3	4.5	6
Germany	2.7	4	5.2
Russia	3.4	1.9	14
Zambia	2	4	11.9

Abbildung 6: aggregierte Daten

3 Aufbau des Makros

Im Folgenden werden nun die wichtigsten Elemente des Makros und die Überführung des Datensatzes in eine Diagonalmatrix beschrieben. Die einzelnen Schritte beziehen sich dabei im Wesentlichen auf Datensatzstrukturen in Anlehnung an Beispiel (A). Sämtliche Code-Fragmente befinden sich innerhalb einer Makro-Umgebung.

3.1 Schritt 1: Aufbereitung der Tabelle

Die notwendigen Tabellen und Elemente zur Steuerung des Makros werden in der üblichen Weise über die Makro-Schnittstelle übertragen:

```
%bubble(TabIn=test,cat1=var1,cat2=var2,group=country,...);
```

Innerhalb des Makros wird dann auf diese Makro-Variablen zugegriffen. Als erster wesentlicher Schritt (nach Plausibilitätsprüfungen zu den Übergabeparametern) müssen die Häufigkeiten der einzelnen Kategorien-Kombinationen ermittelt werden. Dazu wird zunächst Szenario (A) über eine Makrovariable abgefragt. Sobald mehrere Gruppen vorhanden sind, wird **PROC FREQ** mit einem **BY**-Statement ausgeführt. Man sollte sich hier der Tatsache bewusst sein, dass alle Makro-Variablen und -Anweisungen vor der Abarbeitung des eigentlichen Codes aufgelöst und durch die entsprechenden Einträge aus der Symbol-Tabelle ersetzt werden.

```
%if &scenario. = "individuell" %then %do;
    proc freq data = &TabIn.;
        %if "&group." ne "" %then %do;
             by &group.;
        %end;
        tables &cat1.*&cat2./out=_final;
    run;
%end;
```

Der obenstehende Code könnte dann nach der Auflösung der Makro-Anweisungen und Variablen z.B. folgendermaßen aussehen:

```
proc freq data = Test
    by Gruppe
    tables Kategorie1* Kategorie1 /out=_final;
run;
```

Erst jetzt wird der Code sequentiell abgearbeitet und ausgeführt. Die Ergebnistabelle enthält nun alle *N* Kategorienkombinationen und die Variable *count* für die Häufigkeit der jeweiligen Kategorien-Kombination.

3.2 Schritt 2: Zählen der Untergruppen

Als nächstes muss ermittelt werden, wie viele Untergruppen in der Datenstruktur enthalten sind. Dazu wird zunächst über eine IF-Abfrage festgestellt, ob die Gruppierungsvariable leer ist (d.h. es gibt keine Untergruppen). Ist dies nicht der Fall, wird der Datensatz entsprechend der Gruppierungsvariablen sortiert. In einem Datenschritt wird dann ein Zähler eingesetzt, der sich jeweils mit Beginn einer neuen Gruppe in der sequentiellen Reihenfolge der Daten um eins erhöht. Wenn das Ende des Datensatzes erreicht ist (Abfrage über END=eof hinter der SET-Anweisung), wird zum einen die Anzahl der Untergruppen über die Zählvariable und zum anderen die Gesamtzahl *N* der Kategorienkombinationen an Makrovariablen übergeben. Eine Übergabe muss hier entweder mit einer **PROC SQL**-Anweisung oder mit **CALL SYMPUT** erfolgen.

```
%if "&group." ne "" %then %do;
    proc sort data = &TabIn.; by &group.; run;
    data &TabIn.; set &TabIn. END = eof;
        by &group.;
            if first.&group. then _i + 1;
            if eof then call symput ('last',_N_);
            if eof then call symput ('n_groups',_i);
   run;
%end;
```

In der sortierten Tabelle werden somit die letzten Einträge zur Anzahl der Gruppen und zur Anzahl der Kategorienkombinationen an die Makrovariablen übergeben:

N	Gruppe	Kategorie1	Kategorie2	count (*size*)	i
1	Angestellter	4	2	1	1
2	Angestellter	5	3	5	1
3	Angestellter	1	3	2	1
4	Arbeitslos	1	3	3	2
…	…	…	…	…	…
21	Selbstständig	5	2	4	3

Abbildung 7: Tabelle mit der Häufigkeit der einzelnen Kategorienkombinationen und der Anzahl der Gruppen

3.3 Sortieren der Tabelle

Damit die kleineren Bubbles nicht von größeren (in derselben Kategorien-Kombination) überdeckt werden, ist nun noch ein Sortieren der Ergebnistabelle nach Häufigkeit der Kategorien-Kombination und Gruppen notwendig:

```
proc sort data=_final; by descending count &group.; run;
```

Die Ergebnistabelle sieht dann folgendermaßen aus:

N	Gruppe	Kategorie1	Kategorie2	count (*size*)	i
9	Angestellter	3	2	5	1
4	Arbeitslos	1	3	3	2
3	Angestellter	1	3	2	1
11	Selbstständig	1	3	2	3
13	Angestellter	4	2	1	1
1	Arbeitslos	4	2	1	2

Abbildung 7: Ergebnistabelle nach Zusammenfassen und Sortieren der Daten

Wenn die Tabelle nun sequentiell abgearbeitet wird, werden so die großen Bubbles zuerst gezeichnet und dann im Vordergrund davor die kleineren.

3.4 Kernstück des Makros

Der Kern des Makros soll wie schon angedeutet aus einer **PROC SGPLOT**-Anweisung bestehen. Die Darstellungs-Attribute des **SCATTER**-Statements sind allerdings Datensatz-übergreifend festgelegt. D.h. wenn die kompletten Kategorienvariable als X und Y übergeben werden, werden zwar an den Koordinatenpunkten die Kreise gezeichnet, allerdings alle mit einer festen Größe. Daher sollen nun innerhalb der **PROC SGPLOT**-Anweisung über Makro-Variablen und -Anweisungen für jede Zeile aus der Ergebnistabelle einzelne **SCATTER**-Statements während der Laufzeit erzeugt werden, d.h. jedes einzelne Bubble erhält ein eigenes Statement und somit kann jedem Bubble eine individuelle Größe zugeordnet werden. Dadurch wird das Makro sehr flexibel. Der entsprechende Code des Kernstücks sieht folgendermaßen aus:

```
proc sgplot data=_plot_
    ...
    %do i=1 %to &last.
        SCATTER x=_x&i. y=_y&i./
            MARKERATTRS = (
            SIZE        = &&count&i
            COLOR       = &&color&i.
            SYMBOL      =  &_symbol. )
            TRANSPARENCY=  &transparency.;
    %end;
    ...
run;
```

Über die **%DO**-Schleife wird auf die einzelnen Zeilen des Datensatzes zugegriffen. Um auf die einzelnen Werte (`_x&i.,_y&i.`) der Kategorien-Kombinationen zuzugreifen, müssen die *N* Kategorien-Kombinationen allerdings in jeweils *M* Einzelvariablen mit jeweils nur einem Eintrag zerlegt werden (s. unten). Ein zusätzliches Problem entsteht durch die Darstellungs-Attribute wie Größe und Farbe: Diese Optionen können **nicht** auf die Ergebnistabelle zugreifen! D.h. sie müssen über Makrovariablen gesteuert werden, die für die einzelnen Bubbles individualisiert sind.
Das Kernstück des Makros enthält also einzelne Werte aus der Ergebnistabelle (`_x&i.`, `_y&i.`), Makrovariablen, die das Aussehen einzelner Bubbles steuern (`&&count&i.`, `&&color&i.`), sowie Makrovariablen, die das Aussehen des gesamten Plots steuern (`&_symbol., &transparency.`).[18]

3.5 Überführung der Tabelle in eine Diagonalmatrix

Die Überführung der Kategorien-Kombinationen mit *N* Einträgen in *N* Variable mit jeweils einem Eintrag erfolgt durch den folgenden Code:

```
data _plot_; set _final;
    by descending count &group.;
    array _x[&last.] _x1-%sysfunc(cats(_x,&last.));
    array _y[&last.] _y1-%sysfunc(cats(_y,&last.));
    array _count[&last.] _count1-%sysfunc(cats(_count,&last.));
    ...
    %do _n = 1 %to &last.;
        if _N_ = &_n. then do;
            _x[&_n.] = &cat1.;
            _y[&_n.] = &cat2.;
            _count[&_n.] = sqrt(count/3.14159)+...;
        end;
    end;
run;
```

Zunächst werden über die Arrays die einzelnen Variablen in der Größenordnung von *N* Einträgen angelegt.[19] Über die **%DO**-Makro-Schleife und die Makro-Kontrollvariable

[18] Die Makro-Variable &_symbol. legt fest, ob der Kreis ohne (Wert "circle") oder mit Füllung (Wert "circleFilled“) gezeichnet wird. Diese Makrovariablen könnten theoretisch auch pro Bubble festgelegt werden.

[19] Die Verwendung von Arrays ist nicht zwingend erforderlich, die Variablen könnten auch einfach bei der Übergabe der Werte angelegt werden. Durch die Arrays werden die Variablen allerdings erst in der entsprechenden Reihenfolge angelegt und später mit Werten gefüllt, was die Tabelle übersichtlicher macht (Variablenreihenfolge: _x1 _x2 _x3 ... _y1 _y2 _y3... usw.) . Ohne Arrays wird immer die aktuelle Variable angelegt und sofort mit einem Wert beschrieben, so dass die Variablenreihenfolge in der Tabelle dann diese Form hat: _x1 _y1 _count1 ... _x2 _y2 _count2 ... usw. Eine weitere Möglichkeit wäre die Verwendung der LENGTH-Anweisung zur Festlegung der Variablenreihenfolge.

&_n. wird dann auf die einzelnen Zeilen _N_ des Datensatzes zugegriffen und die Werte übergeben. Die resultierende Tabelle hat folgende Form:

N	Gruppe	Kat1	Kat2	count	_x1	_x2	_x3	_y1	_y2	_y3	_count1	...	_red1	_blue1
1	Angest.	3	2	5	3			2			25.4		180	120
2	Angest.	1	3	4		1			3					
3	Arbeiter	4	1	3			4			1				
4	Selbst.	2	4	3										
5	Selbst	1	3	2										

Abbildung 8: Diagonalmatrix der Ergebnistabelle

Nun kann über die **%DO**-Schleife innerhalb von **PROC SGPLOT** auf die einzelnen Werte über die Makro-Variablen `_x&i.` und `_y&i.` zugegriffen werden:

```
proc sgplot data=_plot_
    ...
    %do i=1 %to &last.
        SCATTER x= _x&i. y=_y&i./MARKERATTRS=(
                                          SIZE= _count1 ) ...;
    %end;
    ...
run;
```

Nach wie vor ist aber das Problem ungelöst, wie den Darstellungs-Parametern des Plots die entsprechenden Werte übergeben werden können (hier _count1 und das Attribut SIZE). Dazu müssen die individualisierten Kategorienvariablen an entsprechende individualisierte Makro-Variablen übergeben werden:

```
data _plot_; set _final;
    by descending count &group.;
    array _color[&last.] __color%sysfunc(cats(_color,&last.));
    ...
    %do _n=1 %to &last.;
        if _N_ = &_n. then do;
            _count[&_n.] = sqrt(count/3.14159)+...;
            _color[&_n.] = cats("CX",put(_red[&_n.],hex2.),
                                     put(_green[&_n.],hex2.),
                                     put(_blue[&_n.],hex2.);
            call symput(cats('color',&_n.),_color[&_n.]);
            call symput(cats('count',&_n.),_count[&_n.]);
        end;
    end;
run;
```

Die Übergabe erfolgt mit den **CALL SYMPUT**-Statements am Ende nach der Übergabe der Werte an die Arrays. Es entstehen also *N* individualisierte Makro-Variablen je Plot-Attribut.
Auf diese Makro-Variablen kann nun innerhalb von **PROC SGPLOT** durch doppelte Auflösung zugegriffen werden:

```
proc sgplot data=_plot_
    %do i=1 %to &last.
        SCATTER x=_x&i. y=_y&i./MARKERATTRS=(
                                          SIZE = &&count&i.) ...;
    %end;
run;
```

Die einzelnen Makro-Variablen für die Attribute werden dann entsprechend dem folgenden Beispiel zu einzelnen Werten pro Bubble aufgelöst:

```
SIZE = &&count&i.  →   SIZE = &count3   →   SIZE = 25
```

Durch die Auflösung des Makro-Codes werden so einzelne **SCATTER**-Statements generiert, die dann sequentiell abgearbeitet werden:

```
proc sgplot data=_plot_;
        SCATTER x=3 y=4/MARKERATTRS=(SIZE = 25);
        SCATTER x=3 y=4/MARKERATTRS=(SIZE = 22);
        SCATTER x=5 y=1/MARKERATTRS=(SIZE = 17);
        SCATTER x=2 y=1/MARKERATTRS=(SIZE = 14);
        ...
run;
```

3.6 Legenden-Bildung

In ähnlicher Form wie die Gestaltung der einzelnen **SCATTER**-Statements lässt sich auch die Bildung der Legende für den Plot flexibel gestalten und durch Makro-Variablen steuern:

```
proc sgplot data=_plot_;
...
    SCATTER x=_x&i. y=_y&i./...;
    %if &legend.=on %then %do;
        %if "group." ne "" %then %do;
            keylegend
                %do i=1 %to &n_groups.;
                    "%sysfunc(cats(group_,&i.))"
```

```
                %end;
                    /location=outside position=bottom ...
                    ;
        %end;
    %end;
run;
```

Das LEGEND-Statement wird also zur Laufzeit nur dann generiert, wenn überhaupt eine Legende erwünscht ist (`&legend.=on`) und wenn Untergruppen vorhanden sind (`"group." ne ""`). Die Zuordnung der Untergruppen (die ebenfalls in der Diagonalmatrix generiert werden und mit LEGENDLABEL= und NAME= dem entsprechenden Bubble bzw. **SCATTER**-Statement zugeordnet werden; hier nicht gezeigt) erfolgt dann über die Generierung der jeweiligen Legende-Namen in der Laufzeit.[20] Wichtig ist das letzte Semikolon, das die zu generierende **LEGEND**-Anweisung abschließen muss. Der durch die Makro-Variablen und -Anweisungen generierte Code könnte dann z.B. so aussehen:

```
proc sgplot data=_plot_;
    SCATTER x=3 y=4/MARKERATTRS=(SIZE = 25);
    SCATTER x=3 y=4/MARKERATTRS=(SIZE = 22);
    ...
            keylegend "group1" "group2" "group3"
                    /location=outside position=bottom
                    ;
run;
```

4 Anwendungsbeispiele des Bubble-Plot-Makros

Das komplette Makro enthält noch einige Funktionalitäten mehr als hier exemplarisch aufgezeigt. Die wichtigsten Elemente zur Umsetzung des Makros wurden oben dargestellt. Im Folgenden werden nun noch ein paar Anwendungsbeispiele aufgezeigt.

4.1 Einfachste Form der Anwendung: keine Untergruppen

In der einfachsten Form der Anwendung entspricht der Datensatz Szenario (A) und enthält keine Untergruppen und Einzelbeobachtungen (siehe Abbildung 9).

ID	Var1	Var2
1	2	4
2	2	4
3	3	1
4	2	4

Abbildung 9: Einfacher Datensatz mit Einzelbeobachtungen und ohne Gruppen

[20] Auch das Erscheinungsbild der restlichen hier gezeigten und weiteren Optionen des Legend-Statements kann natürlich über Makro-Variablen flexibel gesteuert werden.

Der Aufruf des Makros sieht folgendermaßen aus:

```
%bubble(tabIn=test,cat1=var1,cat2=var2);
```

Das Ergebnis:

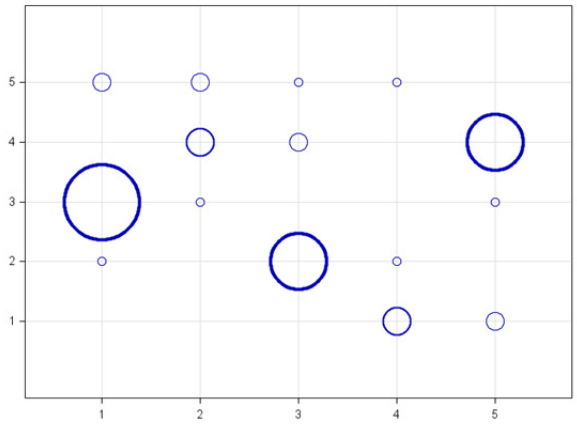

Abbildung 10: Ergebnisausgabe einfachste Form

Hierbei sind die Standardeinstellungen des Makros:

- Skalierungsreferenz = Radius
- Skalierungsfaktor = 10
- Bubbles nicht gefüllt
- Keine Legende

4.2 Einzelbeobachtungen mit Untergruppen

Die Datenstruktur (entspricht Szenario (A); Beispiel Fragebogen) enthält hier zusätzlich zu den Einzelbeobachtungen eine Gruppierungsvariable:

ID	Var1	Var2	Level (group)
1	2	4	High
2	2	4	High
3	3	1	Middle
4	2	4	Low

Abbildung 11: Datenstruktur mit Gruppierungsvariable

Der Aufruf des Makros könnte folgendermaßen aussehen, wobei es hier nun wichtig ist, die Gruppierungsvariable mit zu benennen. Zusätzlich werden hier noch die Achsen beschriftet, die Referenz der Bubble-Größen auf *Fläche* gesetzt, die Legende über zwei Zeilen geschrieben und die Bubbles gefüllt. Da hier einige Bubbles in derselben Kategorien-Kombination dieselbe Größe aufweisen, werden diese mit der Shift-Option leicht versetzt dargestellt:

```
%bubble(tabIn=test, cat1=var1, cat2=var2, group=level, legend_on=1,
    label_x=category 1, label_y=category 2,ref=area, shift=2, fill=1,
    legend_rows=2);
```

Der Ergebnis-Plot gestaltet sich wie folgt (es sind auch andere Darstellungsformen möglich):

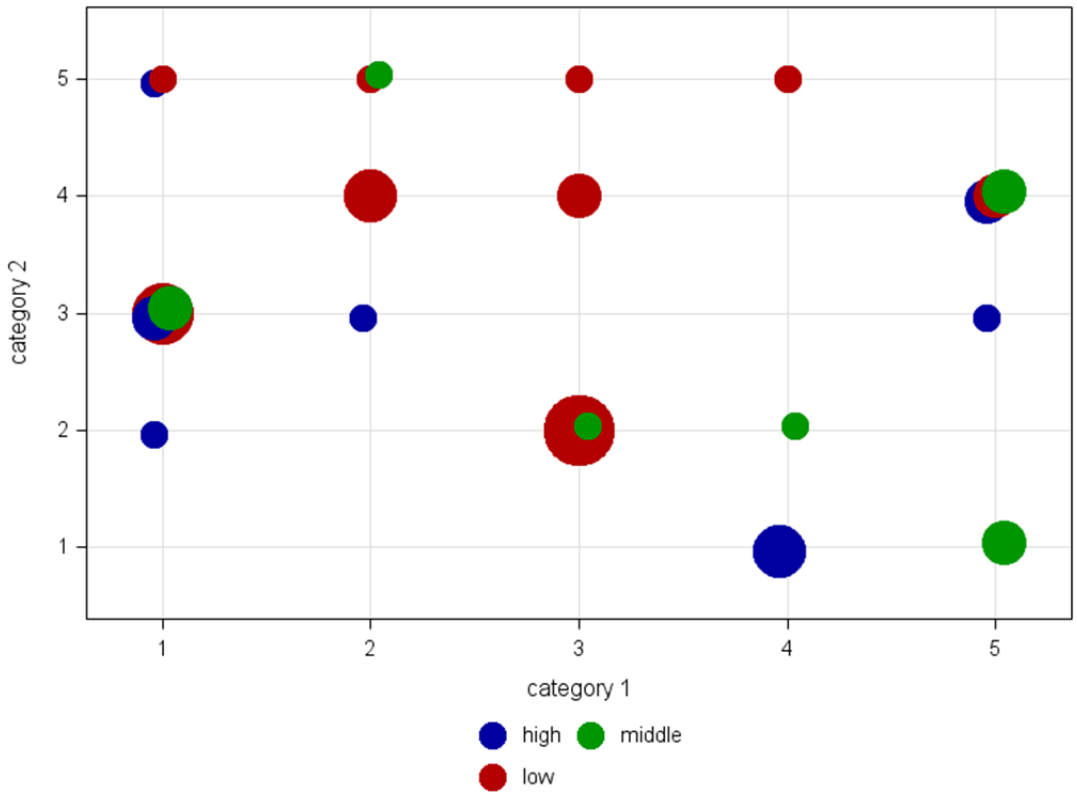

Abbildung 12: Plot mit Gruppierungsvariablen und Shift-Option

4.3 Aggregierte Daten mit Untergruppen

Das Makro ist auch in der Lage, neben Einzelbeobachtungen bereits aggregierte Daten zu verarbeiten (Szenario (B) in der Einführung). Die Datenstruktur hierzu sieht folgendermaßen aus:

Country	Var1	Var2	Size
Australia	2.3	4.2	6.9
Germany	2.5	4.7	5.1
Russia	3.1	1	14.5
Zambia	2	4	11.9

Abbildung 13: Datensatz mit aggregierten Daten

Hier stellt nun jede Zeile eine eigene Untergruppe dar. Der Makro-Aufruf dazu:

```
%bubble(tabIn=test2, cat1=var1, cat2=var2, count=size,
    group=country, label_x=category 1, label_y=category 2,
    fill=1, ref=area, transparency=0.9);
```

Neben den bereits im letzten Abschnitt erwähnten Optionen wurde hier noch die transparente Darstellung der Bubbles gewählt, um Überlappungen besser sichtbar zu machen. Das Ergebnis hat folgendes Aussehen:

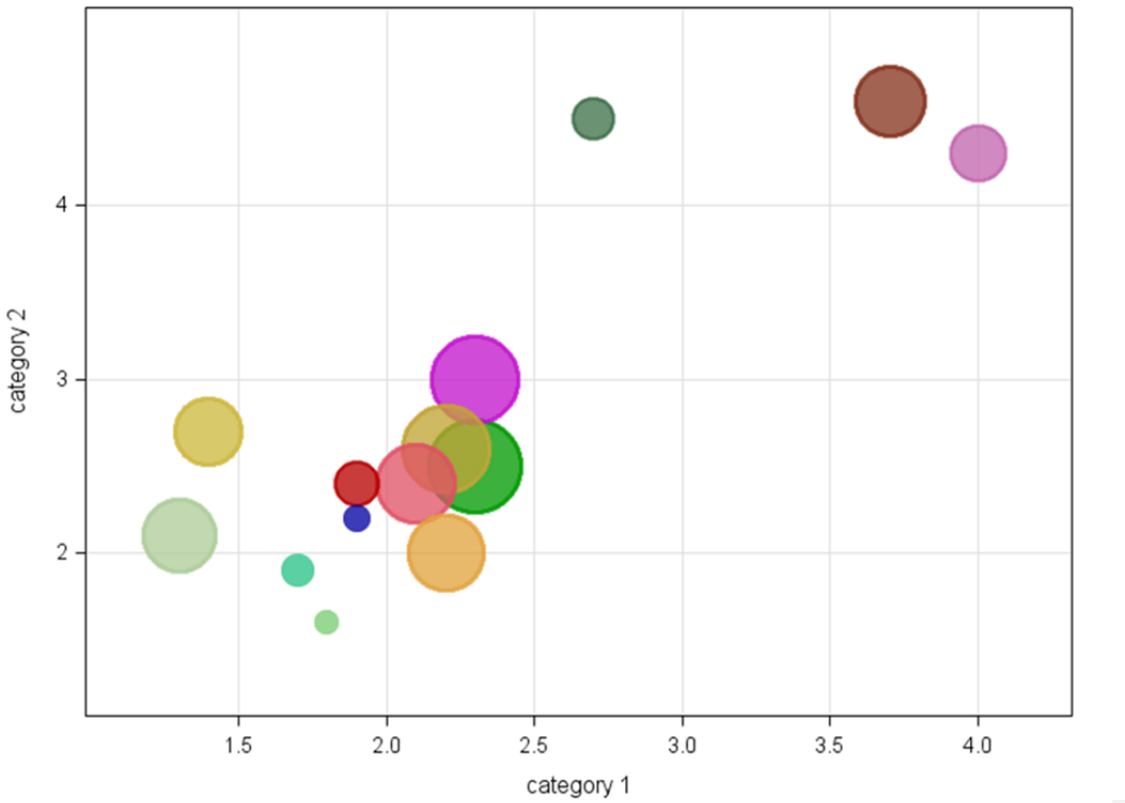

Abbildung 14: Darstellung aggregierter Daten mit transparenten Bubbles