Benutzung der compare() Funktion zur Darstellung von Unterschieden zwischen zwei Zeichenketten

Véronique Bourcier
Director - xxformat GmbH
c/o STARTPLATZ, Im Mediapark 5
50670 Köln
veronique.bourcier@xxformat.com

Zusammenfassung

Der Inhalt von 2 Zeichenketten, die in unterschiedlichem Zusammenhang verwendet werden, kann sich auf Grund des finalen Einsatzes und technischer Limitierung leicht unterscheiden. In diesem Falle können die `compare()` Funktion, ODS Syntax und referenzierte Umwandlungstabellen benutzen werden, um Unterschiede zwischen zwei Zeichenketten durch unterschiedliche Färbung darzustellen.

Schlüsselwörter: Compare, ODS EXCEL, ODS LISTING, validvarname=any, n-literal

1 Einführung

1.1 Warum möchten wir Unterschiede zwischen zwei Zeichenketten mit Farben hervorheben?

Ich kann mir mindestens drei Situationen vorstellen, in denen wir die Unterschiede zwischen zwei Zeichenketten mit Farben hervorheben möchten:

- Um sicherzustellen, dass die Unterschiede zwischen zwei Zeichenkettenversionen korrekt sind, während Sie eine Validierung durchführen, indem Sie zwei Versionen derselben Daten zu unterschiedlichen Zeitpunkten oder eine projektspezifische Version mit einer Standard- / Referenzversion vergleichen.
- Um Werte zu harmonisieren.
- Um sicherzustellen, dass neue Einträge, die noch nicht existieren, angefordert werden können (z. B. neuer Wert in einer Kodeliste).

Farben beschleunigen im Vergleich zu Zahlen die Überprüfung und geben die Position an, wo Unterschiede auftreten, vor allem bei langen Zeichenketten. Augen können sehr schnell die Unterschiede erfassen. Eine automatisierte Überprüfung ist immer schneller und zuverlässiger als eine rein manuelle Überprüfung. Ist ein Benutzer schneller über Unterschiede informiert, kann er zeitnah reagieren.

1.2 Was wird in diesem Dokument behandelt?

Hier sprechen wir über die Erstellung einer Datei, die Farben verarbeiten kann (Excel, HTML, etc.). Diese Datei enthält den Originaltext und einen Referenztext, gegen den er verglichen wird. Der Text zwischen dem ersten und dem letzten Unterschied ist rot markiert. Darüber hinaus werden Ausdrücke, die als Synonyme (Synonym-Tabelle) bezeichnet werden, blau markiert.

Erst entdecken Sie jedes einzelne Konzept, das später in Abschnitt 7 verwendet wird:

Abschnitt 2: `compare()` Funktion
Abschnitt 3: Verwenden Sie Farben in `proc print` und `proc report`
Abschnitt 4: Markieren Sie in blau Synonyme
Abschnitt 5: Markieren Sie in rot Unterschiede zwischen zwei Zeichenketten
Abschnitt 6: Tauschen Sie Zeichenketten

Abschnitt 7 beschreibt detailliert die verschiedenen Schritte, um die Datei Anlistungsunterschiede zu generieren:

Schritt 1: Einführung in die Daten
Schritt 2: Zeichenketten vergleichbar machen
Schritt 3: Unterschiede in rot markieren
Schritt 4: Synonyme in blau markieren
Schritt 5: Unterschiede in rot und Synonyme in blau markieren

2 `compare()` Funktion

2.1 Standardeinstellung

2.1.1 `compare()` Funktion: Argumente

Die Funktion `compare()` hat drei Argumente, von denen zwei obligatorisch sind und eins optional. Die beiden obligatorischen Argumente beziehen sich auf die beiden zu vergleichenden Zeichenketten. Das optionale Argument listet Änderungen an.

2.1.2 `compare()` Funktion: Ergebnis

Die Funktion `compare()` gibt eine Ganzzahl zurück. Die Zahl gibt die Position des am weitesten links liegenden Zeichens an, durch die sich Zeichenketten unterscheiden oder den Wert `0`, wenn es keinen Unterschied gibt.

Das Vorzeichen des Wertes zeigt an, ob Zeichen der ersten Zeichenkette "kleiner" sind als das der zweiten Zeichenkette.

Tabelle 1: compare() Funktion – Ergebnis

#	string 1	string 2	compare (string1 , string2)	
#1			0	Kein Unterschied
#2	A2Z	A2Z	0	Kein Unterschied
#3	ABC	ABCD	-4	1. Unterschied beim 4. Zeichen
#4	WXYZ	XYZ	-1	1. Unterschied beim 1. Zeichen
#5	XYZ	WXYZ	1	1. Unterschied beim 1. Zeichen

- Im Fall `#3`, ist leer 'kleiner' als `D`.
- Im Fall `#4`, ist `W` 'kleiner' als `X`.
- Im Fall `#5`, ist `X` 'größer' als `W`.

2.1.3 Sortierung von Zeichenwerten

Es gibt zwei Arten von Sortierreihenfolgen: die Reihenfolge für EBCDIC-Zeichen, die von IBM entwickelt wurde und die Reihenfolge für ASCII-Zeichen. Die verwendete Reihenfolge auf ihrem Computer hängt vom Betriebssystem ab.

EBCDIC-Zeichen werden in Systemen wie CMS oder OS/390 (Mainframe) verwendet. Hier die Reihenfolge der EBCDIC-Zeichen (für die englische Sprache), die vom SAS-System verwendet werden:

```
leer . < ( + | & ! $ * ) ; ¬ - / , % _ > ? : # @ ' = "
a b c d e f g h i j k l m n o p q r ~ s t u v w x y z
{ A B C D E F G H I } J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
```

Leer < Kleinbuchstaben < Großbuchstaben < Ziffer

ASCII-Zeichen werden in Systemen wie Macintosh, MS-DOS, OpenVMS, PC DOS, UNIX und Derivaten, Windows oder OS/2 verwendet. Hier ist die Reihenfolge der ASCII-Zeichen (für die englische Sprache), die vom SAS-System verwendet werden:

```
leer ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ? @
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^
_
a b c d e f g h i j k l m n o p q r s t u v w x y z { } ~
```

Leer < Ziffer < Großbuchstaben < Kleinbuchstaben

2.2 `compare()` Funktion: Modifikatoren

Wir brauchen nur die beiden obligatorischen Parameter der `compare()` - Funktion, um die Farbe einer gegebenen Zeichenkette zu ändern, wenn Unterschiede mit einer anderen Zeichenkette identifiziert werden. Trotzdem möchte ich noch die vier von der `compare()` - Funktion verwendeten Modifikatoren vorstellen.

Hier nun die Definition der vier Modifikatoren:

i/I Dieser Modifikator ignoriert **Groß-/Kleinschreibungsunterschiede**.

l/L Dieser Modifikator entfernt **führende Leerzeichen**, bevor er die beiden Ketten vergleicht.

n/N Anführungszeichen und abschließende n der **n-Literal-Syntax** werden ignoriert sowie **Groß-/Kleinschreibungsunterschiede**.

: Dieser Modifikator **trennt die längsten der beiden Zeichenketten**, bevor er sie vergleicht.

Die folgende Tabelle zeigt das Ergebnis der `compare()`- Funktion in Abhängigkeit vom Modifikator, der in einer Umgebung mit ASCII-Zeichen verwendet wird.

Tabelle 2: compare() Funktion - Modifikatoren

			compare(string1,string2)						compare(strip(ref_string),strip(string),':')
#	string1	string2		I	L	N	LN	:	:
#1	ABC	ABC	0	0	0	0	0	0	0
#2	ABC	AbC	-2	0	-2	0	0	-2	-2
#	ABC	ABC	1	1	0	1	0	1	1

			compare(string1,string2)						compare(strip(ref_string),strip(string),':')
#	string1	string2		I	L	N	LN	:	:
3									
#4	ABC	'AbC'n	1	1	1	0	0	1	1
#5	ABC	'AbC1'n	1	1	1	-4	-4	1	1
#6	ABC	'AbC'n	1	1	1	1	0	1	1
#7	ABC	AB	3	3	3	3	3	3	0

- Beim `I`-Modifikator wird der Unterschied bei **Groß-/Kleinschreibung** in Beispiel `#2` ignoriert
- Beim `L`-Modifikator wird der Unterschied durch f**ührende Leerzeichen** ignoriert. `#3` zeigt keinen Unterschied, während `#6` einen Unterschied anzeigt.
- Bei `N`-Modifikator wird der Unterschied bei **Groß-/Kleinschreibung** in Beispiel `#2` ignoriert; Zusätzlich werden **die Anführungszeichen und das abschließende n der n-Literal-Syntax** im Fall `#4`, `#5` und `#6` ignoriert:
  - Bei `#4` wird kein Unterschied gefunden.
  - Bei `#5` findet sich der erste Unterschied in Position 4 (`ABC` gegen `ABC1`).
  - Bei `#6` ist der erste Unterschied in Position 1 (führender Leerzeichen).
- Bei `LN` werden zunächst führende Leerzeichen aus dem Vergleich entfernt, weil `L` vor `N` steht. Das Ergebnis von `#3` und `#6` ist verändert.
- Beachten Sie, dass beim Doppelpunkt (`:`)-Modifikator abschließende Leerzeichen entfernt werden müssen, um keinen Unterschied zwischen den beiden Zeichenketten-Teilmengen (`AB` gegen `AB`) zu beobachten.

2.3 Beispiel für n-Literal

Die n-Literal-Syntax ist beim Lesen von Excel-Dateien interessant. Hier ein Beispiel:

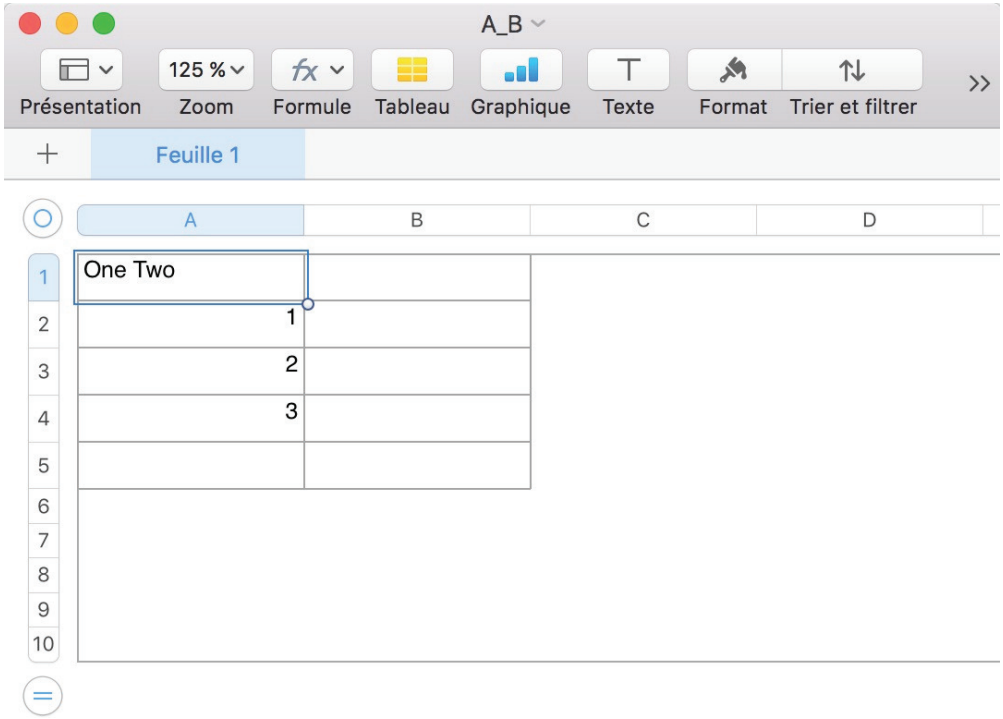

Abbildung 1: Beispiel für eine zu lesende Excel-Datei

```
options validvarname=any;
libname xls xlsx '/…/KSFE/A_B.xlsx' ;
proc print data=xls.'Feuille 1'n width=min noobs;
run;
```

In dem Beispiel heißt die Registerkarte `Feuille 1`, d.h. ein Registerkartenname, der ein Leerzeichen enthält. Dieser Name ist auch der Datei-Name, der im **`proc print`** verwendet wird. Er wird in Anführungszeichen gesetzt, gefolgt von `n`. Damit das Programm richtig funktioniert, benötigen wir die Option `validvarname = any`.

3 Farben

3.1 Farben mit `proc print` und `proc report`

Hier sind vier Beispiele, in denen Farben mit dem Schlüsselwort `foreground` zugeordnet werden:

```
data one;
   x='ABC'; output;
   x='XYZ'; output;
run;
```

```
*Beispiel 1;

proc print data=one;
   var x / style(column)=[foreground=blue];
run;

*Beispiel 2;

proc report data=one;
   columns x;
   define x / display style(column)=[foreground=blue];
run;
```

Sowohl `proc print` als auch `proc report` geben im obigen Beispiel das gleiche Ergebnis aus.

```
*Beispiel 3: Farben mit einem Format zuordnen;

proc format;
   value $ example 'ABC'='blue';
run;
proc print data=one;
   var x / style(column)=[foreground=$example.];
run;

*Beispiel 4: Zeichen mit compute/endcomp zuordnen;
proc report data=one;
   columns x;
   define x / display;
   compute x;
      if x='ABC' then call define (_ROW_
                        ,'style','style=[foreground=blue]');
   endcomp;
run;
```

3.2 ODS-Format

Bei der Verwendung einer Lösung wie SAS Studio kann das Standardziel `html5(web)` Farben anzeigen. Andernfalls müssen wir ein Ziel aktivieren, das Farben darstellen kann (HTML, EXCEL, RTF, PDF und so weiter). Um eine Excel-Datei zu generieren, verwenden wir `ods excel`-Anweisung (verfügbar ab SAS 9.4):

```
ods excel file='…/one.xlsx';
…
ods excel close;
```

Beim Entwickeln / Aktualisieren eines Programms kann eine Textdatei praktischer sein. Daher gibt es `ods listing`.

```
ods listing file='/…/one.txt';
proc print data=one;
run;
ods listing close;
```

Um ein anderes Ausgabeziel zu stoppen, können wir folgende Syntax verwenden:

```
ods _all_ close;
```

Wenn wir nur das SAS Studio default `html5(web)` Ausgabeziel stoppen wollen, können wir das wie folgt machen:

```
ods html5(web) close;
```

3.3 Füge Farben in einem Titel hinzu

Um Farben in einem Titel zu verwenden, definieren wir zunächst ein Zeichen unserer Wahl.

```
ods escapechar='^';
```

Dann verwenden wir dieses Zeichen als Präfix von `{…}`.

```
title 'Example: ^{style[color=red]Red Text} Back Text';
```

4 Markieren von Synonymen in blau

Das primäre Ziel des Vortrags ist es, Ihnen zu erklären, wie man eine Datei erzeugt, die Unterschiede zwischen zwei Zeichenketten mit Farben darstellt. Einige Arten von Unterschieden sind möglicherweise nicht relevant. Wenn es unser Ziel ist, Unterschiede in der Bedeutung zwischen zwei Zeichenketten zu identifizieren, dann wird die Verwendung eines Synonyms nicht als ein Unterschied betrachtet.
In Abschnitt 7 werden wir sehen, wie wir Differenzen in der Bedeutung von Differenzen in Phrasierungen (Synonyme) unterscheiden. Bisher sehen wir nur den Code, um in blau Begriffe die also Synonyme referenziert sind darzustellen.

4.1 `synonym` Datensatz

Die Synonyme werden in einem Datensatz namens `synonym` mit zwei Variablen referenziert: `synonym` und `ref_value`. Wenn ein Begriff in der Spalte `synonym` in unseren Daten auftaucht, dann wird er in blau geschrieben werden.

Tabelle 3: synonym Datensatz

synonym	ref_value
yrs	years
SBP	Systolic Blood Pressure
DBP	Diastolic Blood Pressure

4.2 Aktuelle Daten

Hier sind unsere aktuellen Daten.

Das Ziel ist es, in blau ein Synonym zu setzen, das in der `description` Variable erscheint.

Tabelle 4: description Datensatz - Aktuelle Daten

key	description
1	18 to 65 yrs old yrs old
2	18-65 yrs oxd
3	29-65 yrs old
4	29
5	19 to 65 yrs olx
6	SBP is lower than x
7	Diastolic Blood Pressure is lower than x

4.3 `tranwrd()` Funktion

Wir beginnen mit einer rein manuellen Lösung, um die Beweisführung zu verstehen. Die Funktion `tranwrd()` wird verwendet, um `^{style[color=blue]` und `}`, um Synonyme hinzuzufügen.

```
data description;
   set description;
   length new $200;
   new=description;
   new=tranwrd(new,'yrs','^{style[color=blue]yrs}');
   new=tranwrd(new,'SBP','^{style[color=blue]SBP}');
   new=tranwrd(new,'DBP','^{style[color=blue]DBP}');
run;
```

4.4 Verwenden Sie Makrovariablen

Wir können die Style-Syntax in Makros-Variablen abspeichern:

```
%let blue_start=^{style[color=blue];
%let blue_end  =};
```

4.5 Verallgemeinerung

Der Programmiercode kann in einem Datenschritt verallgemeinert werden.

```
data _null_;
   set synonym end=eof;
   if _n_=1 then
      do;
         call execute('data description;  ');
         call execute('   set description;');
         call execute('   length new $200;');
         call execute('   new=description;');
      end;
   call execute('new=tranwrd('
                  || ' new'
                  || ',"'' || synonym || '"'
                  || ',"&blue_start.' || synonym
                  || ' &blue_end."'
                  ||       ');');
   if eof then call execute ('run;');
run;
```

5 Unterschiede zwischen zwei Zeichenketten anlisten

Ich benutze Dummy-Daten, damit Sie die verwendete Logik aus Abschnitt 7 verstehen, insbesondere, wie man die Position der ersten Unterschiede und die Länge der Zeichenfolge bis zum letzten Unterschied identifiziert. Wie viele Kombinationen von `x` und `y` können eine Zeichenkette von 4 Zeichen haben? Hier ist die Liste der 16 möglichen Kombinationen.

Tabelle 5: Kombinationen von x und y für eine Zeichenkette von 4 Zeichen

	x, Startposition	Länge zwischen dem ersten und dem letzten x
yyyy	0	0
xyyy	1	1
xxyy	1	2

	x, Startposition	Länge zwischen dem ersten und dem letzten x
xyxy	1	3
xxxy	1	3
xyyx	1	4
xyxx	1	4
xxyx	1	4
xxxx	1	4
yxyy	2	1
yxxy	2	2
yxyx	2	3
yxxx	2	3
yyxy	3	1
yyxx	3	2
yyyx	4	1

Wie haben wir die Position des ersten `x` gefunden? Wie haben wir die Anzahl der Zeichen zwischen dem ersten `x` und dem letzten `x` gefunden?

- **Die Position des ersten `x` (`x_start`)** wird durch die `index()`-Funktion bestimmt.
- **Die Position des letzten `x` (`x_end`)** ist die gesamte Zeichenkettenlänge minus der Position des ersten x in der umgekehrten Zeichenkette plus `1`.
- **Die Länge zwischen dem ersten `x` und dem letzten `x` (`x_lgth`)** ist die Position des letzten `x` (`x_end`) minus der Position des ersten `x` (`x_start`) plus `1`.

```
data one (drop=x_end);
   set one;
   x_start=index(new,'x');
   if x_start=0 then x_lgth=0;
   else
      do;
         x_end                                  =length(trim(new))-
index(reverse(trim(new)),'x')+1;
         x_lgth =x_end-x_start+1;
      end;
run;
```

Wir brauchen die Startposition und die Zeichenkettelänge, um die `substr()`-Funktion zu verwenden. Wir müssen nur mit 5 Szenarien arbeiten:

Tabelle 6: 5 Szenarien

Kein x	…	y				
Beginnend und endend mit x	**x…x**	red_start	x			red_end
Endend mit x	**…x**	y		red_start	x	red_end
Beginnend mit x	**x…**	red_start	x	red_end	y	
x in der Mitte	**…x…**	y	red_start	x	red_end	y

Der Programmcode sieht wie folgt aus:

```
ods escapechar='^';
%let red_start = ^{style[color=red];
%let red_end   = };

data one;
   set one;
   length _new $40;

   *...;
   if x_start=0 then _new=new;

   *x...x;
   else if x_start=1 and x_lgth=length(trim(new))
   then _new = "&red_start."
            || new
            || "&red_end.";

   *x...;
   else if x_start=1 and x_lgth < length(trim(new))
   then _new = "&red_start."
            || substr(new,x_start,x_lgth)
            || "&red_end."
            || substr(new,x_start+x_lgth);

   *...x;
```

```
   else if x_start>1 and x_lgth = length(trim(new))-
x_start+1
   then _new = substr(new,1,x_start-1)
             || "&red_start."
             || substr(new,x_start)
             || "&red_end." ;

   *...x...;
   else if x_start>1 and x_lgth < length(trim(new))-
x_start+1
   then _new = substr(new,1,x_start-1)
             || "&red_start."
             || substr(new,x_start,x_lgth)
             || "&red_end."
             || substr(new,x_start+x_lgth);
run;
```

6 Zeichenketten austauschen

Wir haben zwei Ziele:

- Ziel 1: wir wollen die Zeichen `//aa\\` der `x`-Variablen durch `//zzzz\\` der `y`-Variablen ersetzen.
- Ziel 2: wir wollen die Zeichen `//11\\` der x-Variablen durch `//2\\` der `y`-Variablen ersetzen.

Tabelle 7: Von x nach x_new

x	y	x_new
x//aa\\b//11\\c	b//zzzz\\mm//2\\s	x//zzzz\\b//2\\c

Ziel 1: um `//aa\\` der `x`-Variablen durch `//zzzz\\` der `y`-Variablen zu ersetzen, benötigen wir Folgendes:

- die Position von `aa` (`x_start`) und die Länge von `aa` (`x_lgth`)
- die Position von `zzzz` (`y_start`) und die Länge von `zzzz` (`y_lgth`)

Damit können wird die Zeichenkette durch Verwendung der Funktion `substr()` extrahieren. Dann müssen wir nur noch die beiden Zeichenketten mit der Funktion `tranwrd()` austauschen.

Ziel 2: wir wiederholen die Operation so lange wir `//` Zeichen finden. Mit der Funktion `count()` wird die Anzahl der benötigten Schleifen identifiziert.

Mit der Funktion `index()` wird die Startposition der ersten `//` Zeichen identifiziert. Diese Funktion kann nur die Position des ersten Auftretens zurückgeben. Daher wurde am Ende der Schleife der Anfang des Textes gelöscht.

Original-Zeichenketten werden in den Variablen `x_old` und `y_old` gespeichert.

```
data one (keep=x_old y_old x_new);
   set one;
   length x_old y_old x_new $30;
   cnt=count(x,'//');
   x_old=x;
   y_old=y;
   x_new=x;
   do i=1 to cnt;
      x_start = index(x,'//') + 2;
      x_lgth  = index(x,'\\') - x_start;
      y_start = index(y,'//') + 2;
      y_lgth  = index(y,'\\') - y_start;
      x_new   = tranwrd(x_new
                        ,substr(x,x_start,x_lgth)
                        ,substr(y,y_start,y_lgth));
      x = substr(x,x_start+x_lgth+2);
      y = substr(y,y_start+y_lgth+2);
   end;
run;
```

7 Markieren Sie Unterschiede in rot und Synonyme in blau

7.1 Schritt 1: Einführung in die Daten

Zuerst hier die synonym-Datei:

Tabelle 8: synonym Datensatz

synonym	ref_value
yrs	years
SBP	Systolic Blood Pressure
DBP	Diastolic Blood Pressure

Wir arbeiten mit zwei Datensätzen: Einer mit dem Referenztext (`ref_description` Variable) und einer mit dem zu prüfenden Text (`description` Variable). Die Datensätze sind durch eine Variable namens `key` verknüpft. Beide Datensätze werden zusammengeführt. Das Ergebnis sieht wie folgt aus:

Tabelle 9: ref_description + description Variablen

key	ref_description	description
1	18-65 years old	18 to 65 yrs old yrs old
2	18-65 years old	18-65 yrs oxd
3	18-65 years old	29-65 yrs old
4	18-65 years old	29
5	18-65 years old	19 to 65 yrs olx
6	Systolic Blood Pressure is lower than x	SBP is lower than x
7	DBP is lower than x	Diastolic Blood Pressure is lower than x

7.2 Schritt 2: Zeichenketten vergleichbar machen

Wir erstellen zwei neue Variablen …_syn:

- Wir fügen //…\\ um die Ausdrücke hinzu, die ref_value im synonym-Datensatz sind.
- Wir fügen //…\\ um die Ausdrücke hinzu, die synonym im synonym-Datensatz und ersetzen die Ausdrücke durch den ref_value.

Wir **ersetzen** in Ausdrücken **auch Leerzeichen durch Sternchen**. Zum Beispiel wird aus Systolic Blood Pressure dann Systolic*Blood*Pressure.

```
data _null_;
   set &insyn. end=eof;
   if _n_=1 then
      do;
         call execute ("data &outdsn.;");
         call execute ("   set &indsn.;");
         call execute ("   length &outvar. $200;");
         call execute ("   &outvar.  = &invar.;");
      end;
   ************************************************;
   * Example: replace Systolic Blood Pressure
   *          by      //Systolic*Blood*Pressure\\;
   ************************************************;

   call execute("&outvar.=tranwrd(&outvar."
```

```
          || ",'"   || trim(ref_value)                    ||
"'"
          || ",'//" || tranwrd(trim(ref_value),' ','*') ||
"\\');");

   ********************************************;

   ref_value=tranwrd(trim(ref_value),' ','*');

   ********************************************;
   * Example: replace SBP
   *          by      //Systolic*Blood*Pressure\\;
   ********************************************;

   call execute("&outvar.=tranwrd(&outvar."
            || ",'"   || trim(synonym)   || "'"
            || ",'//" || trim(ref_value) || "\\');");

   if eof then call execute('run;');
run;
```

Das Ergebnis sieht aus wie folgt aus:

Tabelle 10: Zeichenketten vergleichbar machen

key	ref_description_syn	description_syn
1	18-65 //years\\ old	18 to 65 //years\\ old yrs old
2	18-65 //years\\ old	18-65 //years\\ oxd
3	18-65 //years\\ old	29-65 //years\\ old
4	18-65 //years\\ old	29
5	18-65 //years\\ old	19 to 65 //years\\ olx
6	//Systolic*Blood*Pressure\\ is lower than x	//Systolic*Blood*Pressure\\ is lower than x
7	//Diastolic*Blood*Pressure\\ is lower than x	//Diastolic*Blood*Pressure\\ is lower than x

7.3 Schritt 3: Unterschiede in rot markieren (_red Variable)

Tabelle 11: Unterschiede in rot markieren

key	_red
1	18 to 65 //years\\ old yrs old
2	18-65 //years\\ oxd
3	29-65 //years\\ old
4	29
5	19 to 65 //years\\ olx
6	//Systolic*Blood*Pressure\\ is lower than x
7	//Diastolic*Blood*Pressure\\ is lower than x

Um Text in rot zu schreiben, suchen wir mit einem Programmcode (ähnlich wie in Abschnitt 5) zuerst die Position des ersten Unterschiedes und die Länge bis zum letzten Unterschied.

```
data dif (drop=x_start x_end x_lgth);
   set dif ;
   x_start
=abs(compare(description_syn,ref_description_syn));
   if x_start=0 then x_lgth=0;
   else
      do;
         x_end   =length(trim(description_syn))
                  -
abs(compare(reverse(trim(description_syn))

,reverse(trim(ref_description_syn))
                                    ))
                  +1;
         x_lgth=x_end-x_start+1;
      end;
```

Dann teilen wir den Text in mehrere Teile und fügen die Syntax für Farben hinzu.

```
   if x_start=0 then _red=new;
   *x...x;
   else if x_start=1 and x_lgth=length(trim(new))
   then _red = "&red_start."
             || new
```

```
              || "&red_end.";

   length _new $300;
   *...;
   if x_start=0 then _red=description_syn;

   *x...x;
   else if x_start=1 and
x_lgth=length(trim(description_syn))
   then _red = "&red_start."
            || description_syn
            || "&red_end.";

   *x...;
   else if x_start=1 and
           x_lgth < length(trim(description_syn))
   then _red = "&red_start."
            || substr(description_syn,x_start,x_lgth)
            || "&red_end."
            || substr(description_syn,x_start+x_lgth);

   *...x;
   else if x_start>1 and
           x_lgth = length(trim(description_syn))-x_start+1
   then _red = substr(description_syn,1,x_start-1)
            || "&red_start."
            || substr(description_syn,x_start)
            || "&red_end.";

   *...x...;
   else if x_start>1 and
           x_lgth < length(trim(description_syn))-x_start+1
   then _red = substr(description_syn,1,x_start-1)
            || "&red_start."
            || substr(description_syn,x_start,x_lgth)
            || "&red_end."
            || substr(description_syn,x_start+x_lgth);
run;
```

Das Ergebnis wird in der Variable `_red` gespeichert.

7.4 Schritt 4: Synonyme in blau markieren (`_blue` Variable)

Wie in Abschnitt 4, markieren wir Synonyme im Originaltext in blau. Dann fügen wir `//…\\` um diese Synonyme hinzu, um später Werte austauschen zu können. Das Ergebnis wird in der Variable `_blue` gespeichert.

Tabelle 12: Synonyme in blau markieren

key	description	_blue
1	18 to 65 yrs old yrs old	18 to 65 //yrs\\ old yrs old
2	18-65 yrs oxd	18-65 //yrs\\ oxd
3	29-65 yrs old	29-65 //yrs\\ old
4	29	29
5	19 to 65 yrs olx	19 to 65 //yrs\\ olx
6	SBP is lower than x	//SBP\\ is lower than x
7	Diastolic Blood Pressure is lower than x	Diastolic Blood Pressure is lower than x

```
%let blue_start=^{style[color=blue];
%let blue_end  =};

data _null_;
   set synonym end=eof;
   if _n_=1 then
      do;
         call execute('data dif;');
         call execute('   set dif;');
         call execute('   length _blue $200;');
         call execute('   _blue=description;');
      end;
   call execute('_blue=tranwrd('
                || ' _blue'
                || ',"' || synonym || '"'
                ||  ',"//&blue_start.'  ||  synonym  ||  '
&blue_end.\\"'
                || ');');
   if eof then call execute ('run;');
run;
```

7.5 Schritt 5: Markieren von Unterschieden in rot und Synonymen in blau

Nun tauschen wir Zeichenketten die mit //...\\ umgeben sind zwischen der in Schritt 4 erstellten _blue-Variablen und der in Schritt 3 erstellten _red-Variablen mit einer ähnlich der in Abschnitt 6 dargestellten Syntax aus. Das Ergebnis wird in der description_new Variable gespeichert.

Tabelle 13: Unterschiede in rot und Synonyme in blau markieren (1/2)

key	_red	_blue
1	18 to 65 //years\\ old yrs old	18 to 65 //yrs\\ old yrs old
2	18-65 //years\\ oxd	18-65 //yrs\\ oxd
3	29-65 //years\\ old	29-65 //yrs\\ old
4	29	29
5	19 to 65 //years\\ olx	19 to 65 //yrs\\ olx
6	//Systolic*Blood*Pressure \\ is lower than x	//SBP\\ is lower than x
7	//Diastolic*Blood*Pressur e\\ is lower than x	Diastolic Blood Pressure is lower than x

```
data dif (drop=cnt i _red  red_start  red_lgth
                     _blue blue_start blue_lgth);
   set dif;
   length description_new $300;
   cnt=count(_red,'//');

   description_new=_red;
   if index(_blue) ne 0 then do i=1 to cnt;
      red_start  = index(_red,'//')  + 2;
      red_lgth   = index(_red,'\\')  - red_start;
      blue_start = index(_blue,'//') + 2;
      blue_lgth  = index(_blue,'\\') - blue_start;
      description_new=tranwrd(description_new
                              ,substr(_red ,red_start,
red_lgth)

,substr(_blue,blue_start,blue_lgth));
      _red =substr(_red ,red_start +red_lgth +2);
```

```
      _blue=substr(_blue,blue_start+blue_lgth+2);
   end;

   ********************************************;
   * Remove //, \\ and *;
   ********************************************;

description_new=transtrn(description_new,'//',trimn(''));

description_new=transtrn(description_new,'\\',trimn(''));
   description_new=tranwrd (description_new,'*',' ');
run;
```

Haben Sie die `transtrn()` und `trimn()` Funktionen bemerkt? Wir verwenden die Funktion `tranwrd()` nicht, weil wir `//` und `\\` durch nichts und nicht durch ein Leerzeichen ersetzen wollen. Hier ist das Endergebnis:

Tabelle 14: Unterschiede in rot und Synonyme in blau markieren (2/2)

key	ref_description	description_new
1	18-65 years old	18 to 65 yrs old yrs old
2	18-65 years old	18-65 yrs oxd
3	18-65 years old	29-65 yrs old
4	18-65 years old	29
5	18-65 years old	19 to 65 yrs olx
6	Systolic Blood Pressure is lower than x	SBP is lower than x
7	DBP is lower than x	Diastolic Blood Pressure is lower than x

8 Alternativlösungen

8.1 Alternative zur Vordergrundfarbe

Anstatt die Schriftfarbe zu ändern, können wir Zeichen auch unterstreichen oder in kursiv, fett oder mit gelber Hintergrundfarbe darstellen. Markieren oder unterstreichen eines Textes erlaubt auch nicht druckbare Zeichen wie Leerzeichen zu finden und darzustellen.

```
%let dif_start=^{style[color=red];
%let dif_start=^{style[fontstyle=italic];
%let dif_start=^{style[fontweight=bold];
%let dif_start=^{style[background=yellow];
%let dif_start=^{style[textdecoration=underline];
```

8.2 Alternative zur `compare()`-Funktion

Die `compare()`-Funktion erlaubte uns, Zonen zu finden, in denen ein oder mehrere Unterschiede auftraten. Sie können durch `sounds like` oder `rxparse` Funktionalitäten, sowie `compged()`, `complev()` oder `soundex()` Funktionen alternative Suchlösungen benutzen.

Wenn Unterschiede in Interpunktion und Leerzeichen ignoriert werden sollen, entfernen wir diese einfach mit der `compress()`-Funktion, bevor wir die Zeichenketten vergleichen.

Sollen Unterschiede ignoriert werden, so können wir die Zeichenketten mit der Funktion `upcase()` oder `lowcase()` oder mit dem `i`-Modifikator der `compare()`-Funktion einfach in Großbuchstaben oder Kleinbuchstabenumwandeln.