Data Normalization for Dummies Using SAS®

Venu Perla, Ph.D.
Clinical Programmer, Emmes Corporation, Rockville, MD 20850

SAS Certified Base Programmer for SAS 9
SAS Certified Advanced Programmer for SAS 9
SAS Certified Clinical Trials Programmer Using SAS 9
SAS Certified Statistical Business Analyst Using SAS 9: Regression and Modeling

Abstract
Life scientists often struggle to normalize non-parametric data or ignore normalization prior to data analysis. Based on statistical principles, logarithmic, square-root and arcsine transformations are commonly adopted to normalize non-parametric data for parametric tests. Several other transformations are also available for normalizing data. However, for many, identification of right transformation for non-parametric data is a tricky job. The objective of this paper is to develop a SAS program that identifies right transformation and normalize non-parametric data for regression analysis. To achieve this objective, PROC SQL, PROC TRANSREG, PROC REG, PROC UNIVARIATE, PROC STDIZE, PROC CORR, PROC SGPLOT, PROC IMPORT and PROC PRINT of SAS are utilized in this paper. Finally, SAS MACROS are developed on this code for reuse without hassles.

1. Introduction

Are you a dummy? Are you lazy? Do you really have no time? If your answer is 'Yes' to any of these questions, and if you are performing regression analysis, read this paper and apply steps mentioned here to normalize your data using SAS. If your answer is 'No' to all the questions, and if you are performing regression analysis, then you may have to apply formal statistical knowledge to normalize your data (Figure 1). Formal statistics include several transformations (logarithmic, square-root, arcsine etc) that are based on certain rules. However, SAS programming steps mentioned in this paper are not intended to replace formal statistical knowledge existing elsewhere. In other words, this paper is intended for true or conditional dummies. Parametric tests, such as an ANOVA, t-test or linear regression, can be applied to a dataset if it meets certain assumptions. One of the assumptions is that the data should be normally distributed. Parametric tests on non-normal data produce false results. The objective of this paper is to show how '6-step' protocol transforms a dataset from non-parametric to parametric for regression analysis. It is important to note that the variables used in the parametric analysis must be continuous in nature (quantitative, interval or ratio values). Discrete variables (categorical, qualitative, nominal or ordinal values) are not right candidates for parametric analysis.

A raw data on two interrelated plant metabolites (X and Y) is tested and normalized in this paper. There are 51 observations in this replicated data. Data analysis is carried out by SAS® 9.4 software with windows operating
system. Data used in this paper is imported from a sheet (XY_Data) of Microsoft® Office Excel 97-2003 file (data1.xls) (see XY_Data in Appendix). PROC IMPORT is utilized to import ‘XY_Data’ and renamed it as ‘HEALTH’ (Table 1).

```
%let path= C:\Users\Perla\Desktop\;

title "Importing data from excel";
proc import file=&path.data1.xls
   out=health replace
   dbms=xls;
   sheet=XY_Data;
   getnames=yes;
run;

title "Checking imported data";
proc print data=health;
run;
```

Table 1

<table>
<thead>
<tr>
<th>Obs</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>2</td>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>2.2</td>
<td>15.3</td>
</tr>
<tr>
<td>49</td>
<td>1.4</td>
<td>0.8</td>
</tr>
<tr>
<td>50</td>
<td>1.4</td>
<td>1.4</td>
</tr>
<tr>
<td>51</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

For importing XY_Data, macro EXCEL_IMPORT is developed on above code (see Appendix). This macro can be utilized in future for analysis of similar data by running following code:

```
%excel_import (excel_file= , excel_sheet= , dataset=);
```

2. Data Normalization

After importing data into SAS, a ‘6-step’ protocol for normalization of data for regression analysis using SAS is presented in Figure 2. Programming aspects of each step are also discussed in this section.

Step 1: Check Scatter Plot and Correlation Matrix

Relationship between X- and Y-variables can be visualized using PROC SGPLOT and PROC CORR.

```
ods graphics on;
title "Scatter plot of X and Y";
proc sgpplot data= health;
   scatter x=x y=y;
run;

title "Correlation between X and Y";
proc corr data = health;
   var x y;
run;
ods graphics off;
```

Scatter plot of X and Y indicates that there is no clear relationship between these two variables (Figure 3). Results on Pearson correlation coefficients indicate a weak correlation between X- and Y-variables (Table 2).
Above code is utilized to develop a macro, ‘SCATTER_CORR’ (See Appendix). This macro can be utilized in future for analysis of similar data by running following code:

```sas
%scatter_corr (dataset= , xvar= , yvar= );
```
**Step 2: Perform Regression Analysis and Normality Tests**

There is an indication of a weak correlation between X and Y (Pearson correlation coefficient: 0.35). Further analysis is carried out on this raw data using PROC REG and PROC UNIVARIATE. LACKFIT option of MODEL statement in PROC REG determines whether this linear model is a good fit for this replicated data or not? Residual analysis and normality tests are carried out using PROC UNIVARIATE with NORMAL option. If data is normal after 2nd step, no further steps are required to execute to normalize the data.

```r
ODS graphics on;
title "Regression analysis";
proc reg data = health plots(only)=diagnostics (unpack);
   model y = x/lackfit;
   output out = mdlres r = resid;
run;
ODS graphics off;
proc univariate data = mdlres normal;
   var resid;
run;
```

Analysis of variance indicates that LACK OF FIT for the linear model is significant *(Table 3)*. This suggests that further in-depth analysis has to be carried out on this raw data before rejecting the model.

### Table 3

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>1</td>
<td>30.39738</td>
<td>30.39738</td>
<td>6.92</td>
<td>0.0114</td>
</tr>
<tr>
<td>Error</td>
<td>49</td>
<td>180.95008</td>
<td>3.61531</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lack of Fit</td>
<td>19</td>
<td>184.35341</td>
<td>9.70281</td>
<td>112.10</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Pure Error</td>
<td>30</td>
<td>2.59667</td>
<td>0.08656</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>50</td>
<td>213.34745</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parameter estimates and adjusted $R^2$ value for the raw data are provided in **Table 4A** and **4B**, respectively. Adjusted $R^2$ value is negligible (0.11).

### Table 4A

| Variable | Label | DF | Parameter Estimate | Standard Error | t Value | Pr > |t| |
|----------|-------|----|--------------------|----------------|---------|-------|---|
| Intercept| Intercept | 1  | 0.12362            | 0.50780        | 0.24    | 0.0867|
| X        | X     | 1  | 0.93325            | 0.35490        | 2.63    | 0.0114|

### Table 4B

|                  |                  |                  |                  |
|------------------|------------------|------------------|
| Root MSE         | 1.95328          | R-Square         | 0.1237           |
| Dependent Mean   | 1.24902          | Adj R-Sq         | 0.1058           |
| Coeff Var        | 156.38516        |                  |                  |

Distribution of residuals for Y is not normal for the raw data *(Figure 4)*.
Furthermore, significant p values for four tests of normality are the true testimony of non-normal distribution of data (Table 5).

Table 5

<table>
<thead>
<tr>
<th>Test</th>
<th>Statistic</th>
<th>p Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shapiro-Wilk</td>
<td>W</td>
<td>0.293089 Pr &lt; W &lt; 0.0001</td>
</tr>
<tr>
<td>Kolmogorov-Smirnov</td>
<td>D</td>
<td>0.399871 Pr &gt; D &lt; 0.0010</td>
</tr>
<tr>
<td>Cramer-von Mises</td>
<td>W-Sq</td>
<td>2.459645 Pr &gt; W-Sq &lt; 0.0050</td>
</tr>
<tr>
<td>Anderson-Darling</td>
<td>A-Sq</td>
<td>12.24508 Pr &gt; A-Sq &lt; 0.0050</td>
</tr>
</tbody>
</table>

Above code is utilized to develop a macro ‘REG_NORMALITY’ (See Appendix). This macro can be utilized in future for analysis of similar data by running following code:

\%reg_normality (dataset=health, xvar=x, yvar=y);

Step 3: Transform Data into Non-zero and Non-negative Data

Box-Cox power transformation can be adopted to normalize this raw data. Data should be converted to non-zero and non-negative values before testing for Box-Cox power transformation. Following code transforms X- and Y-variables into non-zero and/or non-negative variables only when '0' or negative values are encountered in the data.

PROC SQL is used to transform X- and Y-variable data into non-zero and non-negative data. Table HEALTH_COX is created from dataset HEALTH in this procedure. Proc SQL reproduced original data as there are no zeros and no negative values (Table 6).

```
title "Transforming X and Y values into non-zero and non-negative values";
proc sql;
  create table health_cox as
  select case
    when min(x) <= 0 then (- (min(x)) + x + 1)
    else x end as X,
    case
    when min(y) <= 0 then (- (min(y)) + y + 1)
    else y end as Y
  from health;
quit;
proc print data=health_cox;
```
run;

Macro ‘TRANSFORM_ZERO_NEG’ is developed for above PROC SQL code (See Appendix). This macro can be invoked in future by following statement:

```sas
%transform_zero_neg (dataset= ,xvar= ,yvar= ,pre_trans_dataset= );
```

**Step 4: Perform Box-Cox Power Transformation**

Box-Cox power transformation on non-zero and non-negative data is performed using PROC TRANSREG with ODS GRAPHICS on.

```sas
title "Box-Cox power transformation: Identification of right exponent (Lambda)";
ods graphics on;
proc transreg data= health_cox;
   model boxcox(y) = identity(x);
run;
ods graphics off;
```

Above code generated Box-Cox analysis for Y (Figure 5). Selected lambda (-0.75 at 95% CI) is the exponent to be used to transform the data into normal shape.

In order to get convenient lambda value, above SAS code is executed without ODS GRAPHICS statement.

```sas
proc transreg data = health_cox;
   model boxcox(y)=identity(x);
run;
```
This code generated best lambda, lambda with 95% confidence interval, and convenient lambda (Table 7). Convenient lambda is used for transforming Y-variable in this analysis.

**Table 7**

<table>
<thead>
<tr>
<th>Lambda</th>
<th>R-Square</th>
<th>Log Like</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.00</td>
<td>0.54</td>
<td>58.758 *</td>
</tr>
<tr>
<td>-0.75</td>
<td>0.55</td>
<td>59.996 &lt;</td>
</tr>
<tr>
<td>-0.50</td>
<td>0.55</td>
<td>58.952 *</td>
</tr>
</tbody>
</table>

Macro ‘BOX_COX_LAMBDA’ is developed on above code (See Appendix). This macro can be utilized in future for analysis of similar data by running following code:

```
%box_cox_lambda (pre_trans_dataset=, xvar=, yvar=);
```

PROC SQL program is used to transform Y-variable. Code for common convenient lambda values (-2, -1, -0.5, 0, 0.5, 1 and 2); respective Y-transformations (1/Y^2, 1/Y, 1/sqrt(Y), log(Y), sqrt(Y), Y and Y^2); and respective transformed-Y variable names (neg_2_y, neg_1_y, neg_half_y, zero_y, half_y, one_y, and two_y) are incorporated in the program.

```sql
title "Transformation of Y-values with convenient lambda";
proc sql;
create table health_trans as
select x, y,
  1/(y**2) as neg_2_y,
  1/(y**1) as neg_1_y,
  1/(sqrt(y)) as neg_half_y,
  log(y) as zero_y,
  sqrt(y) as half_y,
  y**1 as one_y,
  y**2 as two_y
from health_cox;
quit;
```

PROC SQL generated ‘HEALTH_TRANS’ table (Table 8). ‘neg_1_y’ is the corresponding transformed Y-variable for the convenient lambda -1. This ‘neg_1_y’ variable is used for further analysis.
Macro ‘TRANSFORM_LAMBDA’ is defined on above PROC SQL code (See Appendix). This macro can be utilized in future for analysis of similar data by running following code:

```sas
%transform_lambda (pre_trans_dataset= , xvar= , yvar= , trans_dataset= );
```

**Step 5: Standardize X-variable**

After transformation of Y-variable, in order to obtain meaningful Y-intercept, X-variable is standardized using PROC STDIZE. Dataset ‘HEALTH2’ is generated from table ‘HEALTH_TRANS’ in this procedure. OPREFIX option is used to prefix the original X-variable name with the word, ‘Unstdized_’. On the other hand, standardized X-values are stored under X.

```sas
title "Standardized X-variable after Y-transformation";
proc stdize data=health_trans
  oprefix=Unstdized_
  method=mean
  out=health2;
  var x;
run;
```

```sas
proc print data=health2;
run;
```

Generated dataset ‘HEALTH2’ is shown below with standardized X-variable in the last column as X (Table 9).

**Table 9**

<table>
<thead>
<tr>
<th>Obs</th>
<th>Unstdized_X</th>
<th>neg_2_y</th>
<th>neg_1_y</th>
<th>neg_half_y</th>
<th>zero_y</th>
<th>half_y</th>
<th>one_y</th>
<th>two_y</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.4</td>
<td>0.4</td>
<td>0.62500</td>
<td>2.50000</td>
<td>1.58114</td>
<td>-0.91629</td>
<td>0.63246</td>
<td>0.4</td>
<td>0.16</td>
</tr>
<tr>
<td>2</td>
<td>0.6</td>
<td>0.5</td>
<td>4.00000</td>
<td>2.00000</td>
<td>1.41421</td>
<td>-0.69315</td>
<td>0.70711</td>
<td>0.5</td>
<td>0.25</td>
</tr>
<tr>
<td>3</td>
<td>2.2</td>
<td>15.3</td>
<td>0.00427</td>
<td>0.00636</td>
<td>0.25565</td>
<td>2.72785</td>
<td>3.91152</td>
<td>15.3</td>
<td>234.09</td>
</tr>
</tbody>
</table>

Macro ‘STDIZE_X’ is defined on above code (See Appendix). It can be invoked in future by calling following statement:

```sas
%stdize_x (trans_dataset= , trans_stdize_dataset= , xvar= );
```

**Step 6: Perform Regression Analysis and Normality Tests**

Regression analysis and normality tests are again performed on the transformed and standardized dataset ‘HEALTH2’ by calling previously defined macro ‘REG_NORMALITY’. Variable X is the standardized X, and ‘neg_1_y’ is the transformed Y.

```sas
%reg_normality (dataset=health2, xvar=x, yvar=neg_1_y);
```

With transformed data, LACK OF FIT for linear model is turned out to be non-significant, which indicates that the linear model is acceptable for X and Y (Table 10). Parameter estimates for intercept and X are significant (Table 11). As compared to the raw data, adjusted R² value with transformed data is improved from 0.11 to 0.53 (Table 12).
Other results indicate that transformed data is normally distributed (Figure 6; Table 13). Non-significant p-value with Kolmogorov-Smirnov normality test further confirms that data is normally distributed (Table 13).

### Table 10

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>1</td>
<td>6.00630</td>
<td>0.00630</td>
<td>57.29</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>49</td>
<td>6.84719</td>
<td>0.13974</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lack of Fit</td>
<td>19</td>
<td>3.18401</td>
<td>0.16758</td>
<td>1.37</td>
<td>0.2135</td>
</tr>
<tr>
<td>Pure Error</td>
<td>30</td>
<td>3.66317</td>
<td>0.12211</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>50</td>
<td>14.85349</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 11

| Variable   | DF  | Parameter Estimate | Standard Error | t Value | Pr > |t| |
|------------|-----|--------------------|----------------|---------|------|---|
| Intercept  | 1   | 1.24012            | 0.05234        | 23.69   | <.0001 |
| x          | 1   | -0.51397           | 0.06790        | -7.57   | <.0001 |

### Table 12

| Root MSE  | 0.37382   |
| R-Square  | 0.5390    |
| Dependent Mean | 1.24012   |
| Adj R-Sq  | 0.5296    |
| Coeff Var | 30.14346  |

### Figure 4

Distribution of Residuals for Y

### Figure 6

Distribution of Residuals for neg_1_y

### Table 13

<table>
<thead>
<tr>
<th>Test</th>
<th>Statistic</th>
<th>p Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shapiro-Wilk</td>
<td>W</td>
<td>0.948413</td>
</tr>
<tr>
<td>Kolmogorov-Smirnov</td>
<td>D</td>
<td>0.106341</td>
</tr>
<tr>
<td>Cramer-von Mises</td>
<td>W-Sq</td>
<td>0.131357</td>
</tr>
<tr>
<td>Anderson-Darling</td>
<td>A-Sq</td>
<td>0.92551</td>
</tr>
</tbody>
</table>
3. Normalization Eliminates Misleading Results
Negligible positive correlation exists between X and Y in raw data (Adjusted $R^2$: 0.11). After normalization, there is a moderate positive correlation between X and Y (Adjusted $R^2$: 0.53). In other words, normalization eliminates misleading results.

Raw data (non-parametric):
Y = 0.124 + 0.933X (Adjusted $R^2$: 0.11)

Normalized data:
Y = 1.240 – 0.514X (Adjusted $R^2$: 0.53)

4. Limitations and Solutions
Outliers in a data may drastically affect normalization. Three out of four tests of normality are still significant in this analysis (Table 13). It indicates that there is a room for further improvement of data with respect to normalization. In general, non-parametric nature of data after step 6 indicates presence of outliers in the dataset. There is at least one outlier and leverage observation that is influencing the normal distribution here (Figure 7). Techniques to isolate and rescue outliers while normalizing the data were presented at the Ohio SAS Users Group Conference (Perla, 2015A), and at the Philadelphia Area SAS Users Group Fall Meeting (Perla, 2015B) (Figure 8 and 9). Another potential limitation for ‘6-step’ protocol is the field of study. Perhaps, ‘6-step’ protocol can be applied in any field. However, sometimes, it may be more meaningful to adopt a transformation that is commonly used in that field of study.

5. Conclusions
In summary, misleading results are produced if parametric tests, such as t-test, ANOVA or linear regression, are applied on non-parametric data. A ‘6-step’ protocol discussed in this paper is a good option for normalizing data for
regression analysis. In the real world, outliers are the major limitation while normalizing the data. I have recently explained a technique to isolate and rescue outliers while normalizing the data. Refer Perla (2005A and B) for more details and macro definitions.

References

Acknowledgments
I would like to thank the organizers for giving me an opportunity to present this paper at the Philadelphia SAS® Users Group Winter Meeting on March 16, 2016 at the Tuttleman Center, Philadelphia University, Philadelphia, PA, USA.

Trademark Citations
SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Author Biography
Venu Perla, Ph.D. is a SAS Certified Advanced Programmer, Clinical Programmer and Statistical Business Analyst for SAS 9. Dr. Perla is also a biomedical researcher with about 14 years of research and teaching experience in an academic environment. He served the Purdue University, Oregon Health & Science University, Colorado State University, West Virginia State University R&D Corporation, Kerala Agricultural University (India) and Mangalayatan University (India) at different capacities. Dr. Perla has published 15 scientific papers and 2 book chapters, obtained 1 international patent on orthopaedic implant device, gave 9 talks and presented 18 posters at national and international scientific conferences in his professional career. Dr. Perla was invited to serve as an editorial board member for several national and international scientific journals. He was trained in clinical trials and clinical data management. Currently, he is actively employing SAS® programming techniques in clinical data analysis.

Contact Information
Phone (Cell): (304) 545-5705
Email: venuperla@yahoo.com
vperla@emmes.com
LinkedIn: https://www.linkedin.com/pub/venu-perla/2a/700/468
Appendix


<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td>2.2</td>
<td>15.3</td>
</tr>
<tr>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>0.7</td>
<td>0.6</td>
</tr>
<tr>
<td>2.5</td>
<td>1.1</td>
</tr>
<tr>
<td>0.4</td>
<td>0.5</td>
</tr>
<tr>
<td>0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>1.3</td>
<td>0.9</td>
</tr>
<tr>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>1.8</td>
<td>1.6</td>
</tr>
<tr>
<td>0.5</td>
<td>1.8</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>0.3</td>
<td>0.7</td>
</tr>
<tr>
<td>1.4</td>
<td>0.9</td>
</tr>
<tr>
<td>0.8</td>
<td>0.6</td>
</tr>
<tr>
<td>1.3</td>
<td>1</td>
</tr>
<tr>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td>1.2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2.1</td>
</tr>
<tr>
<td>0.7</td>
<td>0.6</td>
</tr>
<tr>
<td>1.3</td>
<td>1.1</td>
</tr>
<tr>
<td>1.1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1.3</td>
</tr>
<tr>
<td>0.6</td>
<td>0.7</td>
</tr>
<tr>
<td>2.1</td>
<td>1.7</td>
</tr>
<tr>
<td>1.8</td>
<td>1.4</td>
</tr>
<tr>
<td>1.2</td>
<td>0.8</td>
</tr>
<tr>
<td>1</td>
<td>0.7</td>
</tr>
<tr>
<td>2.1</td>
<td>1.5</td>
</tr>
<tr>
<td>1.4</td>
<td>1</td>
</tr>
<tr>
<td>0.7</td>
<td>0.8</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>0.9</td>
<td>0.7</td>
</tr>
<tr>
<td>1.2</td>
<td>0.5</td>
</tr>
<tr>
<td>1.1</td>
<td>0.7</td>
</tr>
<tr>
<td>2.5</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0.7</td>
</tr>
<tr>
<td>0.9</td>
<td>0.8</td>
</tr>
<tr>
<td>3</td>
<td>2.7</td>
</tr>
<tr>
<td>4.2</td>
<td>1.5</td>
</tr>
<tr>
<td>0.9</td>
<td>1</td>
</tr>
<tr>
<td>1.9</td>
<td>1.6</td>
</tr>
<tr>
<td>1</td>
<td>0.8</td>
</tr>
<tr>
<td>1.2</td>
<td>0.7</td>
</tr>
<tr>
<td>0.8</td>
<td>0.7</td>
</tr>
<tr>
<td>1.4</td>
<td>0.8</td>
</tr>
<tr>
<td>1.4</td>
<td>1.4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Macro ‘EXCEL_IMPORT’:
Macro ‘EXCEL_IMPORT’ is defined below for importing Excel files. Where, ‘EXCEL_FILE=’ is name of the excel file to be used; ‘EXCEL_SHEET=’ is name of the excel sheet to be imported; and ‘DATASET=’ is name of the output dataset. File extension and DBMS statement in the code may be modified according to the Excel version used.

```sas
%macro excel_import (excel_file= , excel_sheet= , dataset= );
   title "Importing data from excel";
   proc import file="&path.&excel_file..xls"
      out=&dataset replace
dbms=xls;
   sheet=&excel_sheet;
   getnames=yes;
   run;
   title "Dataset from imported excel data";
   proc print data=&dataset;
   run;
%mend excel_import;
```

This macro can be invoked by calling following code for this paper:

```sas
%excel_import (excel_file=data1, excel_sheet=XY_Data, dataset=health);
```

Macro ‘SCATTER_CORR’:
Macro ‘SCATTER_CORR’ is defined below. Where, ‘DATASET=’ is name of the dataset to be used for analysis; and ‘XVAR=’ and ‘YVAR=’ are the names of the X- and Y-variables, respectively.

```sas
%macro scatter_corr (dataset= , xvar= , yvar= );
   ods graphics on;
   title "Scatter plot of &xvar and &yvar";
   proc sgplot data= &dataset;
      scatter x=&xvar y=&yvar;
   run;
   title "Correlation between &xvar and &yvar";
   proc corr data = &dataset;
      var &xvar &yvar;
   run;
   ods graphics off;
%mend scatter_corr;
```

This macro can be invoked by following statement for this paper:

```sas
%scatter_corr (dataset=health, xvar=x, yvar=y);
```

Macro ‘REG_NORMALITY’:
Macro ‘REG_NORMALITY’ is defined below for regression analysis and normality tests. Where, ‘DATASET=’ is name of the dataset to be used for analysis; and ‘XVAR=’ and ‘YVAR=’ are names of the X- and Y-variables, respectively.

```sas
%macro reg_normality (dataset= ,xvar= ,yvar= );
   ODS graphics on;
   title "Regression analysis: Dataset &dataset";
   proc reg data = &dataset plots(only)=diagnostics (unpack);
      model &yvar = &xvar/lackfit;
      output out =mdlres r=resid;
   run;
```

This macro can be invoked by calling following code for this paper:

```sas
%reg_normality (dataset=health, xvar=x, yvar=y);
```
proc univariate data= mdlres normal;
   var resid;
run;
ods graphics off;
%mend reg_normality;

This macro can be invoked by following statement for this paper:

%reg_normality (dataset=health, xvar=x, yvar=y);

Macro ‘TRANSFORM_ZERO_NEG’:
Macro ‘TRANSFORM_ZERO_NEG’ is defined below. Where, ‘DATASET=’ is the name of the input dataset to be used for transforming X- and Y-values; ‘XVAR=’ and ‘YVAR=’ are names of the X- and Y-variables to be transformed, respectively; and ‘PRE_TRANS_DATASET=’ is name of the output dataset to be created with transformed X- and Y-variables.

%macro transform_zero_neg (dataset= ,xvar= ,yvar= ,pre_trans_dataset=);
   title "Transforming &xvar and &yvar values into non-zero and non-negative values";
   proc sql;
      create table &pre_trans_dataset as
         select case
            when min(&xvar) <= 0 then (-min(&xvar))+&xvar+1
            else &xvar
            end as &xvar,
            case
            when min(&yvar) <= 0 then (-min(&yvar))+&yvar+1
            else &yvar
            end as &yvar
         from &dataset;
      quit;

   proc print data=&pre_trans_dataset;
run;
%mend transform_zero_neg;

This macro can be invoked by following statement for this paper:

%transform_zero_neg
   (dataset=health,xvar=x,yvar=y,pre_trans_dataset=health_cox);

Macro ‘BOX_COX_LAMBDA’:
Macro ‘BOX_COX_LAMBDA’ is defined below. Where, ‘PRE_TRANS_DATASET=’ is name of the input dataset with non-zero and non-negative values; and ‘XVAR=’ and ‘YVAR=’ are names of the X- and Y-variables, respectively.

%macro box_cox_lambda (pre_trans_dataset= ,xvar= ,yvar= );
   title "Box-Cox power transformation: Identification of right exponent (Lambda)";
   ods graphics on;
   proc transreg data= &pre_trans_dataset;
      model boxcox(&yvar) = identity(&xvar);
   run;
   ods graphics off;

   proc transreg data = &pre_trans_dataset;
      model boxcox(&yvar)=identity(&xvar);
   run;
%mend box_cox_lambda;
This macro can be invoked by following statement for this paper:

```sas
%box_cox_lambda (pre_trans_dataset=health_cox, xvar=x, yvar=y);
```

Macro ‘TRANSFORM_LAMBDA’:
Macro ‘TRANSFORM_LAMBDA’ is defined below. Where, ‘PRE_TRANS_DATASET=’ is name of the input dataset with non-zero and non-negative X- and Y-values; ‘XVAR=’ and ‘YVAR=’ are names of the X- and Y-variables, respectively; and ‘TRANS_DATASET=’ is name of the output dataset with transformed data.

```sas
%macro transform_lambda (pre_trans_dataset=, xvar=, yvar=, trans_dataset=);
  title "Transformation of &yvar.-values with convenient lambda";
  proc sql;
    create table &trans_dataset as
      select &xvar, &yvar,
        1/(&yvar**2) as neg_2_&yvar,
        1/(&yvar**1) as neg_1_&yvar,
        1/(sqrt(&yvar)) as neg_half_&yvar,
        log(&yvar) as zero_&yvar,
        sqrt(&yvar) as half_&yvar,
        &yvar**1 as one_&yvar,
        &yvar**2 as two_&yvar
      from &pre_trans_dataset;
  quit;
  proc print data=&trans_dataset;
  run;
%mend transform_lambda;
```

This macro can be invoked by following statement for this paper:

```sas
%transform_lambda (pre_trans_dataset=health_cox, xvar=x, yvar=y, trans_dataset=health_trans);
```

Macro ‘STDIZE_X’:
Macro ‘STDIZE_X’ is defined below. Where, ‘TRANS_DATASET=’ is name of the input dataset; ‘TRANS_STDIZE_DATASET=’ is name of the output dataset; and ‘XVAR=’ is name of the X-variable to be standardized.

```sas
%macro stdize_x (trans_dataset=, trans_stdize_dataset=, xvar=);
  title "Standardized &xvar.-variable after Y-transformation";
  proc stdize data=&trans_dataset
oprefix=Unstdized_
method=mean
out=&trans_stdize_dataset;
  var &xvar;
run;
proc print data=&trans_stdize_dataset;
run;
%mend stdize_x;
```

This macro can be invoked by following statement for this paper:

```sas
%stdize_x (trans_dataset=health_trans, trans_stdize_dataset=health2, xvar=x);
```