Paper 158-2023

Univariate Outlier Detection Using SAS

Fan Yang, Johnson and Johnson Vision

ABSTRACT

Outlier detection is an important task for many data analysis projects. It can help identify unusual observations in a dataset that may affect the statistical or analytical method used or have a large influence on the analysis results. This paper will introduce five statistical methods to detect outliers in a univariate dataset. These five methods range from a simple use of the mean and standard deviation statistics to the use of more advanced statistical techniques such as Robust Regression and the Medcouple statistic. This paper also covers some programming aspects about how to implement these five methods in SAS. A custom SAS macro is built using the SAS/IML language to implement an advanced 'FAST' algorithm to calculate the Medcouple statistic for outlier detection that seems not to have been done yet in the SAS community. Finally, the sensitivity of outlier detection for these five methods are compared in a simulation study with some recommendations drawn. The target audience for this paper are those SAS practitioners or data analysts who may want to refresh or enrich their knowledge about the statistical methods for outlier detection and who may benefit from learning how to apply these methods in SAS.

INTRODUCTION

An outlier is an unusual observation in a dataset whose value is very different from others. A presence of outliers is very common in a real-life data analysis with many possible causes: a failed sample, a special-cause variation, a measurement error, existence of a subpopulation, a skewed distribution, or pure randomness.

Detection of outliers is very important in many data analysis projects. Outliers in a dataset may require additional investigation to understand why their values are different from others. In running a statistical analysis, outliers may cause violation of assumptions that are crucial for the chosen statistical method. In such a case, detection of outliers may help evaluate if the statistical method is valid for its intended use. Finally, in some data analysis tasks, a dataset that contains outliers may need to be detected and given a separate analysis due to their large influence on the analysis results. For instance, in running a statistical process control, outliers may need to be detected and removed from the calculation of the control limits so that the control limit values are not inflated by the outliers (Weeraratne, 2016).

Several statistical methods are developed to detect outliers in a univariate dataset (i.e. a dataset contains one continuous variable). They may fall under two general categories: to detect outliers by running a hypothesis test or to detect outliers based on a set of outlier criteria. The first category includes, for examples, Grubbs' Test (Grubbs, 1969), Dixon's Q Test (Dixon, 1951), Generalized Extreme Studentized Deviate (ESD) Test (Rosner, 1983). One disadvantage of these methods is that these hypothesis tests typically can only detect one outlier at a time. So, in order to find all outliers in a dataset, one may have to run such a test sequentially. The methods in the second category are more efficient because they calculate a set of outlier criteria and all outliers in a dataset can be detected all at once against the criteria.

This article introduces five popular outlier detection methods in the second category: Z-Score, Median Absolute Deviation (MAD), Robust Regression, Boxplot, and Medcouple Adjusted Boxplot. The derivation of the outlier criteria under each method will be presented followed by a discussion of their SAS implementation. Their performances in term of outlier sensitivity are compared based on a simulation study.

OUTLIER DETECTION METHODS

Z-SCORE METHOD

Consider a univariate dataset (X1, X2, …, Xn) has a mean $\overline{X}$ and standard deviation S. The Z-Score

statistic is defined as the number of standard deviations above or below the mean value, or

$$Z_i = \frac{X_i - \overline{X}}{S}$$

Given a user specified k parameter, the datapoint Xi will be considered an outlier if

$$|Z_i| > k$$

The equivalent outlier criteria under the Z-Score method are:

$$\text{Lower Cutoff Value for Outliers: } \overline{X} - k \times S$$

$$\text{Upper Cutoff Value for Outliers: } \overline{X} + k \times S$$

In a special case where S = 0 (meaning all the data values are the same), the Z-Score statistic is undefined, and there will be no outlier detection in the dataset.

The Z-Score method for the outlier detection is based on the well-known sigma rule for a Normal Distribution, which states that a Normal Distribution covers 95% or 99.7% of the population within ±2 sigma or ±3 sigma distance from the mean, respectively. So, a common choice of k=3 will help identify outliers that are outside the 99.7% of the range of a Normal Distribution. However, when a univariate dataset follows a skewed distribution, the Z-Score method may not be appropriate, because its outlier criteria are symmetric which will be over-sensitive on one side, but less-sensitive on the other side. Another drawback for the Z-Score method is that the estimate of the mean and standard deviation may be inflated by the existence of outliers in the dataset. This may reduce the Z-Score method's sensitivity of finding the outliers.

The SAS implementation of the Z-Score method is very simple, as it only involves the calculation of the summary statistics mean and standard deviation. Both statistics can be easily obtained using the standard SAS procedure PROC MEANS or PROC UNIVARIATE (see **Output 1**).

```
proc univariate data=input_dataset noprint;
var x;
output out=stat mean=mean std=stdev;
run;
```

Output 1. SAS PROC UNIVARIATE to Compute Mean and Standard Deviation for Z-Score Method

MEDIAN ABSOLUTE DEVIATION METHOD

The absolute deviation of each data point to the median value of a univariate dataset is defined as

$$d_i = |X_i - M|$$

The median value of d_1, d_2, …, d_n is called Median Absolute Deviation, or MAD. At a large sample size,

$$\text{MAD} \approx 0.6745 \times \sigma$$

where σ is the standard deviation (Rousseeuw & Croux, 1993). The test statistic using the MAD statistic for outlier detection is

$$\text{MAD Statistic for } X_i = 0.6745 \times (X_i - M)/MAD$$

So given a user specified k parameter, the datapoint X_i will be considered an outlier if

$$|\text{MAD Statistic for } X_i| > k$$

The equivalent outlier criteria under the MAD method are (Leys, Ley, Klein, Bernard, & Licata, 2013):

$$\text{Lower Cutoff Value for Outliers: } M - k \times MAD/0.6745$$

$$\text{Upper Cutoff Value for Outliers: } M + k \times MAD/0.6745$$

The MAD method mimics the Z-Score method, but uses the median statistics instead of mean and standard deviation. The MAD method also gives a set of symmetric outlier criteria which restrict its good use to a symmetric distribution. Also, another limitation for the use of the MAD method is on the case where 50% of more of the datasets are equal. In such a scenario, the MAD statistic will be equal to zero, which causes a failure of the MAD method for outlier detection.

The median and MAD statistic can be directly calculated in SAS using PROC MEANS or PROC UNIVARIATE with the keyword "MEDIAN" and "MAD" in the OUTPUT (see **Output 2**).

```
proc univariate data=input_dataset noprint;
var x;
output out=stat median=median mad=mad;
run;
```

Output 2. SAS PROC UNIVARIATE to Compute Median and MAD Statistics for MAD Method

ROBUST REGRESSION METHOD

Consider a simple regression model for a univariate dataset (X_1, X_2, …, X_n):

$$X_i = \mu + e_i$$

where μ is the mean parameter for the population and e_i are the residuals that are assumed to be independently and normally distributed with a constant standard deviation σ.

The robust regression using the least trimmed square (LTS) method is an alternative to the regular least square regression (Draper, 1988). The robust regression estimates the unknown parameter μ by

$$\hat{\mu}_{LTS} = arg\ min\ Q_{LTS}(\mu)$$

where

$$Q_{LTS}(\mu) = \sum_{i=1}^{h} r_{(i)}^2$$

$r_{(1)}^2 \le r_{(2)}^2 \le \ldots \le r_{(n)}^2$ are the ordered squared residuals $r_{(i)}^2 = (X_i - \mu)^2, i = 1, 2, \ldots, n$, and $h = (3n + 2)/4$ (the default value used by the SAS PROC ROBUSTREG procedure). The outcome of robust regression using LTS method is a pair of two estimates: $\hat{\mu}_{RobustReg}$ (called Location) and $\hat{\sigma}_{RobustReg}$ (called Scale). The robust regression residual for each data point is calculated as

$$r_i = (X_i - Location)/Scale = (X_i - \hat{\mu}_{RobustReg})/\hat{\sigma}_{RobustReg}$$

Given a user specified k parameter, the datapoint X_i will be considered an outlier if

$$|r_i| > k$$

The equivalent upper and lower outlier cutoff values for outlier determination under the LTS Robust Regression method are (Rousseeuw & Leroy, 2005):

$$\text{Lower Cutoff Value for Outliers: } Location - k \times Scale$$

$$\text{Upper Cutoff Value for Outliers: } Location + k \times Scale$$

Because the Robust Regression estimates the mean (μ) and standard deviation (σ) parameters using the order statistics of the residuals, it is considered a robust method against the presence of the outliers in the dataset. This makes the LTS method more sensitive to detect outliers than the Z-Score method.

The Robust Method for Outlier Detection can be implemented in SAS using the PROC ROBUSTREG procedure (Chen, 2002). The “method=LTS” is specified to use the least trimmed square of the residuals. The OUTPUT statement with the keyword “OUTLIER” will generate a variable to indicate outliers (based on default k=3 or to use CUTOFF option to specify any k value after the MODEL statement). The estimated location and scale parameters are saved in the ODS table “LTSLocationScale” which can be used to calculate the outlier criteria values (see **Output 3**).

```
proc robustreg data=input_data method=LTS;
model x = ;
output out=output_data outlier=outlier_indicator;
run;
```

Output 3. SAS PROC ROBUSTREG for Robust Regression Outlier Detection Method

BOXPLOT METHOD

A Boxplot (also known as Box-Whisker Plot) is popular graphical tool to visualize the data distribution based on five order statistics (the minimum, first quartile [Q1], median [Q2], third quartile [Q3] and the maximum). It is also useful to detect and visualize outliers in a dataset.

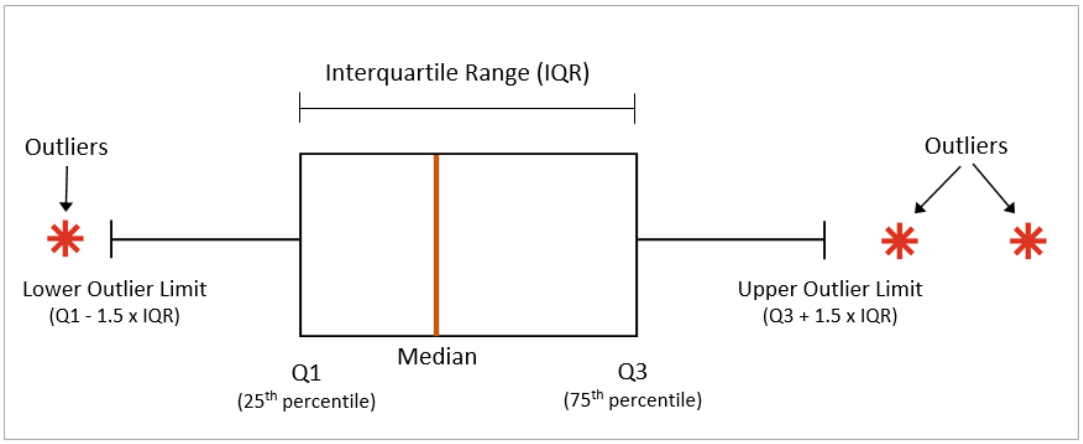

Figure 1. Illustration of a Boxplot

The outliers are defined as any observations that are above Q3 + 1.5 x IQR, or below Q1 – 1.5 IQR, where IQR is the interquartile range (Q3 – Q1). So under the Boxplot method, the outlier criteria are:

$$\text{Lower Cutoff Value for Outliers: } Q1 - 1.5 \times IQR$$

$$\text{Upper Cutoff Value for Outliers: } Q3 + 1.5 \times IQR$$

The Q1, Q3, and IQR statistics used to construct a Boxplot is less sensitive to outliers, which makes the Boxplot a robust method for outlier detection against the outliers themselves. Furthermore, because the quartile values of Q1 and Q3 are not symmetric around the median Q2 for a skewed distribution, the outlier criteria are also adjusted accordingly for the lower and upper side. This could improve the outlier detection for a skewed distribution.

The Boxplot for a dataset can be generated using the HBOX (horizontal boxplot) or VBOX (vertical boxplot) statement in the SAS PROC SGPLOT procedure. The outlier criteria can also be calculated numerically from the quartile statistics Q1 and Q3 which can be obtained from PROC MEANS or PROC UNIVARIATE (see **Output 4**).

```
proc univariate data=input_data noprint pctldef=4;
var x;
output out=stat q1=q1 q3=q3 qrange=iqr;
run;
```

Output 3. SAS PROC UNIVARIATE to Compute Q1, Q3, IQR Statistics for Boxplot Method

Note that the optional argument "PETLDEF=" in PROC UNIVARIATE is used to specify one of the five definitions to compute the quartile statistics (percentiles). The default choice is PETLDEF=5, while other choices from 1 to 4 can be used to match the quartile calculations from other software packages (e.g. PETLDEF=3 to match the "interpolated_invert_cdf" method in Python's numpy.percentile function, PETLDEF=4 to match the "PERCENTILE.EXC" function in Microsoft EXCEL, the "weibull" method in Python numpy.percentile function, and the MINITAB software).

MEDCOUPLE ADJUSTED BOXPLOT METHOD

The Medcouple (MC) value is a robust statistic ranged between -1 and 1 that measures how skewed the univariate distribution is (Brys, Hubert, & Struyf, 2004). The scenarios of MC = 0, MC > 0, and MC < 0 indicates a symmetric, right-skewed, and left-skewed distribution.

Given a univariate dataset (X_1, X_2, …, X_n), its Medcouple value statistic can be calculated by the following algorithm:

Step 1. Sort the data from the largest to the smallest: $X_1 \geq X_2 \geq \ldots \geq X_n$ and obtain the median value of the dataset X_m.

Step 2. Split the dataset to two subsets: $X^+ = \{X_i \mid X_i \geq X_m\}$ and $X^- = \{X_j \mid X_j \leq X_m\}$. Let p and q denote the cardinality of the two subsets: $p = |X^+|$ and $q = |X^-|$.

Step 3. Calculate the Medcouple kernel value for each pair of the values from X^+ and X^-.

$$h(X_i^+, X_j^-) = \begin{cases} \dfrac{(X_i^+ - X_m) - (X_m - X_j^-)}{X_i^+ - X_j^-} & \text{if } X_i^+ > X_j^- \\ \text{Sign}(p - 1 - i - j) & \text{if } X_i^+ = X_j^- \end{cases}$$

Step 4. The value of the Medcouple statistic is the median of all $h(X_i^+, X_j^-)$.

The Medcouple Adjusted Boxplot modifies the length of the upper and lower whiskers based on the calculated Medcouple statistic so that the upper whisker becomes longer than the lower whisker for a right skewed distribution, and the upper whisker is shorter than the lower whisker for a left-skewed distribution (Hubert & Vandervieren, 2008). This introduces the following outlier criteria under the Medcouple Adjusted Boxplot:

Case 1. Medcouple ≥ 0

$$\text{Lower Cutoff Value for Outliers: } Q1 - 1.5\ IQR\ e^{-4\ MC}$$

$$\text{Upper Cutoff Value for Outliers: } Q1 + 1.5\ IQR\ e^{3\ MC}$$

Case 2. Medcouple < 0

$$\text{Lower Cutoff Value for Outliers: } Q1 - 1.5\ IQR\ e^{-3\ MC}$$

$$\text{Upper Cutoff Value for Outliers: } Q1 + 1.5\ IQR\ e^{4\ MC}$$

Note that when the distribution is symmetric, MC=0 and the outlier criteria under the Medcouple Adjusted Boxplot would be equal to those under the regular Boxplot. In other words, the regular Boxplot can be considered a special case of Medcouple Adjusted Boxplot for a symmetric distribution.

Recall that the lower and upper outlier criteria for a regular Boxplot is asymmetric for a skewed distribution because Q1 and Q3 are not symmetric around the median. However, for a regular Boxplot, the length of the whiskers used for the lower and upper side is the same as 1.5 x IQR. The Medcouple Adjusted Boxplot further adjusts the length of the lower and upper whiskers and the outlier criteria according to the Medcouple statistic, which reflects the skewness of the distribution. In this sense, the Medcouple Adjusted Boxplot is a better outlier detection method for a skewed distribution than the regular Boxplot.

The calculation of the Medcouple statistic can be conducted following the 4 steps illustrated above, which involves (1) sorting and calculating the median of the original dataset of n values, which has a complexity of O(nLogn), (2) calculating the kernel values $h(X_i^+, X_j^-)$ for the 'p x q' pairs, which has a complexity of $O(n^2)$, and (3) calculating the median value of the 'p x q' pairs of the kernel values, which has a complexity of $O(n^2)$ too. So, the overall complexity of the 4-step algorithm is $O(n^2)$.

Knowing that the kernel $h(X_i^+, X_j^-)$ is a non-decreasing function for each variable X_i^+ and X_j^-, an improved algorithm has developed to reduce the complexity to O(nLogn) (Johnson & Mizoguchi, 1978). This will involve an adaptive search of the median of $h(X_i^+, X_j^-)$ without evaluating every pair of $h(X_i^+, X_j^-)$. A technique called weighted median is used to achieve this.

Both the 4-step algorithm (called the Naïve Algorithm) and the improved one (called the Fast Algorithm) can be implemented in SAS. The appendix section of this paper includes a custom SAS macro that can calculate the Medcouple statistic using either method. Because the Fast Algorithm involves many vector or matrix-based calculations, the macro was built with the use of SAS/IML language. Figure 2 compares the efficiency of the two algorithms (Naïve vs Fast) to compute the Medcouple statistic for datasets with increasing sizes. The running time for the Naïve and Fast algorithm approximately follow a quadratic and linear trend in correspondence to their $O(n^2)$ and O(nLogn) complexity, respectively. The Fast algorithm can significantly reduce the computing time over the Naïve algorithm when a dataset has a size n > 2000.

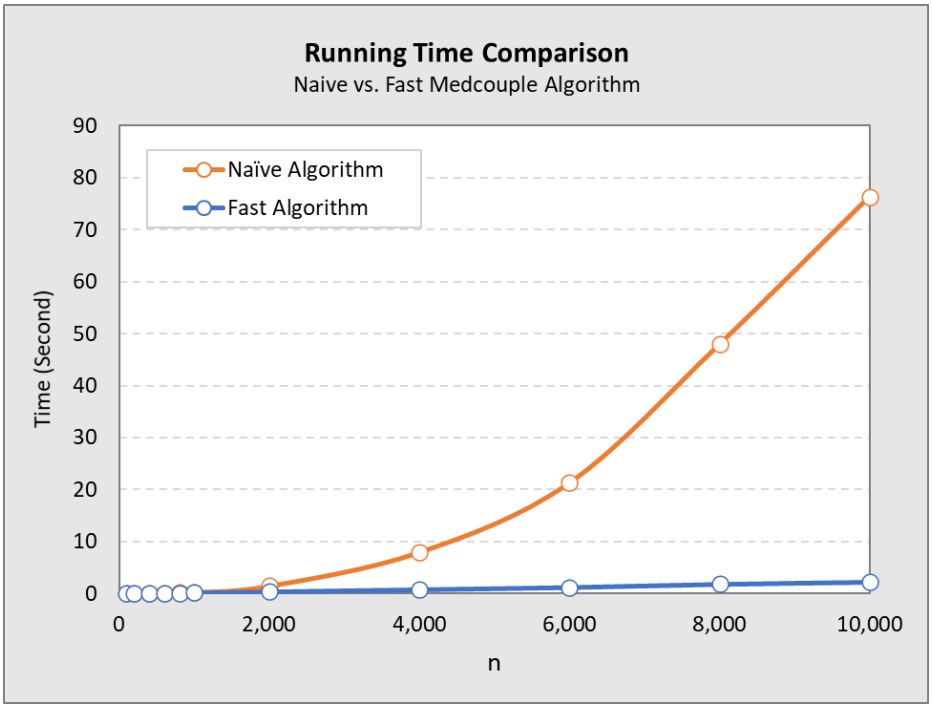

Figure 2. Running Time Comparison between Naïve and Fast Medcouple Algorithm

(Each data point is the average of 10 running time.)

COMPARE OUTLIER DETECTION METHODS

The five outlier detection methods are compared in a series of simulation studies. Each study involves 1,000 random samples simulated from a known distribution with three choices of the sample size: 20, 50, and 100. The five outlier detection methods are applied to each random sample. The percentages of the 1000 random samples that have at least one upper-side outlier detected under each method, as well as the average number of upper-side outliers per sample, are calculated and summarized in **Table 1**. Both metrics reflect the method's sensitivity for the outlier detection. Note that the reported outlier percentage and average outlier number per sample are for the upper side outliers only. A similar performance can be projected for the lower side outlier detection for a symmetric distribution.

Distribution	Sample Size	Univariate Outlier Detection Method									
		Z-Score		Median Absolute Deviation		Robust Regression		Boxplot		Medcouple Adj. Boxplot	
		% Sample w. Outliers	Avg # of Outliers per Sample	% Sample w. Outliers	Avg # of Outliers per Sample	% Sample w. Outliers	Avg # of Outliers per Sample	% Sample w. Outliers	Avg # of Outliers per Sample	% Sample w. Outliers	Avg # of Outliers per Sample
Normal Distribution	20	0.3%	0.0	12%	0.2	11%	0.1	12%	0.1	26%	0.4
	50	4%	0.0	14%	0.2	13%	0.2	23%	0.3	37%	0.9
	100	11%	0.1	20%	0.3	20%	0.2	32%	0.5	42%	1.0
Cauchy Distribution	20	35%	0.4	79%	1.4	75%	1.2	78%	1.3	80%	1.4
	50	60%	0.7	98%	3.6	97%	3.1	98%	3.7	98%	4.0
	100	74%	1.0	100%	7.0	100%	6.2	100%	7.5	100%	8.0
Lognormal Distribution	20	47%	0.5	87%	2.0	93%	2.1	77%	1.3	32%	0.5
	50	92%	1.2	99%	5.0	100%	5.7	98%	3.7	49%	0.9
	100	100%	2.3	100%	9.7	100%	11.7	100%	7.4	69%	1.6
Extreme Value Distribution (shape = 1, right-skewed)	20	76%	0.8	99%	3.3	99%	3.1	94%	2.0	55%	0.8
	50	100%	1.3	100%	8.4	100%	8.1	100%	5.8	87%	2.0
	100	100%	1.9	100%	16.5	100%	16.8	100%	11.9	98%	4.0
Extreme Value Distribution (shape = -0.5, left-skewed)	20	0.0%	0.0	1.7%	0.0	0.4%	0.0	1.3%	0.0	22%	0.4
	50	0.0%	0.0	0.0%	0.0	0.0%	0.0	0.2%	0.0	24%	0.6
	100	0.0%	0.0	0.0%	0.0	0.0%	0.0	0.0%	0.0	19%	0.6
Mixed Normal Distributions: p=0.95 for Normal (0, 1), p=0.05 for Normal (4, 0.5)	20	15%	0.2	58%	0.8	62%	0.9	58%	0.8	43%	0.7
	50	58%	0.7	89%	2.1	91%	2.2	92%	2.2	68%	1.7
	100	82%	1.6	98%	4.4	99%	4.6	100%	4.9	82%	3.7

Table 1. Compare Five Univariate Outlier Detection Methods

Notes: (1) The percentage of samples with outliers and the average number of outliers per sample are calculated based on 1,000 simulated random samples for each distribution and sample size choice. (2) The outlier detection in this simulation study is for the upper side only.

First, the simulation study shows that as the sample size increases from 20, 50, to 100, more and more outliers are detected per sample. This is true for all distribution choices and all outlier detection methods. This tendency can be expected because the number of outliers in a sample follows the binomial distribution, whose probably increases along with the sample size n.

Secondly, the simulation results in Table 1 show that the Z-Score method has the lowest sensitivity for outlier detection among the five methods for a symmetric distribution (Normal Distribution and Cauchy Distribution). This is because the Z-score method is based on the non-robust mean and standard deviation statistics that can be easily affected by outliers, while other methods mainly use the robust statistics such as median, quartiles, or order statistics. The Median Absolute Deviation and Robust Regression methods have similar sensitivity, followed by the Boxplot method and Medcouple Adjusted Boxplot method that has the highest sensitivity for a symmetric distribution.

Thirdly, when the distribution has a skewed shape (Lognormal Distribution, Extreme Value Distribution, or Mixed Distribution), the Medcouple Adjusted Boxplot method can detect less outliers on the same side of the distribution sknewness, but more outliers on the opposite side. This is because the lower and upper outlier criteria under the Medcouple Adjusted Boxplot method are accommodated to the skewness of the distribution through the Medcouple statistic. This may help reduce the false positive rate in outlier detection for a skewed distribution. The simulation also shows that sensitivity for the Median Absolute Deviation, Robust Regression, and Boxplot methods are quite comparable for a skewed distribution, but their sensitivities are all higher than the Z-Score method.

While the five methods may have different sensitivity to outliers under different distribution types, the actual choice of the outlier detection method in a real-life task shall be based on analysis objective. Some analysis projects may prefer a higher sensitivity with more outliers identified the better, while other analysis projects may want to tolerate some moderate outliers, especially when some of the outliers can be explained by a skewed distribution.

CONCLUSION

This article introduces five outlier detection methods for a univariate dataset that can be fully implemented in SAS. The five outlier detection methods are based on the use of different statistics, parametric or non-parametric. The five methods have shown different sensitivities by the different distribution scenarios. The choice of the outlier method shall be based on the analysis objective.

APPENDIX – SAS MACRO TO CALCULATE MEDCOUPLE STATISTIC BY NAÏVE AND FAST ALGORITHM

```
%macro medcouple_calc(input=, var=, method=NAIVE);

  /* Macro to calculate the MedCouple Value for a univariate dataset **;
  ** The Medcouple Value is used to construct a Medcouple Value Adjusted Boxplot
for Outlier Detection **;
  ** The macro implements two algorithm methods:
     - NAIVE: complexity of O(N*N)
     - FAST: complexity of O(N*LogN)
  */

  %local nobs;
  proc sql noprint; select count(&var.) into: nobs from &input.; quit;
  %if &nobs. > 0 %then %do;  * Only run medcouple calculation when the sample size
> 0;

    ** Naive Algorithm for Medcouple Calculation **;
    %if %upcase(&method.) = NAIVE %then %do;
      data temp(rename=(&var.=medcouple_x)); set &input.; if &var. ne .; run;
      proc sort data=temp; by descending medcouple_x; run;
      data temp; set temp; id = _n_ - 1; run;
      proc univariate data=temp noprint; var medcouple_x; output out=stat
median=median; run;
```

```
      data temp; if _n_ = 1 then set stat; set temp; run;
      data temp_plus(rename=(id=i)); set temp; if medcouple_x >= median; run;
      data temp_plus; set temp_plus nobs=n; p=n; run;
      data temp_minus(rename=(medcouple_x=medcouple_y id=j)); set temp; if
medcouple_x <= median; run;
      data temp_expand;
        set temp_plus;
        do k = 1 to n;
          set temp_minus point=k nobs=n; output;
        end;
      run;
      data temp_expand;
        set temp_expand;
        if medcouple_x > medcouple_y then h = ((medcouple_x - median) - (median -
medcouple_y)) / (medcouple_x - medcouple_y);
        else h = sign(p - 1 - i - j);
      run;
      proc univariate data=temp_expand noprint; var h; output out=medcouple_out
median=medcouple; run;
      proc datasets library=work noprint;  delete temp stat temp_plus temp_minus
temp_expand; run;
    %end;



    ** FAST Algorithm for Medcouple Calculation **;
    %if %upcase(&method.) = FAST %then %do;
      data temp(rename=(&var.=medcouple_x)); set &input.; if &var. ne .; run;

      proc iml;

        start medcouple_kernal(i, j, p, q) global(z_plus, z_minus);
          /* NOTE: i, j starts from 1 in SAS IML instead of 0 in the standard
algorithm */
          if i > p-1 | i < 0 | j > q-1 | j < 0 then h = 0;
          else if z_plus[i+1] > z_minus[j+1] then h = (z_plus[i+1] +
z_minus[j+1])/(z_plus[i+1] - z_minus[j+1]);
          else h = sign(p-1-i-j);
          return(h);
        finish;

        start greater_h(p, q, u);
          p_vector = J(p, 1);
          j = 0;
          do i = p-1 to 0 by -1;
            do while ((j < q) & (medcouple_kernal(i, j, p, q) > u));
              j = j + 1;
            end;
            p_vector[i+1] = j-1;
          end;
          return(p_vector);
        finish;

        start less_h(p, q, u);
          q_vector = J(p, 1);
          j = q-1;
          do i = 0 to p-1;
            do while ((j >= 0) & (medcouple_kernal(i, j, p, q) < u));
              j = j - 1;
            end;
            q_vector[i+1] = j+1;
          end;
          return(q_vector);
```

```
        finish;

        start weighted_median(x); /* x is a n by 2 matrix with the first column
being the values, and second columns being the weight */
          n = nrow(x);
          call sort(x,1);
          if mod(n, 2) = 1 then do;
            i = 0;
            sum_weight = 0;
            do until (sum_weight > 0.5);
              i = i + 1;
              sum_weight = sum_weight + x[i, 2];
            end;
            return(x[i, 1]);
          end;
          else do;
            i = 0;
            sum_weight_left = 0;
            do until (sum_weight_left >= 0.5);
              i = i + 1;
              sum_weight_left = sum_weight_left + x[i, 2];
            end;
            lower_wm = x[i, 1];
            j = n+1;
            sum_weight_right = 0;
            do until (sum_weight_right >= 0.5);
              j = j - 1;
              sum_weight_right = sum_weight_right + x[j, 2];
            end;
            upper_wm = x[j, 1];
            return((lower_wm+upper_wm)/2);
          end;
        finish;

        use temp;
        read all var {medcouple_x};
        n = nrow(medcouple_x);
        call sort(medcouple_x,1,1)          /* sort by descending order */;
        median = median(medcouple_x);
        r = 2*max(abs(medcouple_x));
        z = (medcouple_x - median)/r;
        z_plus = z[loc(z >= 0),];
        z_minus = z[loc(z <= 0),];

        p = nrow(z_plus);
        q = nrow(z_minus);

        /* Begin Kth pair algorithm (Johnson & Mizoguchi, 1978) */

        /* The initial left and right boundaries, two vectors of size p */
        left_border = J(p, 1, 0);
        right_border = J(p, 1, q-1);

        /* number of entries to the left of the left boundary */
        left_total = 0;

        /* number of entries to the left of the right boundary */
        right_total = p*q;

        /* Since we are indexing from zero, the medcouple index is one less than
its rank. */
        medcouple_index = floor(right_total/2);
```

```
        /* Iterate while the number of entries between the boundaries is greater
than the number of rows in the matrix. */

        loop_ind = 0;
        do while ((right_total - left_total > p) & (loop_ind < 1));

          /* Compute row medians and their associated weights, but skip any rows
that are already empty. */
          middle_idx  = loc(left_border <= right_border)-1;
          m = ncol(middle_idx);
          row_median = J(1, m);
          weights = J(1, m);
          do h = 1 to m;
            k = middle_idx[h];
            l = floor((left_border[k+1] + right_border[k+1])/2);
            row_median[h] = medcouple_kernal(k,l,p,q);
            weights[h] = right_border[k+1] - left_border[k+1] + 1;
          end;
          wm = weighted_median((row_median // (weights/sum(weights)))`);

          /* New tentative right and left boundaries */
          p_border = greater_h(p, q, wm);
          q_border = less_h(p, q, wm);
          p_total = sum(p_border) + nrow(p_border);
          q_total = sum(q_border);

          /* Determine which entries to discard, or if we've found the medcouple */

          if medcouple_index < p_total - 1 then do;
            right_border = p_border;
            right_total = p_total;
          end;
          else do;
            if medcouple_index > q_total - 1 then do;
              left_border = q_border;
              left_total = q_total;
            end;
            else do;
              medcouple = wm;
              loop_ind = 2;
            end;
          end;

        end;

        if loop_ind < 1 then do;
          remaining_idx = loc(left_border <= right_border) - 1;
          remaining_n = 0;
          do i = 1 to ncol(remaining_idx);
            i_idx = remaining_idx[i];
            remaining_n = remaining_n + (right_border[i_idx+1] -
left_border[i_idx+1] + 1);
          end;
          remaining = J(remaining_n, 1);

          l = 1;
          do i = 1 to ncol(remaining_idx);
            i_idx = remaining_idx[i];
            do j_idx = left_border[i_idx+1] to right_border[i_idx+1];
              remaining[l] = medcouple_kernal(i_idx, j_idx, p, q);
              l = l + 1;
            end;
          end;
```

```
            call sort(remaining,1,1);
            medcouple=remaining[medcouple_index - left_total+1];
          end;

          create medcouple_out from medcouple[colname="medcouple"];
          append from medcouple;
          close medcouple_out;

        run;
        proc datasets library=work noprint;  delete temp; run;
      %end;

    %end;

%mend medcouple_calc;
```

REFERENCES

Brys, G., Hubert, M., & Struyf, A. (2004). A robust measure of skewness. *Journal of Computational and Graphical Statistics*, 13(4), 996-1017.

Chen, C. (2002). Paper 265-27 Robust regression and outlier detection with the ROBUSTREG procedure. *Proceedings of the Proceedings of the Twenty-Seventh Annual SAS Users Group International Conference*.

Dixon, W. J. (1951). Ratios Involving Extreme values. *Annals of Mathematical Statistics*, 22(1), 68-78.

Draper, D. (1988). Rank-Based Robust Analysis of Linear Model. I. Exposition and Review. *Statistical Science*, 3(2), 239-257.

Gubbs, F. (1969). Procedures for Detecting Outlying Observations in Samples. *Technometrics*, 11(1), 1-21.

Hubert, M., & Vandervieren, E. (2008). An adjusted boxplot for skewed distributions. *Computational statistics & data analysis*, 52(12), 5186-5201.

Johnson, D. B., & Mizoguchi, T. (1978). Selecting the Kth element in X+Y and X1+X2+...+Xm. *SIAM Journal on Computing*, 7(2), 147-153.

Leys, C., Ley, C., Klein, O., Bernard, P., & Licata, L. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of experimental social psychology*, 49(4), 764-765.

Rosner, B. (1983). Percentage Points for a Generalized ESD many-Outlier Procedure. *Technometrics*, 25(2), 165-172.

Rousseeuw, P. J., & Croux, C. (1993). Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88(424), 1273-1283.

Rousseeuw, P. J., & Leroy, A. M. (2005). *Robust regression and outlier detection*. John Wiley & Sons.

Weeraratne, N. (2016). Consequence of Outliers in variable Control Charts. *International Journal of Advance Research And Innovative ideas in Education*, 2(4), 1025-1033.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Fan Yang, PhD.
Johnson and Johnson Vision
+1 (949) 590-0268
Fyang60@its.jnj.com