SD01

Using Biweights for Handling Outliers

George W. Dombi, PhD., Karmanos Cancer Center, Detroit, MI

Abstract

Values far from the center of the data are called outliers. These can be highly influential in determining the character of descriptive statistics. M-estimators are a class of central tendency measures than replace the mean and are highly resistant to local misbehavior caused by outlier data. Beaton and Tukey introduced the biweight as an iteratively reweighed measure of central tendency. Huber also introduced a different but complementary M-estimator. SAS will calculate both the biweight and the Huber weight function in Proc Stdize. Proc Nlin will allow for biweight and Huber weights to be utilized in user developed equations.

Introduction

In real data, some values are very different from the bulk of the data. Values far from the center of the data can be highly influential in determining the character of descriptive statistics. The mean is sensitive to outliers; for example:

Case 1:	{2, 3, 4, 5, 6}	mean = 4
Case 2:	{2, 3, 4, 5, 67}	mean = 16.2

In Case 2, the value 67 is an outlier. It pulls the mean to a value of 16.2 which is outside the bulk of the remaining data.

Terms like resistant and robust are often used to describe measures that are not so easily affected by outliers. For the purpose of this report, the term resistant will be assigned to descriptive statistics and the term robust will be assigned to inferential statistics. Thus, resistant descriptive statistics are insensitive to local misbehavior caused by outlier data. While robust inferential statistics are tolerant to departures from the assumptions of normal distributions. The mean is not a resistant descriptive statistic. The T-test is not a robust inferential statistic because it relies heavily on the mean. One the other hand, the median is a resistant measure of central tendency. One can see that the median is resistant to outliers since it completely ignores extreme values.

Case 1:	{2, 3, 4, 5, 6}	median = 4
Case 2:	{2, 3, 4, 5, 67}	median = 4

The Anova test, which relies more heavily on the standard deviation, is more robust than the t-test.

The concept of weighing data points relative to their distance from the center of the data set is an idea that is more widely accepted. Weighing is already in use but not acknowledged. The mean can be thought of using a weight equal to 1 for each data point. On the other hand, the median uses a weight of 0 for each rank ordered data point then uses weight equal 1 for the center most data.

What if you want to use a range a weights varying from 0 to 1 with respect to the distance from the center of the data? How would you determine these intermittent values of weight? This is part of the motivation for creating the M-estimators. M-estimators are a class of central tendency measures that have variable weight for each datum. Notice the continuation of the letter ‘M’; Mean, Median, and M-estimators are all measures of central tendency. M-estimators usually provide greater weight for data values near the center of the data cluster and decreasing weight for data away from the center. M-estimators are sometimes denoted by the Greek letter theta, Θ, and are set equal to the sum of (weight *individual data value) / sum of weights.

$$\text{M-estimator} = \Theta = \frac{\Sigma wi*xi}{\Sigma wi}$$

Xi = numbers
Wi = weights 0 … 1

Where Θ is iteratively refined by:

$$\frac{(xi - \Theta)}{\delta} = u$$

δ = MAD (median absolute deviation)
u = parameter descriptor that defines individual M-estimator.

In 1974, Albert Beaton and John Tukey introduced the concept of an iterative reweighed measure of central tendency, called the biweight, as an abbreviation for bisquare weight. The biweight is an M-estimator that satisfies the definitions given above and the weight is calculated as:

```
weight  =     {1-(u^2)/4.685^2}^2         when abs(u) <= 4.685
weight  =                0                when abs(u) > 4.685
```

This is not a very pretty picture in the way the biweight is shown but you can see the square of the square that gives it its name. The cut off point is a user selected value that is most often in the range of 4-6. In this case the cut point is 4.685, where the weight becomes zero. The biweight is considered to be robust since it is not sensitive to outlier data. Biweight depends on the calculated weights and the weights depend on the biweight so we need an iterative solution. Once calculated, the M-estimator becomes the new measure of central tendency for the next iteration.

Figure 1 shows the weight at various values of u. Note that when the difference between the data point and the M-estimator (in this case the biweight) is small, then the weight equals 1. Then as the difference increase, as u increases, the value of the biweight declines. For differences larger than 4.65 the biweight equals zero.

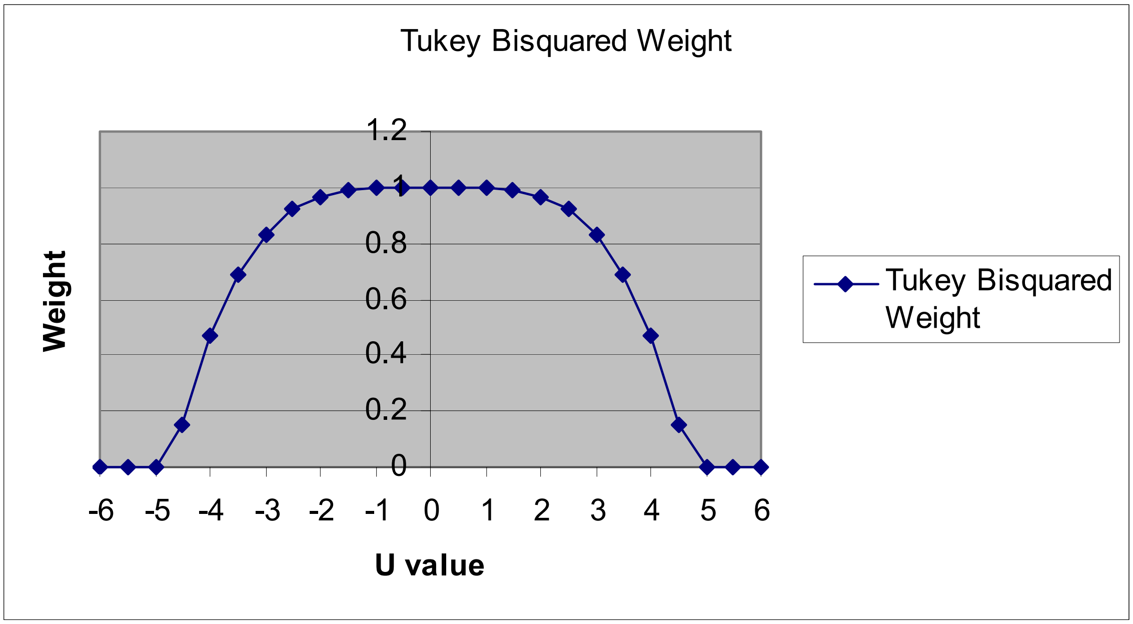

Figure 1. Biweight function decreases to 0 as distance of data point increases away from the middle of the data set.

Another kind of M-estimator is the Huber weight. This also needs a user selected cutoff point. Huber weights never go completely to zero and some people like that because it is easier to integrate as a continuous function. Often the Huber weight and the biweight are used in the same calculation. The Huber weight is utilized first to get near the convergence point and then use the biweights to get a discrete value cut off of zero for the outlier data.

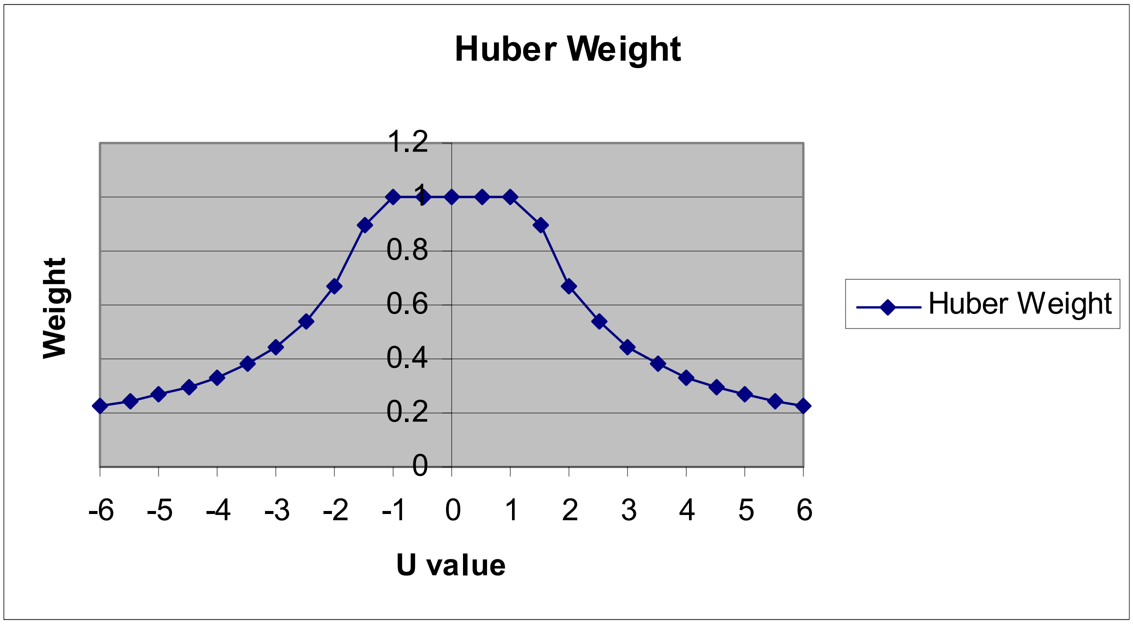

Figure 2. Huber weight function smoothly decreases, but not to 0, as the distance of a data point increases away from the middle of the data set.

Where is the biweight used? An application from Affymetrix is presented below in which genomic data is being calculated using a single step biweight procedure to reduce noise between pairs of data. Afflymetrix uses the single step method as a compromise between need for real time speed versus accuracy of the calculation. To calculate the single-step biweight, the median is first determined from replicates as the measure of central tendency. S, the median absolute deviation, MAD, of the replicates is then calculated as a measure of spread in the data. U combines the median, the cut-off value c and the MAD to form the bisquared weight, Wi Note the use of epsilon to keep the denominator from equaling 0 and blowing-up the equation. Wi is the weight for each point. Finally the tukey biweight is calculated as the sum of the weights*data / sum of the weights as the new measure of the center. This method is supposed to be accurate to at least 80% depending on the spread in the replicates. With little spread the method is 100% equivalent to the fully iterated biweight calculation.

- One-Step Tukey's Biweight
  (implemented in Affymetrix software package):
    - The probe pair vote is weighted more strongly if this probe pair Signal value is closer to the median value for a probe set.
  - tukey.biweight <-- function (x, c=5, epsilon=0.0001)
    - { m = median (x values)
    - s = median (abs (x(i) - m))

- u = (x - m) / ((c * s) + epsilon)
- W = rep (0, length(x))
- i = abs(u) <= 1
- w[i] = ((1 - u^2)^2)[i]
- tukey.biweight (t.bi) = sum(w * x) / sum(w) (like a mean)
- return(t.bi) }

So what does SAS do with the biweight? There are three procedures that currently utilize iterative reweighing techniques. Proc Stdize will produce a biweight or Huber weight for a list of numbers. Proc Anova has nothing to do with the biweight, but I found an internet article that purports to be an Anova engine utilizing the biweight and MAD in place of the mean and standard deviation. Proc Nlin actually mentions the biweight in its older documentation and gives an example of how to setup an iterative re-weighting procedure.

Proc Stdize. First, select a data set. Then define a method option. The method option picks the type of central tendency to be calculated. Use ‘method ABW ‘ to select the Tukey biweight. Use ‘method AHUB’ to select the Huber weight. In the example below, a title statement is added and the selection of the cases. The output is presented below. Note that in case 1, both the biweight and the Huber weight return the mean value of 4.0. In case 2, the Huber weight returns the value of the median and the biweight returns a value smaller than that. It is as if the biweight has fully ignored the existence of the 67 value and is trying to calculate the mean of the remaining values, which would be 3.5. Perhaps the 3.465 value is a single step iteration to 3.5. That is a complete guess on my part. I don’t know if Proc Stdize uses a single step or a full iterative process to find the biweight.

- **proc stdize** data=perm.data method = abw(**4**) pstat; *method = ahub(1.3)
- title2 'METHOD=ABW(4)';
- var Case1 Case2;
- **run**;
-

```
                              Biweight  Huber
Case 1:  {2,3,4,5,6}             4.0      4.0
Case 2:  {2,3,4,5,67}           3.465     4.0
```

As stated before SAS Proc Anova has nothing to do with biweights. But there has been an attempt posted on an internet page to create a bisquare-weighted Anova, which the unknown author calls a bANOVA. I was unable to make it work but that may just be my ignorance with SAS and getting the input data set in the correct form. This bANOVA program tries to create a very robust anova using the biweight as the measure of central tendency and the MAD, median absolute deviation, as the measure of spread.

Finally, here is the PROC NLIN procedure. The first example is setup to fit a straight line. After the data name you can see the ‘nohalve’ option, which shuts off a procedure in PROC NLIN that otherwise takes half steps when trying to fit the data. By shutting off this option, the user is saying that they want to fit using a different protocol that will be supplied below. Then there are the two obligatory title lines. Next are the two initial parameter estimates, the slope and intercept. Proc Nlin seems to run better with initial parameter estimates even for straight lines. The model statement defines the straight line using the Y variable of ‘Space’ and the X variable name of ‘Time’. Now comes the tricky part, which is defining the first derivatives of the model parameters. PROC NLIN utilizes the derivates to help step through parameter space to find the local minimum. Next two biweight constants are defined to regulate the fit. Variable r is used to determine how the fit will proceed. Then output is included in three output files. File c is altered with the next section and fed back into the PROC NLIN to iterate the next step. An example of the fitted data is shown in Figure 3 and a table is given below that to show the individual weights at each point in the data.

```
●  proc nlin data = Biweight nohalve;
●    title1 'Tukey biweight of Straight Line data';
●    title2 'Y-line fitting';
●    parms a= 1 b=1 ;
●    model Space = a + b*Time;  /** straight line model **/
●    der.a = 1;
●    der.b = Time;
●    resid = Space-model.Space;
●    sigma = 2;
●    b2 = 4.685;
●    r = abs(resid/sigma);
●    if r <= b2 then _weight_=(1-(r/b2)**2)**2;
●    if r > b2 then _weight_= 0;
●    output out=c  p=predict r=rbi;
●  data c;
●  set c;
●    sigma = 2;
●    b2 = 4.685;
●    r=abs(rbi/sigma);
●    if r <= b2 then _weight_ = (1-(r/b2)**2)**2;
●    else _weight_ = 0;
●  run;
```

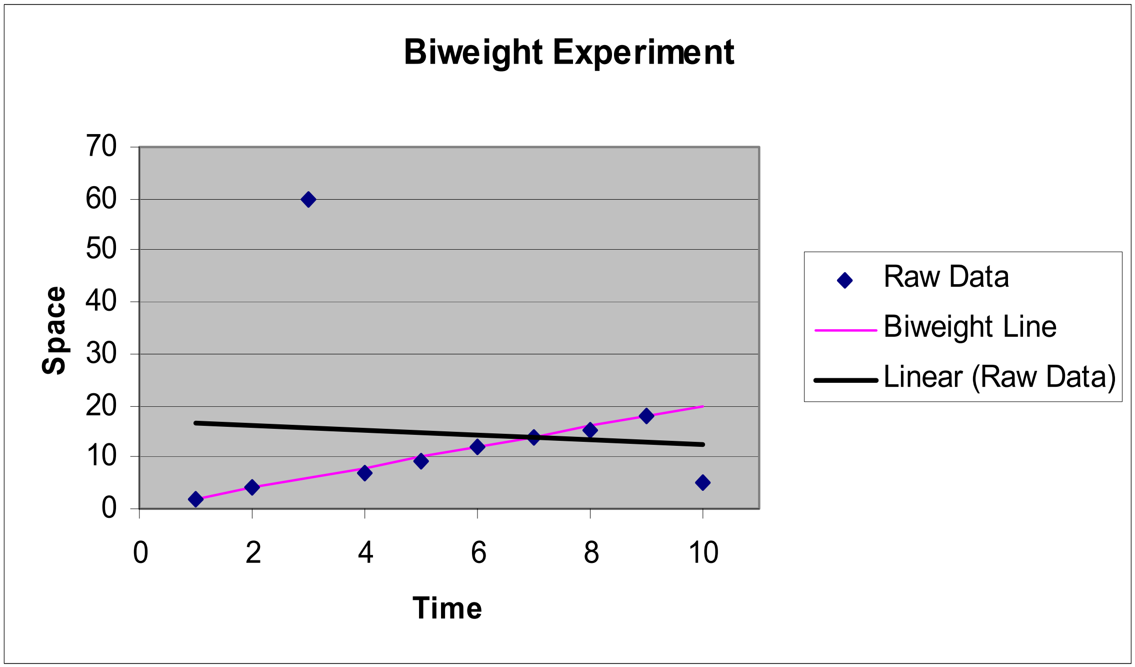

Figure 3. Raw data with two outliers. Black line is the linear regression through the raw data. The Biweight fitted line is in magenta.

Table 1 shows values that produced Figure 3. X and Y values are in the first two columns with two outliers highlighted in bold. Column 3 gives values of the points plotted for the biweight line. Column 4 gives the difference squared between actual y-value and the plotted y-value. Column 5 shows biweight values. Data points close to the line have a weight of 1 and outliers have a weight of 0. Those points just off of the line have intermediate weight values. The last column of numbers plots the least squared regression line with the outlier data in place.

Table 1. Proc Nlin example of straight line biweight fit in data with 2 outlier points.

		Y=mX + b			Biweight	Least Sqr
Xvalue	Yvalue	Predicted	diff^2	Biweight	Diff^2	Line
1	2	2.000037	1.4E-09	1	1.4E-09	14.49091
2	4	4.00003	9.15E-10	1	9.15E-10	14.84848
3	**60**	6.000023	2915.998	**0**	**0**	15.20606
4	7	8.000016	1.000032	0.790101	0.790126	15.56364
5	9	10.00001	1.000018	0.790107	0.79012	15.92121
6	12	12	2.72E-12	1	2.72E-12	16.27879
7	14	13.99999	3.03E-11	1	3.03E-11	16.63636
8	15	15.99999	0.999975	0.790123	0.790103	16.99394
9	18	17.99998	3.92E-10	1	3.92E-10	17.35152
10	**5**	19.99997	224.9992	**0**	**0**	17.70909

The second example shows a PROC NLIN procedure that uses the biweight method to fit a Gaussian line with outliers. After the data set name, the nohalve statement appears and also a method statement. The method statement requests the Marquardt method of steepest descend to find the local minimum. Then two title lines are added. Now there are three fitting parameters; a is for the height, b is for the center and c for the width. Since curve fitting is more art than science, it is sometimes useful to play around with the range of variable space to limit the search for the local minimum. The first 'parms' statement was my initial attempt to select an area where I wanted to search. The second line is a more refined set of boundary conditions. The model to fit the data is a simplified Gaussian model. The next three lines are the first derivatives of the fitting parameters. If your calculus is as rusty as mine, then I suggest finding a web site to solve these equations. The rest of the SAS code is the same as seen in the linear case seen above. Biweights are calculated then plugged into the PROC NLIN to be used for the next level of iteration. This is repeated until there is a convergence or 1000 iterations. The fit is seen in Figure 4.

```
●  proc nlin data = Biweight2 nohalve method=marquardt;
●    title1 'Tukey biweight of Gaussian data';
●    title2 'Y-curve fitting';
●    /** a = height, b = center, c = width **/
●    /* parms a=55 to 70 by .1 b=8 to 12 by .1 c= 10 to 20 by .1; */
●    parms a=60 to 66 by .1 b=9 to 11 by .1 c= 12 to 16 by .1 ;
●    model Count = a*exp(-1*((Time-b)/c)**2);
●    der.a = exp(-1*((Time-b)/c)**2);
●    der.b = (2*a*(Time-b)*exp(-1*((Time-b)/c)**2))/c**2 ;
●    der.c = (2*a*(Time-b)**2*exp(-1*((Time-b)/c)**2))/c**3 ;
●    resid = Count-model.Count;
●    sigma = 2;
●    b2 = 4.685;
●    r = abs(resid/sigma);
●    if r <= b2 then _weight_=(1-(r/b2)**2)**2;
●    if r > b2 then _weight_= 0;
●    output out=c  p=predict r=rbi;
●  data c;
●  set c;
●    sigma = 2;
●    b2 = 4.685;
●    r=abs(rbi/sigma);
●    if r <= b2 then _weight_ = (1-(r/b2)**2)**2;
●    else _weight_ = 0;
●  run;
```

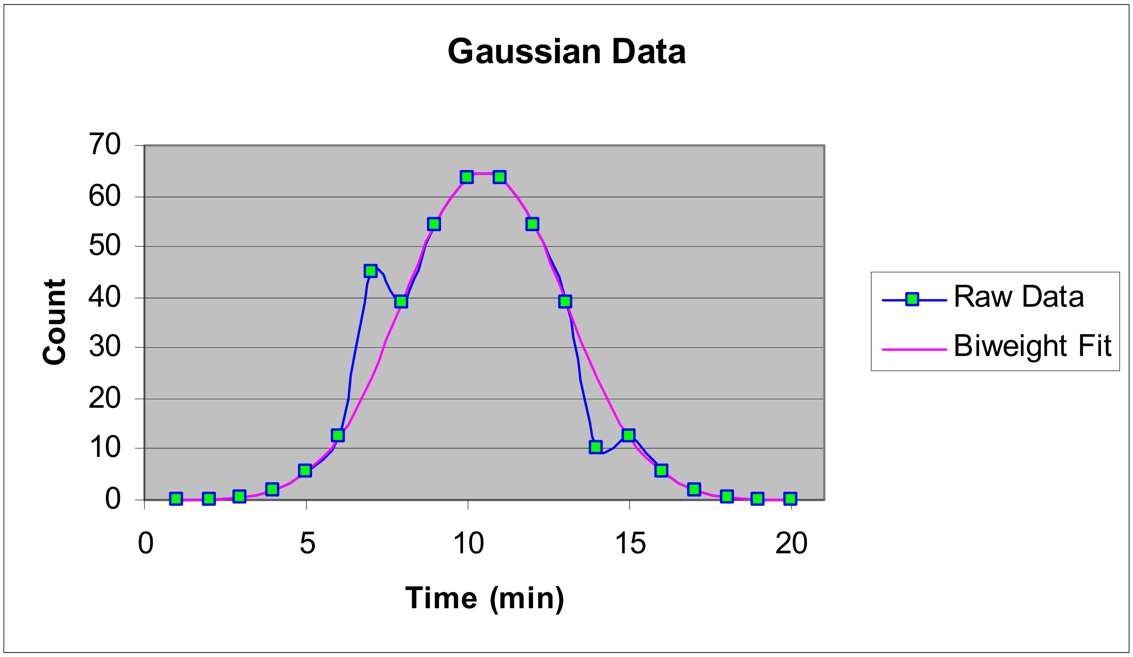

Figure 4. Raw data with two outliers. The darker line simply connects the data points in the raw data. The lighter line is the biweight fitted line.

In conclusion, biweights are a resistant estimator of the center of the data. Biweights are calculated by iteration with some parameters preset by the user. Biweights calculations are computer intensive. Tukey first demonstrated them by hand for a straight line but now they can be more easily calculated on the computer for curvilinear shapes. SAS has addressed biweights in Proc Nlin and Proc Stdize. Hopefully more use of the biweights and Huber weights will be made in later editions of SAS. Hopefully SAS will especially develop a biweight based ANOVA procedure like the bANOVA.

References:

1) **http://mathworld.wolfram.com/TukeysBiweight.html**
2) BANOVA code : **http://pubpages.unh.edu/~wws/**
3) SAS version 8, SAS Institute Inc., Cary, NC.
4) SAS User's Guide: Statistics, Version 5 Edition. NLIN Procedure, Iteratively Reweighted Least Squares: Example 5, pp 598-600.
5) Goodall, Colin, M-estimators of Location: An Outline of the Theory, in Understanding Robust and Exploratory Data Analysis, eds: David C. Hoaglin, Frederick Mosteller and John W. Tukey, 1983, pp. 339-403.
6) Affymetrix use of Tukey Biweight:
http://www.affymetrix.com/support/technical/technotes/statistical_reference_guide.pdf