

COMPUTATIONAL ISSUES
IN FITTING STATISTICAL DISTRIBUTIONS TO DATA

Zaven A. Karian¹, Edward J. Dudewicz²

¹ Department of Mathematics and Computer Science, Denison University, Granville, Ohio 43023 (karian@denison.edu).

²Department of Mathematics, Syracuse University, Syracuse, New York 13244-1150 (dudewicz@syr.edu).

Key words: Fitting distributions, generalized lambda distribution, Johnson system, method of moments, statistical computing, Weibull distribution.

ABSTRACT

In fitting statistical distributions to data the practitioner almost always needs to compute the parameters of a family of distributions. Such computations are frequently difficult to implement and may require considerable effort and computation time. This paper discusses the computing difficulties that can arise in fitting statistical distributions to data and illustrates these problems through examples that use the Weibull distribution, the generalized lambda distribution, and the Johnson system of distributions. In all cases fits are obtained by the method of moments using either the Maple computing environment or the **R** statistical system.

1. INTRODUCTION

At least three major steps must be undertaken when fitting a statistical distribution to data. First, a decision needs to be made about what type (or family) of distributions to consider; next, it must be decided what fitting method to use (e.g., method of moments, maximum likelihood, least squares); finally, based on the two prior decisions, specific computational schemes must be invoked to estimate the parameters associated with the fit.

The computational difficulties that are the focus of this paper vary considerably from one family of distributions to another as well as from one fitting method to another. These prob-

lems will be highlighted through several examples, each of which will illustrate, in addition to specific methodologies, a range of computational problems that could be encountered. One common thread in all of the examples that follow is the use of graphic analysis to guide the computations that need to be made.

In order to keep the emphasis on computation, a single data set will be considered and a single method (the method of moments) will be used in all the examples. The data set comes from the Indiana Twin Study and represents the birth weight (in pounds) of the second member of a twin birth (for more details see Karian and Dudewicz (2000), pp. 99–100). The $n = 123$ birth weights in this study are (in increasing order):

2.81	2.93	3.16	3.19	3.24	3.40	3.56	3.66	3.66	3.78	3.81	3.81
3.83	3.83	3.88	4.12	4.13	4.15	4.22	4.28	4.31	4.31	4.38	4.44
4.44	4.44	4.50	4.56	4.58	4.60	4.63	4.63	4.69	4.69	4.69	4.75
4.75	4.75	4.78	4.81	4.88	4.88	4.94	4.94	4.97	5.00	5.00	5.00
5.06	5.13	5.13	5.13	5.19	5.19	5.19	5.31	5.38	5.38	5.38	5.38
5.38	5.38	5.41	5.48	5.50	5.50	5.50	5.56	5.63	5.63	5.63	5.69
5.69	5.69	5.72	5.75	5.75	5.75	5.81	5.81	5.81	5.85	5.88	5.88
5.94	6.05	6.06	6.10	6.10	6.13	6.13	6.16	6.16	6.18	6.19	6.19
6.19	6.22	6.22	6.25	6.31	6.31	6.31	6.31	6.38	6.38	6.44	6.53
6.56	6.60	6.63	6.69	6.75	6.81	7.10	7.22	7.25	7.31	7.31	7.41
7.75	8.00	8.14									

2. A WEIBULL FIT

This initial illustration attempts to fit a Weibull distribution to the data under consideration. The choice of the Weibull distribution is predicated on its simplicity (only two parameters) and the fact that there is some *a priori* visual evidence that the Weibull distribution may provide a good fit.

The parameterization of the Weibull distribution that is used here has, for $b > 0$, $c > 0$,

the pdf

$$f(x) = \begin{cases} \left(\frac{cx^{c-1}}{b^c}\right) e^{-(x/b)^c} & \text{for } x > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Thus, the basic task in this example is to

1. Compute the first two moments, $\hat{\alpha}_1$ and $\hat{\alpha}_2$, respectively, of the data where

$$\hat{\alpha}_1 = \bar{X} = \frac{1}{n} \sum_{i=1}^n X_i \quad (2)$$

$$\hat{\alpha}_2 = \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2. \quad (3)$$

2. Derive, look up, or somehow obtain the first two moments about the mean, α_1 and α_2 , respectively, of the Weibull distribution.

3. Solve the system of equations $\{\alpha_1 = \hat{\alpha}_1, \alpha_2 = \hat{\alpha}_2\}$ for b and c .

The distribution with the (b, c) pair obtained in Step 3 will be the optimal method of moments fit of a Weibull distribution to this data.

The solving of the system $\{\alpha_1 = \hat{\alpha}_1, \alpha_2 = \hat{\alpha}_2\}$ for b and c is rather easy in this case, as illustrated below in the Maple interaction that executes these three steps (without too much trouble, one could also write a program in **R** that would estimate b and c). In Maple the “>” symbol prompts a command from the user which must end with either a “:” or a “;” — in the former case, the command is executed and no output is issued while in the latter case, the results of the command execution appear at the screen. In the following, **A1hat**, **A2hat**, **A1**, **A2** denote $\hat{\alpha}_1$, $\hat{\alpha}_2$, α_1 , α_2 , respectively.

```
> Data:=[2.81,2.93,3.16, ... ]:
```

```
> n:=nops(Data);
```

```
n := 123
```

```
> A1hat := sum(Data[i], i=1..n)/n; A2hat := sum((Data[i]-A1hat)^2, i=1..n)/n;
```

```
A1hat := 5.366585366
```

```
A2hat := 1.203253381
```

```
> pdf := (c*x^(c-1)/b^c)*exp(-(x/b)^c):
> A1 := simplify(int(x*pdf, x=0..infinity),symbolic);
```

$$A1 := b\Gamma\left(\frac{c+1}{c}\right)$$

```
> A2:=simplify(int(x*x*pdf, x=0..infinity),symbolic)-A1^2;
```

$$A2 := b^2\Gamma\left(\frac{c+2}{c}\right) - \Gamma\left(\frac{c+1}{c}\right)^2$$

```
> solve({A1=a1, A2=a2}, {b, c});
```

Warning, solutions may have been lost

```
> fsolve({A1=A1hat, A2=A2hat}, {b=0..10, c=0..10});
```

$$\{b = 5.803489230, c = 5.661799028\}$$

```
> Fitpdf:=subs({b=5.803489230, c=5.661799028}, pdf);
```

$$Fitpdf := 0.0002685961459x^{4.661799028}\exp(-0.00004744007062x^{5.661799028})$$

Note that the initial attempt to obtain an exact solution to $\{\alpha_1 = \hat{\alpha}_1, \alpha_2 = \hat{\alpha}_2\}$ produced the warning: Warning, solutions may have been lost but the subsequent `fsolve` command gave approximate values for a and b .

Having obtained `Fitpdf`, the pdf of the fitted Weibull distribution, the next natural question to ask is how good is the fit? It always makes sense to apply the “eyeball” test by looking at a graph of the fitted pdf superimposed on a histogram of the data and the graph of the cdf of the fitted distribution superimposed on the empirical cdf. These graphs, given as Figures 1 and 2, respectively, show that the fit is reasonably good. To obtain a quantitative measure of the quality of the fit, we can perform a chi-square test, as done with Maple below. In the Maple interaction that follows, `Fitcdf` stands for the cdf of the Weibull distribution that was fitted; `ClassBdry` establishes the intervals

$$\begin{aligned} &(-\infty, 3.955), [3.955, 4.455), [4.455, 4.955), [4.955, 5.455), [5.455, 5.955), \\ &[5.955, 6.455), [6.455, 6.955), [6.955, \infty); \end{aligned}$$

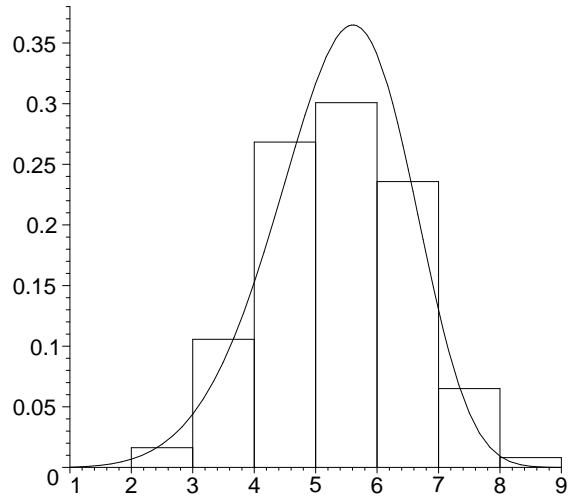


Figure 1. Fitted Weibull pdf with histogram.

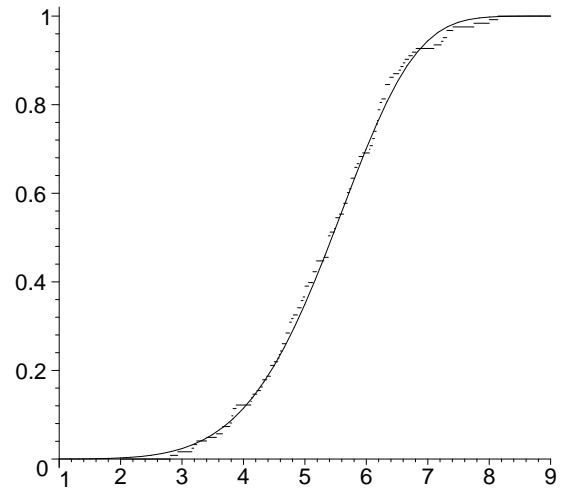


Figure 2. Fitted cdf with empirical cdf.

`Probs`, starts with 0 then computes the cumulative probabilities of the fitted distribution at boundary points and ends with 1; `EF` and `OF` are, respectively, the expected and observed frequencies of each of the 8 classes; and finally, `ChiSq` and `pVal` give the chi-square statistic and its associated p -value for this test.

```
> Fitcdf:=int(Fitpdf, x=0..t):
```

```
> ClassBdry:=seq(3.955+.5*i, i=0..6);
```

$$ClassBdry := [3.955, 4.455, 4.955, 5.455, 5.955, 6.455, 6.955]$$

```
> Probs:=evalf([0,seq(subs(t=ClassBdry[i],Fitcdf), i=1..7),1]);
```

$$Probs := [0., 0.1077804334, 0.2004963842, 0.3354477994, 0.5055225488, 0.6856027720, \\ 0.8390209093, 0.9383647712, 1.]$$

```
> EF:=seq(nops(Data)*(Probs[i+1]-Probs[i]), i=1..8);
```

$$EF := [13.25699331, 11.40406195, 16.59902407, 20.91919418, 22.14986745, 18.87043089, \\ 12.21929501, 7.581133142]$$

```

OF:=[15,11,18,19,22,22,7,9]:
> ChiSq:=sum((EF[i]-OF[i])^2/EF[i], i=1..8); pVal:=1-ChisquareCDF(5,ChiSq);

```

$$ChiSq := 3.552736855$$

$$pVal := 0.6154233280$$

The Kolmogorov-Smirnov statistic for this fit is 0.0462133355 (for reference, the critical point for significance level of 0.20 is .0965).

3. USE OF THE GENERALIZED LAMBDA DISTRIBUTION

The generalized lambda distribution (GLD) is a four parameter family of distributions that can assume a wide range of shapes (see Section 1.4 of Karian and Dudewicz (2000) for details). Because of its flexibility, the GLD has been used for fitting data sets through a variety of methods (method of moments, maximum likelihood, methods based on percentiles, least squares) and has been investigated by many authors (Hahn and Shapiro (1967), Burr (1973), Ramberg and Schmisser (1972, 1974), Ramberg, Dudewicz, Tadikamalla and Mykytka (1979), Dudewicz and Karian (1996, 1999), Öztürk and Dale (1985), Karian and Dudewicz (1999, 2003), Gilchrist (2000)), Karian, Dudewicz and McDonald (1996)).

Unlike the Weibull fit obtained in Section 2, fitting a GLD to the data poses a number of difficulties. The four-parameter GLD family is defined through its quantile function, $Q(y)$, by

$$Q(y) = Q(y; \lambda_1, \lambda_2, \lambda_3, \lambda_4) = \lambda_1 + \frac{y^{\lambda_3} - (1 - y)^{\lambda_4}}{\lambda_2}, \quad (4)$$

where $0 \leq y \leq 1$. All (λ_3, λ_4) points in the first and third quadrants and some points in the second and fourth quadrants allow (4) to define valid distributions (we will restrict the discussion in this paper to points in the first and third quadrants; for a detailed discussion of the parameter space of the GLD, see Karian and Dudewicz (2000), Chapter 1). Since for a GLD fit the four parameters $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ must be estimated, the first four moments will have to be used. To guarantee the existence of the first four GLD moments λ_3 and λ_4 must further be restricted by $-1/4 < \lambda_3$ and $-1/4 < \lambda_4$.

The fitting process based on the method of moments requires, in addition to the first two sample moments given in (2) and (3), the third and fourth moments, $\hat{\alpha}_3$ and $\hat{\alpha}_4$, given by

$$\hat{\alpha}_3 = \sum_{i=1}^n (X_i - \bar{X})^3 / (n\hat{\sigma}^3), \quad (5)$$

$$\hat{\alpha}_4 = \sum_{i=1}^n (X_i - \bar{X})^4 / (n\hat{\sigma}^4). \quad (6)$$

The first four GLD moments $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are given below (see Chapter 2 of Karian and Dudewicz (2000)).

$$\alpha_1 = \mu = E(X) = \lambda_1 + \frac{A}{\lambda_2}, \quad (7)$$

$$\alpha_2 = \sigma^2 = E[(X - \mu)^2] = \frac{B - A^2}{\lambda_2^2}, \quad (8)$$

$$\alpha_3 = E(X - E(X))^3 / \sigma^3 = \frac{C - 3AB + 2A^3}{\lambda_2^3 \sigma^3}, \quad (9)$$

$$\alpha_4 = E(X - E(X))^4 / \sigma^4 = \frac{D - 4AC + 6A^2B - 3A^4}{\lambda_2^4 \sigma^4}, \quad (10)$$

with

$$A = \frac{1}{1 + \lambda_3} - \frac{1}{1 + \lambda_4}, \quad (11)$$

$$B = \frac{1}{1 + 2\lambda_3} + \frac{1}{1 + 2\lambda_4} - 2\beta(1 + \lambda_3, 1 + \lambda_4), \quad (12)$$

$$C = \frac{1}{1 + 3\lambda_3} - \frac{1}{1 + 3\lambda_4} - 3\beta(1 + 2\lambda_3, 1 + \lambda_4) + 3\beta(1 + \lambda_3, 1 + 2\lambda_4), \quad (13)$$

$$D = \frac{1}{1 + 4\lambda_3} + \frac{1}{1 + 4\lambda_4} - 4\beta(1 + 3\lambda_3, 1 + \lambda_4) + 6\beta(1 + 2\lambda_3, 1 + 2\lambda_4) - 4\beta(1 + \lambda_3, 1 + 3\lambda_4). \quad (14)$$

where $\beta(u, v)$ is the beta function with arguments u and v .

To solve the system of equations $\{\alpha_i = \hat{\alpha}_i \text{ for } i = 1, 2, 3, 4\}$ for $\lambda_1, \lambda_2, \lambda_3$, and λ_4 , we note that the last two of these equations,

$$\{\alpha_3 = \hat{\alpha}_3, \alpha_4 = \hat{\alpha}_4\}, \quad (15)$$

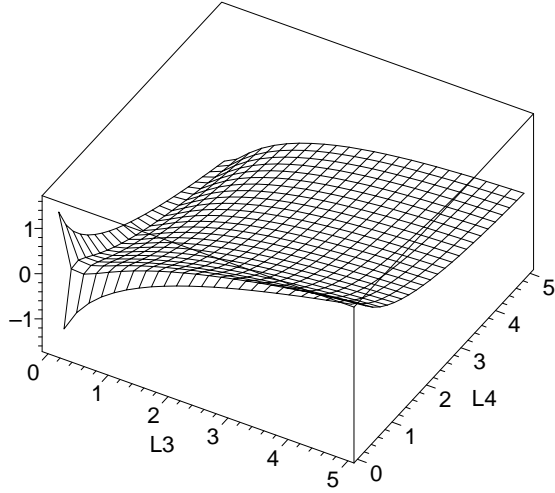


Figure 3. α_3 as a function of λ_3 and λ_4 (designated by L3 and L4) in the first quadrant.

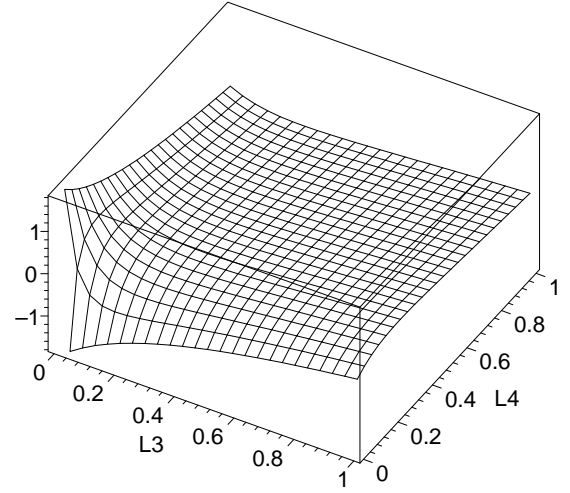


Figure 4. A zoomed in version of Figure 3.

depend only on λ_3 and λ_4 , making it reasonable to try to obtain λ_3 and λ_4 from these two equations, then use (8) to solve for λ_2 and eventually determine λ_1 from (7). To get some insight on how to search for a solution (or solutions), it makes sense to look at the surfaces represented by α_3 and α_4 . Figure 3 shows α_3 as a function of λ_3 and λ_4 (designated by L3 and L4, respectively). The surface appears to be smooth, even when zoomed in near $(0, 0)$ as in Figure 4, and one would expect that most solution searching algorithms would have no difficulty with the $\alpha_3 = \hat{\alpha}_3$ equation.

There should be more concern about the surface for α_4 given in Figure 5, with a zoomed in version in Figure 6. The curvatures of the surface along the λ_3 and λ_4 axes indicate a strong possibility of multiple solutions in those regions and the sharp rise as (λ_3, λ_4) approaches $(0, 0)$ is likely to make it difficult to look for solutions in the vicinity of $(0, 0)$.

A similar investigation for the third (λ_3, λ_4) quadrant yields Figures 7 through 10 that describe the α_3 and α_4 surfaces in this quadrant. The surface shears that appears in Figure 7 are not as ominous as they look. To understand their cause, consider, for a fixed λ_4 , the terms in C of equation (13). When λ_3 approaches $-1/3$ from the left, all the terms, except the first (i.e., $1/(1 + 3\lambda_3)$) assume specific and finite values. However, the first term, hence also C , tends to $-\infty$. But if λ_3 approaches $-1/3$ from the right, then C goes to $+\infty$.

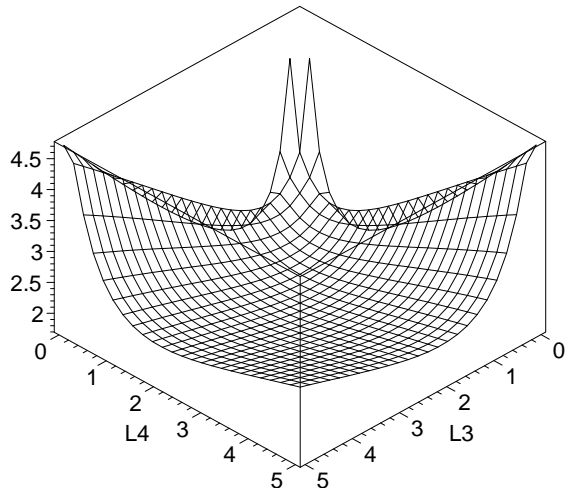


Figure 5. α_4 as a function of λ_3 and λ_4 (designated by L3 and L4) in the first quadrant.

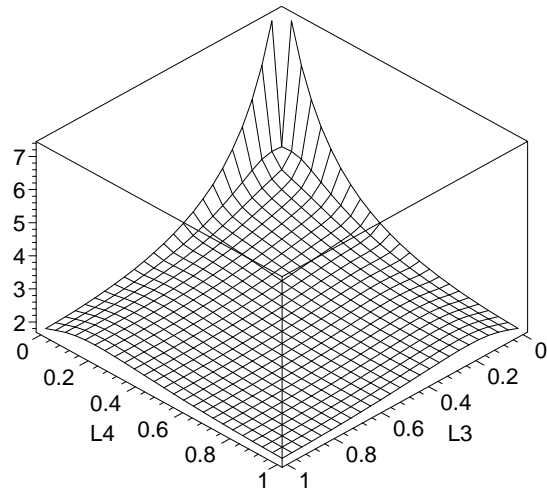


Figure 6. A zoomed in version of Figure 3.

Equation (9) shows that this discontinuity in C is transferred to α_3 and we see its effect in Figure 7. This is not of great concern since, as stated earlier, in the third quadrant we must have λ_3 and λ_4 greater than $-1/4$ for the first four moments to exist and, as can be seen in Figure 8, α_3 is smooth in this region.

Figure 10 gives an indication of another problem: the erratic behavior of α_4 near the origin. The difficulty that we see is due to imprecise calculations and if we substantially improve the accuracy of our computations, this graphic “mirage” disappears, as shown in Figure 11, for which 30-digit accuracy is used. Figure 10 serves as a warning that whenever computations require (λ_3, λ_4) to be near $(0, 0)$, high precision computation, requiring considerable computing time, may be needed. Symbolic computing systems such as Maple and Mathematica allow the user to set arbitrary limits on the precision of computations (at the expense of computation time). However, commonly used programming languages, such as C, FORTRAN, etc., as well as \mathbf{R} , are designed to take advantage of the high-speed computation that is built into the hardware of the computer, making it virtually impossible to improve precision beyond limits set by the hardware.

Figures 3 through 11 give an overview of what to expect when solving (15) but do not provide any assistance regarding the location, even in a general sense, the values of λ_3 and

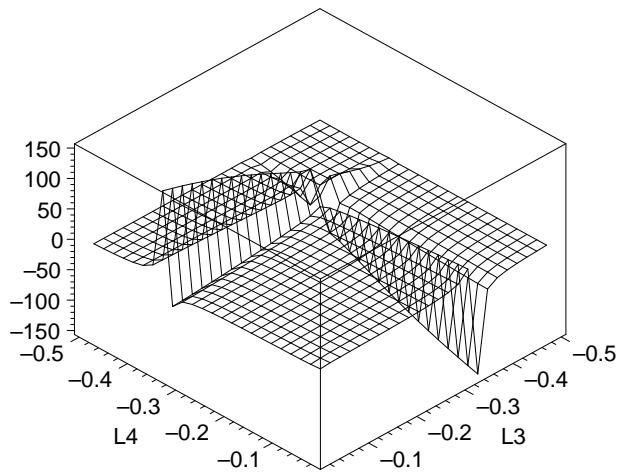


Figure 7. α_3 as a function of λ_3 and λ_4 (designated by L3 and L4) in the third quadrant.

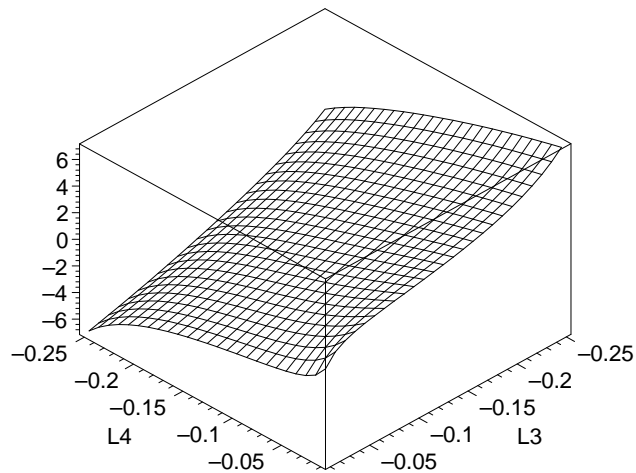


Figure 8. A zoomed in version of Figure 7.

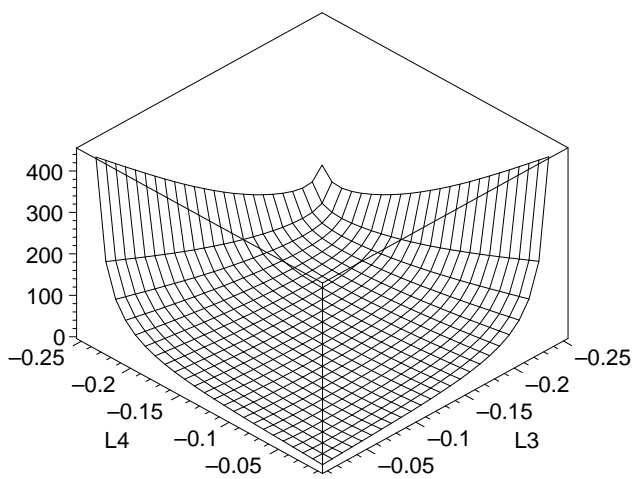


Figure 9. α_4 as a function of λ_3 and λ_4 (designated by L3 and L4) in the third quadrant.

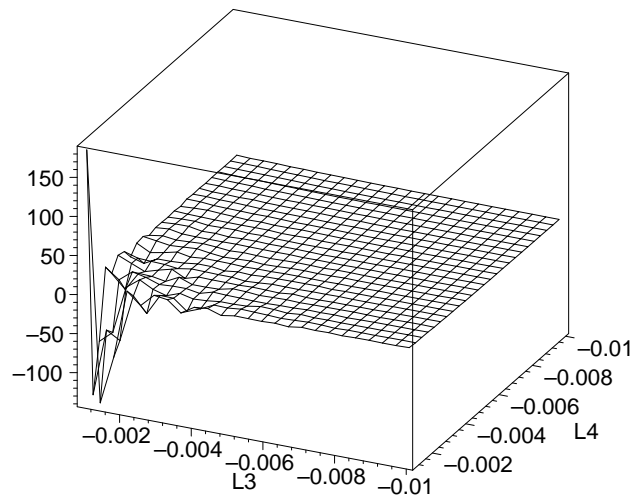


Figure 10. A zoomed in version of Figure 9.

λ_4 that will be appropriate for a specific data set. Suppose we want to fit a GLD distribution to the data to which a Weibull distribution was fitted in the previous section. The easy part of what needs to be done is the computation of $\hat{\alpha}_1, \hat{\alpha}_2, \hat{\alpha}_3, \hat{\alpha}_4$, which results in

$$\hat{\alpha}_1 = 5.366585366, \quad \hat{\alpha}_2 = 1.203253381, \quad \hat{\alpha}_3 = -.01218888030, \quad \hat{\alpha}_4 = 2.766540453.$$

These values, together with Figures 3 through 6 indicate that there is likely to be solution in the first (λ_3, λ_4) quadrant. Figure 11 indicates that even at its low point, near the origin, α_4 stays above 4 and it is not likely that there is a solution in the third quadrant. To get a rough idea of the values of λ_3 and λ_4 that correspond to these $\hat{\alpha}_3$ and $\hat{\alpha}_4$, contour plots of the α_3 and α_4 surfaces, with the specified $\hat{\alpha}_3$ and $\hat{\alpha}_4$ can be plotted, as in Figures 12 and 13. When these plots are superimposed (Figure 14), we see an indication that there is a solution where the contours intersect near the point $(\lambda_3, \lambda_4) = (0.2, 0.2)$.

Having established that (λ_3, λ_4) is roughly $(0.2, 0.2)$, raises several questions: How accurate is this estimation? How accurate do we need to be? How do we measure the accuracy? To get a handle on what we mean by accuracy, we can choose to measure errors by substituting the estimates of λ_3 and λ_4 (0.2 and 0.2, in this case) into the equations for α_3 and α_4 (equations (9) and (10)) and determining how close these are to $\hat{\alpha}_3$ and $\hat{\alpha}_4$ ($-.01218888030$ and 2.766540453 , in this case). Specifically, we take the error to be

$$\max(|\alpha_3 - \hat{\alpha}_3|, |\alpha_4 - \hat{\alpha}_4|). \tag{16}$$

For the data we are considering, substitution of $(\lambda_3, \lambda_4) = (0.2, 0.2)$ into (9) and (10) gives 0 and 2.706008524, respectively, and

$$\max(|\alpha_3 - \hat{\alpha}_3|, |\alpha_4 - \hat{\alpha}_4|) = \max(.01218888030, .060531929) = .060531929.$$

To facilitate the fitting of GLDs to a variety of data sets, attempts have been made to provide tables of $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ for specified sets of $\hat{\alpha}_3$ and $\hat{\alpha}_4$. Karian and Dudewicz (2000), in their Appendix B, provide the most comprehensive tables, along with an algorithm for using them. If these tables and associated algorithm are used for our data set, we get estimates of

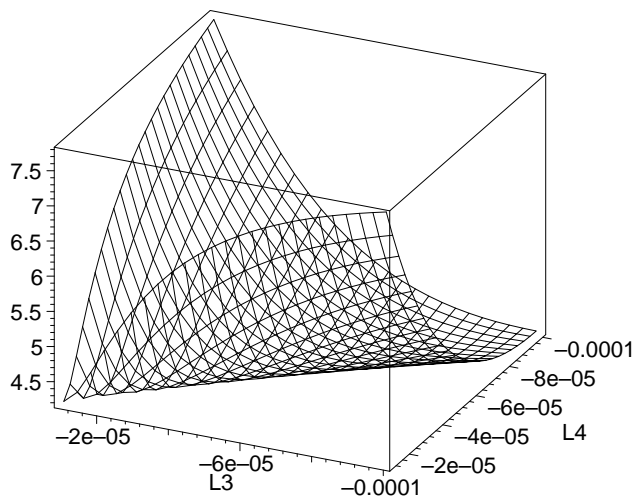


Figure 11. High-precision rendering of α_4 .

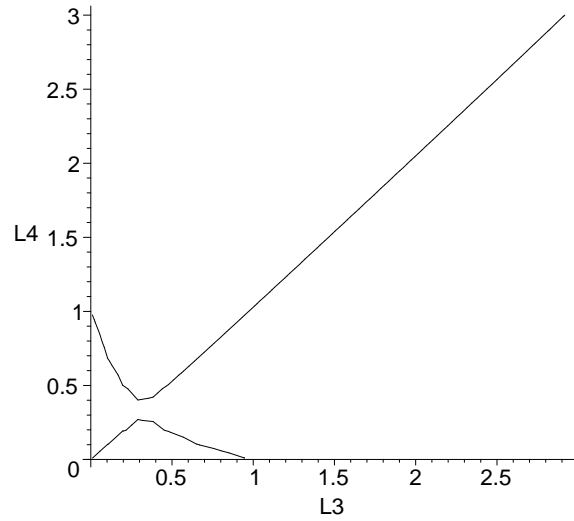


Figure 12. Contour of $\alpha_3 = \hat{\alpha}_3$ (quadrant I).

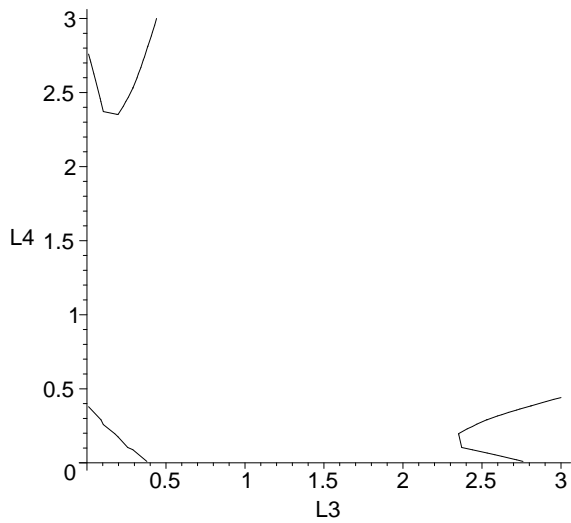


Figure 13. Contour of $\alpha_4 = \hat{\alpha}_4$ (quadrant I).

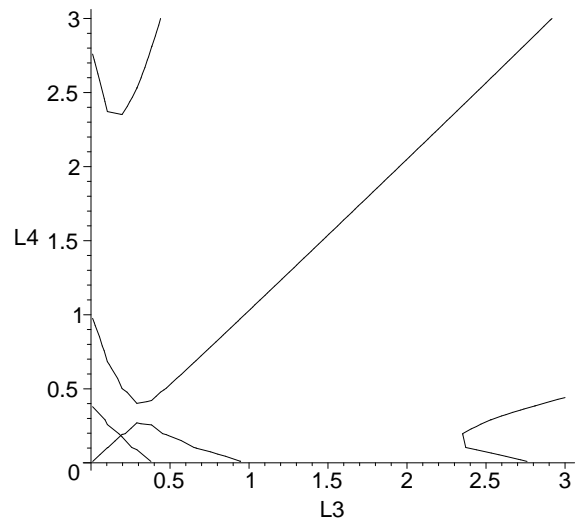


Figure 14. Superimposed Figures 12 and 13.

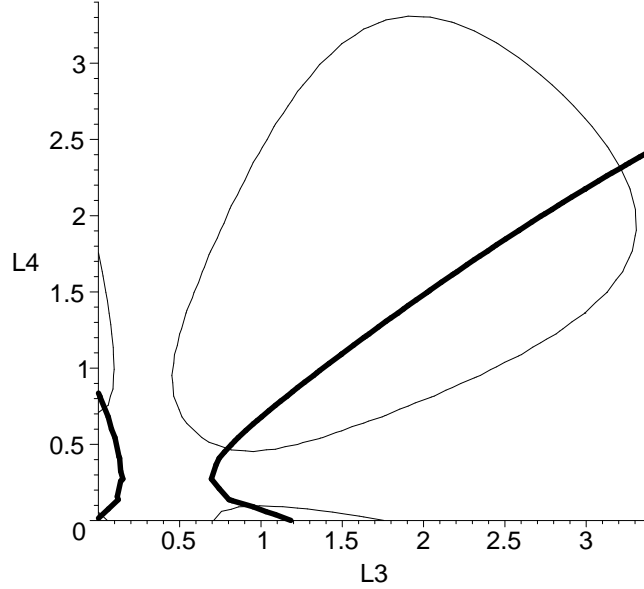


Figure 15. Contour plots of $\alpha_3 = 0.15$ and $\alpha_4 = 2$ (thick curves are for $\alpha_3 = 0.15$).

0.1765 and 0.1765 for λ_3 and λ_4 , respectively, and

$$\max(|\alpha_3 - \hat{\alpha}_3|, |\alpha_4 - \hat{\alpha}_4|) = \max(.01218888030, .033689895) = .033689895,$$

which is a marginal improvement over the graphic solution obtained earlier.

By their nature, graphic solutions lack precision but it is not wise to dispense with them altogether. Graphic analysis not only can guide the investigator's intuition, but can also uncover solutions that may have gone unnoticed. Consider a situation in which $\hat{\alpha}_3 = 0.15$ and $\hat{\alpha}_4 = 2$. Using the table mentioned earlier, we get the solution $\lambda_3 = 0.03145$ and $\lambda_4 = 0.7203$. While this is correct, several other solution have been missed. Figure 15 gives the contour plots of the of the α_3 and α_4 surfaces with $\alpha_3 = 0.15$ and $\alpha_4 = 2$ where the α_4 curves are rendered with a thick line. It looks like, in addition to the tabled solution, there are solutions near the (λ_3, λ_4) points $(.05, .05)$, $(.75, .5)$, $(3.2, 2.4)$, and $(.8, .2)$.

The main disadvantage of graphic and table driven solutions is their lack of accuracy, at least by measures such as the one established in (16). It makes sense, whenever possible, to estimate λ_3 and λ_4 through direct computation. The most efficient algorithms for finding solutions on surfaces start searching for a solution at a "reasonable" point and use gradient

vectors to move in the optimal direction to find a better solution. This is continued until a predetermined level of accuracy is achieved. Such algorithms work best if the surfaces are reasonably smooth; but even then, on some occasions, at some step within the algorithm the search can get misdirected and, once misdirected, never establish a path to a proper solution.

An alternative to a gradient-based algorithms is a “brute force” approach where a rectangular grid is established in (λ_3, λ_4) -space and grid points are tested until one with acceptable error is found. Of course, if the original grid does not cover the area where there is a solution, the search will not succeed; thus, one needs to have a good idea of where to conduct the search. Even if the grid contains a solution, the search can fail because the grid may be too coarse to come across a point that is within the desired error limit. Making the grid overly fine in order to counteract this problem may not work because too fine a grid may result in an intolerably long computation time. For example, moving from a 10×10 grid to a 25×25 grid increases computation time by more than a factor of 6 and moving to a 250×250 grid would increase a computation time by a factor of 625. If the original 10×10 grid required 5 seconds of computation time, the 250×250 grid would require about 52 minutes of computation.

A more acceptable algorithm can be designed to read table entries in order to determine the proper area that needs to be covered by a grid. After all, the smaller this area, the fewer grid lines should be required for a close search. In addition, the grid search itself could be improved. Instead of covering the search area by a 1000×1000 grid, the algorithm could start with a 10×10 grid (let’s say with increments of Δ_3 and Δ_4 between successive grid lines), find the best solution $(\lambda_3, \lambda_4) = (l_3, l_4)$ within that grid and restart a 10×10 grid search over the region

$$l_3 - \Delta_3 \leq \lambda_3 \leq l_3 + \Delta_3, \quad l_4 - \Delta_4 \leq \lambda_4 \leq l_4 + \Delta_4,$$

with a suitably smaller set of Δ_3 and Δ_4 . If this “re-gridding” is repeated three times, it would almost have the same effect as an initial 1000×1000 grid search. If we again assume that a 10×10 grid requires 5 seconds of computation time, this new approach would take

about 15 seconds whereas a 1000×1000 grid would need almost 14 hours.

The platform to use for data fitting computations is another issue that must be dealt with. Maple provides a rich computing environment that makes graphic, numeric, and symbolic computing available to the user. Maple is also structured as a programming language, making it possible to write programs in Maple that can use any of the data or mathematical constructs that are built into Maple. The two disadvantages of Maple are that it is generally very slow when intensive computations are required and it is not the computational tool of choice for most statisticians. For these reasons, programs based on the concepts of the preceding paragraph have been written in **R**, one of the more popular computing systems used by statisticians. As in Maple, the “>” symbol in **R** is a prompt for a command to be entered. There are three programs in connection to a moment-based GLD fit that are relevant here: `FitGLDM(Data)`, `ChiSqGLD(Data, Lambda, Breaks)` and `KSGLD(Data, Lambda)`. The first of these, in addition to finding $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, gives some details related with the fit; the second, requires the data, the λ -vector of the fit and class boundary points as input and returns the chi-square statistic and p -value associated with it; the third program requires the data and the λ -vector and computes the Kolmogorov-Smirnov statistic for the fit. The actual use of these programs is given below.

```
> L <- FitGLDM(Data)
```

```
GLD MOMENT-BASED FIT
```

```
Alphas : 5.366585 1.203253 -0.01218888 2.766540
```

```
Error : 2.37996e-09
```

```
Lambdas: 5.39041 0.2293276 0.1883876 0.1807214
```

```
Support: ( 1.029834 , 9.750985 )
```

```
Data Min-Max: 2.81 8.14
```

```
> ClassBdry <- c(3.955, 4.455, 4.955, 5.455, 5.955, 6.455, 6.955)
```

```
> ChiSqGLD(Data, L, ClassBdry)
```

```

ClassBdry :
  3.955 4.455 4.955 5.455 5.955 6.455 6.955
Observed Frequencies:
  15 11 18 19 22 22 7 9
Expected Frequencies:
  12.64186 12.98157 18.25547 21.30907 20.73148 16.73852 11.08550 9.25653
Chi-Square = 4.240417 p-Value = 0.2366464
[1] 4.2404169 0.2366464

> KSGLD(Data, L)
[1] 0.04824023

```

The purpose of the `L <-` part of the first command in the above is to give a name to the λ -vector of the GLD fit so that it can be used in subsequent commands. The output of the **R** command `L <- FitGLDM(Data)` consists of

1. Alphas or $\hat{\alpha}_1$, $\hat{\alpha}_2$, $\hat{\alpha}_3$, and $\hat{\alpha}_4$.
2. Error, as defined in (16), indicating $\max(|\hat{\alpha}_3 - \alpha_3|, |\hat{\alpha}_4 - \alpha_4|) = 2.37996 \times 10^{-9}$.
3. The λ -vector specifying the GLD fit.
4. The support of the fitted distribution; it's always worth checking if the fitted distribution covers the data range.
5. The minimum and maximum values of the data.

`ClassBdry` defines the class boundaries for the chi-square test that will be performed in the next command, `ChiSqGLD(Data, L, ClassBdry)`. The `ChiSqGLD` program needs to make use of the data, the λ -vector (previously given the name `L`), and the class boundaries. The results of this command are: the list of observed and expected frequencies for each class defined by `ClassBdry` and the chi-square statistic with its associated p -value. The last

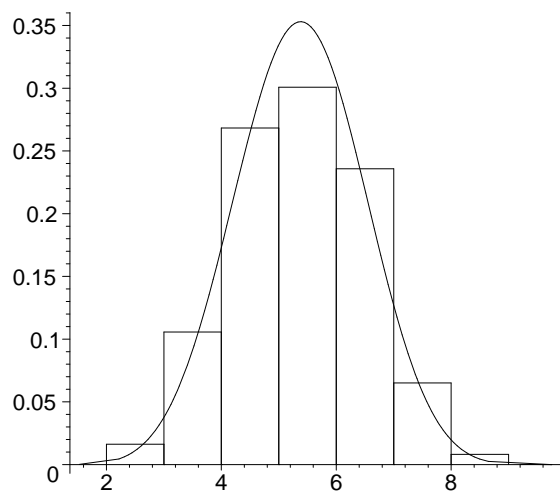


Figure 16. A data histogram and the fitted GLD pdf.

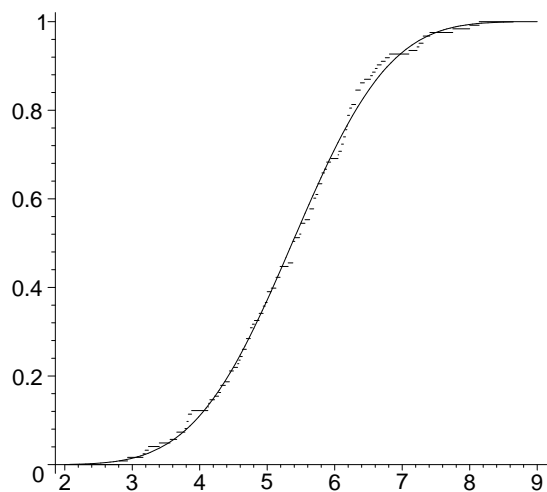


Figure 17. The empirical cdf with the fitted GLD cdf.

program used, `KSGLD(Data, L)`, computes and returns the Kolmogorov-Smirnov statistic for this fit.

Figure 16 displays a histogram of the data with the fitted GLD pdf superimposed and Figure 17 shows the empirical cdf with the superimposed GLD cdf. These figures confirm that the GLD fit indeed is a good one.

4. USE OF JOHNSON'S SYSTEM

Distributions from the Johnson system (Johnson (1949)), have been used to fit data sets by a number of investigators (Johnson (1965), Leslie (1959), Mage(1980)). This system provides distributions with all possible values of α_3 and α_4 . The distributions in this system are derived from the standard normal distribution by setting

$$Z = \gamma + \delta f(Y) \quad (17)$$

where Z is the standard normal random variable. Specific choices of γ , δ and the function $f(Y)$ completely define the random variable Y . The general Johnson system random variable, X , is defined through

$$Y = (X - \xi)/\lambda, \quad (18)$$

giving X flexibility in location and scale. The three classes of distributions that comprise

the Johnson system, designated by S_L (lognormal family), S_B (bounded family) and S_U (unbounded family), result from the choices

$$f(Y) = \begin{cases} \log(Y) & \text{for } S_L \\ \log\left(\frac{Y}{1-Y}\right) & \text{for } S_B \\ \sinh^{-1}(Y) & \text{for } S_U \end{cases} \quad (19)$$

for $f(Y)$. These three families of distributions uniquely cover the entire (α_3, α_4) -space that is possible (see Dudewicz, Zhang and Karian (2004)).

The (α_3, α_4) -space associated with these families is illustrated in Figure 18. Computations associated with fits in the S_L and S_B regions are considerably simpler than those for S_U and Johnson (1965) provides extensive tables for γ and δ for S_U . Since the central focus of this paper is computation and the S_B family poses by far the most challenging computational problems, the discussion that follows will be restricted to S_B . The (α_3, α_4) -space for S_B is bounded below by $\alpha_4 = 1 + \alpha_3^2$ (it is impossible to have points below this curve for any distribution). It is also bounded above by the curve that represents the S_L family, which can be defined (through the parameter $\omega > 1$) by

$$\alpha_3 = \sqrt{\omega - 1}(\omega + 2) \quad (20)$$

$$\alpha_4 = \omega^4 + 2\omega^3 + 3\omega^2 - 3. \quad (21)$$

The pdf of Y for S_B is given by

$$p_Y(y) = \frac{\delta}{\sqrt{2\pi} y(1-y)} \exp\left\{-\frac{1}{2}[\gamma + \delta \log(y/(1-y))]^2\right\} \quad (22)$$

and the pdf of X by

$$p_X(x) = \left(\frac{\delta\lambda}{\sqrt{2\pi}(x-\xi)(\lambda-x+\xi)}\right) \exp\left\{-\frac{1}{2}[\gamma + \delta \ln((x-\xi)/(\lambda-x+\xi))]^2\right\}. \quad (23)$$

The moments of Y are given by

$$E(Y^m) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \{1 + \exp[-(Z - \gamma)/\delta]\}^{-m} \exp(-Z^2/2) dZ \quad (24)$$

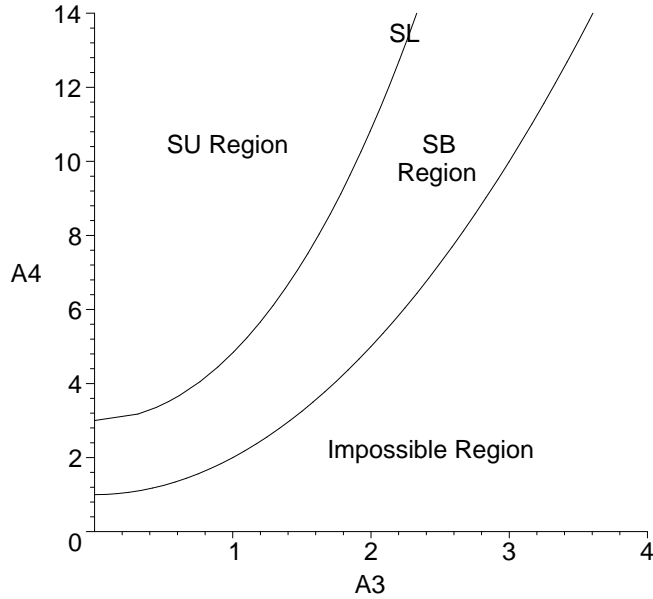


Figure 18. (α_3, α_4) , designated by A3 and A4, respectively, of the Johnson system families.

and since the (α_3, α_4) of X are the same as those for Y , γ and δ could (at least in theory) be determined through (24). Once this is done, finding ξ and λ becomes relatively simple (one computes the mean and variance of Y from (24) and uses (18) to determine ξ and λ).

For specified γ and δ , to compute α_3 and α_4 we need to first obtain $E(Y)$, $E(Y^2)$, $E(Y^3)$, and $E(Y^4)$ from (24), all of which are defined as indefinite integrals. Bukač (1972) and Mage (1980) propose percentile based methods for estimating γ and δ . For fits that use the method of moments, Johnson and Kitchen (1971) describe a scheme for evaluating $E(Y^m)$ by breaking the integral of (24) into three parts and using quadrature methods for evaluating the most difficult of the three integrals. Based on this approach, they provide a very limited set of tables.

Reasonable approximations can be found but at a rather substantial cost in computation time, making it infeasible to even render 3-dimensional surface plots or contour plots of α_3 and α_4 . Exploratory computations at various (γ, δ) pairs indicates that there are regions in (γ, δ) -space where dramatic changes occur in α_3 and α_4 , particularly α_4 , with rather modest variations in γ or δ (i.e., the α_4 surface is steep). In other portions of (γ, δ) -space

the opposite is true: significant alterations in (γ and δ) result in only modest variations in α_3 or α_4 (i.e., the surfaces are flat). These two problems make grid-based searches complicated. For example, if in the first round of a grid search in a region where α_4 is alternately rising and then somewhat flat, we get (g, d) as the optimal point, there will be no assurance that the true optimum is in the rectangle

$$g - \Delta_g \leq \gamma \leq g + \Delta_g, \quad d - \Delta_d \leq \delta \leq d + \Delta_d$$

where Δ_g and Δ_d are the grid increments that are being used. To get around this problem, the new sub-rectangles for subsequent grids are defined from old ones through

$$g - 3\Delta_g \leq \gamma \leq g + 3\Delta_g, \quad d - 3\Delta_d \leq \delta \leq d + 3\Delta_d.$$

This necessary adjustment makes an already slow computation less efficient and even slower.

Tables for γ and δ (far more detailed than the ones given by Johnson and Kitchen (1971)), for specified $\hat{\alpha}_3$ and $\hat{\alpha}_4$, developed through a laborious process are given in Appendix A. These tables cover a similar portion of (α_3, α_4) -space – with different emphasis of detail – as those given in Pearson and Hartley (1971) and for all table entries

$$\max(|\alpha_3 - \hat{\alpha}_3|, |\alpha_4 - \hat{\alpha}_4|) < 10^{-3}.$$

The α'_1 and α'_2 columns are the mean and standard deviation of Y and have to be adjusted before calculating ξ and λ .

The Maple program, `FitJnsnSB`, listed in Appendix B, uses the tables of Appendix A to start the search on a reasonably small rectangle. The first argument to be provided to `FitJnsnSB` is the α -vector; the second is the number of partitions to use along both γ and δ axes; the third argument specifies the number of re-griddings that should be used. The search is terminated when a point with error $\leq 10^{-4}$ is found or when the stipulated number of iterations are exhausted. It is also possible to use `FitJnsnSB` with five arguments where the first three arguments are as described above and the fourth and fifth arguments establish the search intervals for δ and γ , respectively.

The following Maple interaction shows how the data given at the beginning of this paper can be fitted with a distribution from the S_B region of the Johnson system.

```
A:=FindAlphas(Data);
```

```
A := [5.366585366, 1.203253381, -0.01218888030, 2.766540453]
```

```
> J:=FitJnsnSB(A,13,5);
```

```
J := [-0.605831231, 12.09398259, 2.666126613, 0.06800172372, 0.000098774]
```

```
> pdf:=JnsnPDF(A,x,J);
```

```
[-.75498063, 11.33900196]
```

$$\frac{32.24408884}{\sqrt{2\pi}(11.33900196 - x)(0.75498063 + x)} e^{-0.5 \left(0.06800172372 + 2.666126613 \ln\left(\frac{11.33900196 - x}{0.75498063 + x}\right)\right)^2}$$

```
> [Min(Data), Max(Data)];
```

```
[2.81, 8.14]
```

```
> Fitcdf:=int(pdf, x=-0.75498063..t):
```

```
> ClassBdry:=seq(3.955+.5*i, i=0..6);
```

```
ClassBdry := [3.955, 4.455, 4.955, 5.455, 5.955, 6.455, 6.955]
```

```
> Probs:=evalf([0,seq(subs(t=ClassBdry[i],Fitcdf), i=1..7),1]);
```

```
Probs := [0., 0.1026174338, 0.2087262954, 0.3573739943, 0.5301939982,  
0.6981139332, 0.8340908941, 0.9246651330, 1.]
```

```
> EF:=seq(nops(Data)*(Probs[i+1]-Probs[i]), i=1..8);
```

```
EF := [12.62194436, 13.05138998, 18.28366696, 21.25686048, 20.65415200,  
16.72516619, 11.14063138, 9.266188641]
```

```
> OF:=[15,11,18,19,22,22,7,9]:
```

```
> ChiSq:=sum((EF[i]-OF[i])^2/EF[i], i=1..8); pVal:=1-ChisquareCDF(3,ChiSq);
```

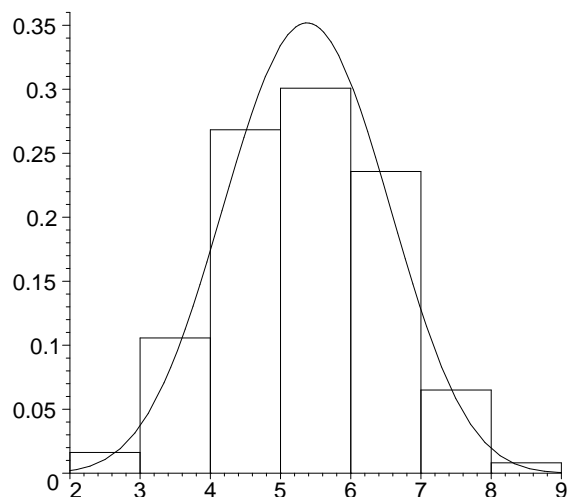


Figure 19. Fitted Johnson system pdf with histogram.

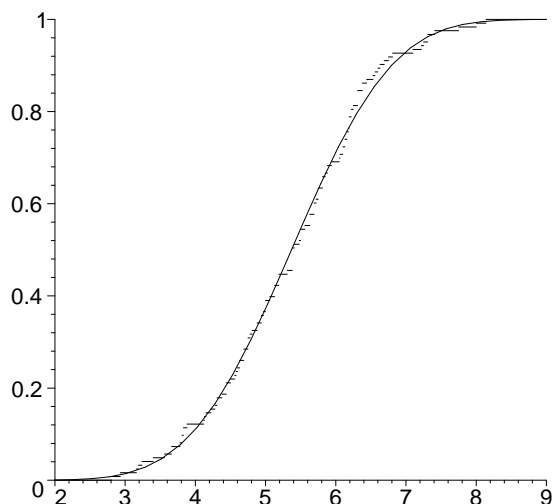


Figure 20. Fitted Johnson system cdf with empirical cdf.

$$ChiSq := 4.312370869$$

$$pVal := 0.2296494776$$

```
> KS(Data,Fitcdf);
```

```
0.0485424361
```

Note that in the above, `FitJnsnSB` assigns to `J` the vector $(\xi, \lambda, \delta, \gamma)$, a vector that is appropriate for $(\hat{\alpha}_1, \hat{\alpha}_2, |\hat{\alpha}_3|, \hat{\alpha}_4)$. Since in this case, $\hat{\alpha}_3 < 0$, we cannot get the Johnson pdf by simply substituting these values of ξ , λ , δ , and γ into (23). The pdf obtained from `JnsnPDF` is correct because, knowing that $\hat{\alpha}_3 < 0$, the `JnsnPDF` program adjusts for the negative α_3 by applying the transformation: $x \rightarrow 2\hat{\alpha}_1 - x$ to the pdf, before returning it to the user. `JnsnPDF` also prints out the support of the distribution (in this case, $[-.75498063, 11.33900196]$) prior to giving the pdf. As the results of the next line indicates, the support covers the data range.

From the histogram of the data with the fitted pdf, shown in Figure 19, and the empirical cdf plotted with the fitted cdf, shown in Figure 20, we expect that the Johnson system pdf is a good fit. As was done in the Weibull fit, the cdf is obtained next and class boundaries are

defined, leading us to the chi-square statistic and its p -value. As a last step, the Kolmogorov-Smirnov statistic is computed. The chi-square and Kolmogorov-Smirnov statistics confirm the visual indication of Figures 19 and 20 that the Johnson system fit is a good one.

5. CONCLUSIONS

To be able to estimate parameters associated with a fit, one needs to be able to develop a package of programs in some computing environment. The choices made in this paper, Maple and **R**, are somewhat arbitrary; Mathematica could have been used instead of Maple and some other statistical system could have been used instead of **R**. The choice of a computing platform is typically based on issues such as the general availability of the platform and the ease with which the necessary computations can be implemented in the chosen environment. With little effort, the Weibull fit could have been implemented in **R** instead of Maple. With considerably greater effort, packages have been developed in Maple and in **R** for fitting GLD distributions (the one in **R** was illustrated in Section 3; see Karian and Dudewicz (2000) for the one in Maple). For Johnson fits in the S_B region, it would be substantially more difficult and time consuming to develop the necessary programs in **R**.

Clearly, both Maple and **R** have their distinct advantages: the powerful computing environment for Maple and the almost universal availability (because various statistical features) of **R** among statisticians. Why can't we have it all? or why can't **R** implement symbolic processing features and arbitrary precision computations similar to Maple? The main reason why we don't have it all, is that it would take a substantial effort to incorporate such features in **R**; another reason may be that those involved with distributing **R** are not aware that such a need exists, or if they are aware, it is not very high on their priority list. Until a more robust computing environment is developed we will continue to have these difficulties. The only hope is that through some sort of corporate agreement, Maple and **R** can forge a merged product incorporating the desirable features of each.

REFERENCES

- Bukač, J. (1972). “Fitting S_B Curves Using Symmetrical Percentile Points,” *Biometrika*, 59, 688–690.
- Burr, I.W. (1973). “Parameters for a General System of Distributions to Match a Grid of α_3 and α_4 ,” *Communications in Statistics*, 2, 1–21.
- Dudewicz, E.J. and Karian, Z.A. (1996). “The Extended Generalized Lambda Distribution (EGLD) System for Fitting Distributions to Data with Moments, II: Tables,” *American Journal of Mathematical and Management Sciences*, 16, 271–332.
- Dudewicz, E.J. and Karian, Z.A. (1999). “Fitting the Generalized Lambda Distribution (GLD) System by a Method of Percentiles, II: Tables,” *American Journal of Mathematical and Management Sciences*, 19, 1–73.
- Dudewicz, E. J., Zhang, C. X. and Karian, Z. A. (2004). “The Completeness and Uniqueness of Johnson’s System in Skewness-Kurtosis Space,” *Communications in Statistics: Theory and Methods*, 33.
- Gilchrist, W.G. (2000). *Statistical Modeling with Quantile Functions*, CRC Press, Boca Raton, Florida.
- Hahn, G.J. and Shapiro, S.S. (1967). *Statistical Models in Engineering*, John Wiley and Sons, New York.
- Johnson, N.L. (1949). “System of Frequency Curves Generated by Methods of Translation,” *Biometrika*, 36, 149–176.
- Johnson, N.L. (1965). “Tables to Facilitate Fitting S_U Frequency Curves,” *Biometrika*, 52, 547–571.
- Johnson, N.L. and Kitchen, J.O. (1971). “Some Notes on Tables to Facilitate Fitting S_B Curves,” *Biometrika*, 58, 223–226.
- Karian, Z.A. and Dudewicz, E.J. (1999). “Fitting the Generalized Lambda Distribution

to Data: A Method Based on percentiles,” *Communications in Statistics: Simulation and Computation*, 28, 793–819.

Karian, Z.A. and Dudewicz, E.J. (2000). *Fitting Statistical Distributions: The Generalized Lambda Distribution and Generalized Bootstrap Methods.*, CRC Press, Boca Raton, Florida.

Karian, Z.A. and Dudewicz, E.J. (2003). “Comparison of GLD Fitting Methods: Superiority of Percentile Fits to Moments in L^2 Norm,” *Journal of the Iranian Statistical Society*, 2, 171–187.

Karian, Z.A., Dudewicz, E.J. and McDonald, P. (1996). “The Extended Generalized Lambda Distribution System for Fitting Distributions to Data: History, Completion of Theory, tables, Applications, the ‘Final Word’ on Moment Fits,” *Communications in Statistics: Simulation and Computation*, 25, 611–642.

Leslie, D.C.M. (1959). “Determination of Parameters in the Johnson System of Probability Distributions,” *Biometrika*, 46, 229–243.

Mage, David T. (1980). “An Explicit Solution for S_B Parameters Using Four Percentile Points,” *Technometrics*, 22, 247–251.

Öztürk, A. and Dale, R.F. (1985). “Least Squares Estimation of the Parameters of the Generalized Lambda Distribution,” *Technometrics*, 27, 81–84.

Pearson, E.S. and Hartley, H.O. (1971). *Biometrika Tables for Statisticians*, Volume II, Cambridge University Press.

Ramberg, J.S., Dudewicz, E.J., Tdikamalla, P.R. and Mykytka, E.F. (1979). “A Probability Distribution and Its Uses in Fitting Data,” *Technometrics*, 21, 201–214.

Ramberg, J.S. and Schmeiser, B.W. (1972). “An Approximate Method for Generating Symmetric Random Variables,” *Comm. ACM*, 15, 987–990.

Ramberg, J.S. and Schmeiser, B.W. (1974). “An Approximate Method for Generating Asymmetric Random Variables,” *Comm. ACM*, 17, 78–82.

Appendix B: Maple Code for FitJnsnSB

```

FitJnsnSB := proc(Alphas::list, Grid::posint, Iter::posint, PT1::list, PT2::list)
    local Q, B1List, B2List, A3, TA, TB, NRows, pt1, pt2, pt3, pt4, Pt1, Pt2,
        f, EYm, mu, EY2, EY3, EY4, s, B1, B2, s3, s4, f3, f4, BestErr,
        K, k, l, Delta3, Delta4, i, j, BB1, BB2, Err1, Err2, Err, mean,
        sd, lambda, c, InList, q, qq, FileNo1, FileNo2, FileName1, FileName2;

    if nargs < 5 then

        B1List:= [0, .05, seq(0.1*i, i=1..20)];
        A3 := abs(Alphas[3]):
        if A3 < B1List[1] or A3 > B1List[nops(B1List)] then
            RETURN([0,0,0,0])
        fi:
        j:=0:
        for i from 1 to nops(B1List) do
            if A3 > B1List[i] then j:=j+1 fi:
        od:
        FileNo1:=j; FileNo2:=j+1:
        FileName1:="u:\\texfiles\\R\\JnsnSB"||FileNo1;
        FileName2:="u:\\texfiles\\R\\JnsnSB"||FileNo2;
        read(FileName1): TA:=%:
        read(FileName2): TB:=%:

        NRows:=nops(TA)/3;
        B2List:= [seq(TA[1+3*i], i=0..NRows-1)];
        if Alphas[4] < TA[1] or Alphas[4] > TA[3*NRows-2] then
            RETURN([0,0,0,0])
        fi:
        j:=1:
        while Alphas[4] > B2List[j] do j:=j+1 od:
        pt1:= [TA[3*(j-1)+2], TA[3*(j-1)+3]];
        pt2:= [TA[3*(j-2)+2], TA[3*(j-2)+3]];
        NRows:=nops(TB)/3;
        B2List:= [seq(TB[1+3*i], i=0..NRows-1)];
        if Alphas[4] < TB[1] or Alphas[4] > TB[3*NRows-2] then
            RETURN([0,0,0,0])
    end if;
end proc;

```

```

fi:
j:=1:
while Alphas[4] > B2List[j] do j:=j+1 od:
pt3:=[TB[3*(j-1)+2], TB[3*(j-1)+3]];
pt4:=[TB[3*(j-2)+2], TB[3*(j-2)+3]];
Pt1:=[min(pt1[1],pt2[1],pt3[1],pt4[1]), max(pt1[1],pt2[1],pt3[1],pt4[1])];
Pt2:=[min(pt1[2],pt2[2],pt3[2],pt4[2]), max(pt1[2],pt2[2],pt3[2],pt4[2])];

else Pt1:=PT1: Pt2:=PT2 fi:

f:=(1/sqrt(2*Pi))*(1+exp(-(z-Gamma)/delta))(-m)*exp(-z2/2):
EYm:=int(f, z=-infinity..infinity):
mu:=evalf(subs(m=1,EYm)):
EY2:=evalf(subs(m=2,EYm)):
EY3:=evalf(subs(m=3,EYm)):
EY4:=evalf(subs(m=4,EYm)):
s:=sqrt(EY2-mu2):
B1:=(EY3-3*EY2*mu+2*mu3)/s3:
B2:=(EY4-4*EY3*mu+6*EY2*mu2-3*mu4)/s4:
s3 := Pt1[1]; f3 := Pt1[2]; s4 := Pt2[1]; f4 := Pt2[2];
BestErr := 100000; K:=1;
while BestErr > 0.0001 and K <= Iter do
  Delta3 := (f3-s3)/(Grid); Delta4 := (f4-s4)/Grid;
  for i from s3 to f3 by Delta3 do
    for j from s4 to f4 by Delta4 do
      BB1:=subs({delta=i,Gamma=j}, B1):
      BB2:=subs({delta=i,Gamma=j}, B2):
      Err1 := evalf(abs(BB1-abs(Alphas[3]))):
      Err2:=evalf(abs(BB2-Alphas[4])):
      Err := max(Err1, Err2):
      if Err < BestErr then
        k:=i: l:=j: BestErr:=Err
      fi:
    od:
  od:
  K := K+1;
  s3 := k-3*Delta3; f3 := k+3*Delta3;
  s4 := l-3*Delta4; f4 := l+3*Delta4;

```

```
od:
mean := evalf(subs({delta=k, Gamma=1}, mu)):
sd := evalf(subs({delta=k, Gamma=1}, s)):
lambd := sqrt(Alphas[2]/(sd^2));
c := Alphas[1] - lambd*mean;
evalf([c, lambd, k, 1, BestErr])
end:
```