Abstract

One of the hardest things to predict is the outcome of a sporting event. There is an indisputable factor of unpredictability. It's the reason we have great upsets and the reason results aren't predetermined by statisticians, but it's also the reason betting parlors are so successful. For centuries people have tried to accurately gauge potential victories and losses of a player or a team, and for just as long no one has been purely successful. This is due to the constant unpredictability. In this paper, we try to get as close as possible to eliminating that unpredictability. Using statistical analysis, we identify which variables are most closely correlated with success in a competitive fitness sport, and rank competitors according to how well they perform in said variables. We use the resulting scores to calculate the probability of winning using a logistic function, in what we believe can be considered a successful attempt at minimizing the profits of betting parlors.

The Problem

Over the course of this Semester, we worked to come up with a viable solution to the problem posed to us by Jason Heidecker at BeastScore.com. We were given data on the performance of 478 competitors and their placements in a national competition, and were asked to formulate a scoring system, or a "Beast Score" that can accurately predict future performance. We were additionally asked to create a 100x100 table of probabilities where the cell B(i,j) represents the probability that someone with a Beast Score of i beats someone with a Beast Score of j. The data set was comprised of 9 nine different variables: Timed Workout F, Timed Workout G, Timed Workout H, 400 meter sprint, Weight lift B, Weight lift C, Weight lift D, Weight lift S, and Age. Aside from the 400 meter sprint and Age categories, the details of each exercise remained undisclosed.

Initial Approach

We examined various ranking and scoring systems across athletic, academic and recreational events and decided that we wanted our model to resemble the Elo Rating System used in Chess, in which the probability of a player winning is based on the difference in scores between the two players. For any given matchup, the probability of the two players beating each other always adds up to 1. Observe the following logistic formulas, used to calculate the probability of player A beating player B, and vice versa, where D represents the difference in rating between the two players[13]:

$$E_A = \frac{1}{1+10^{D/400}} \quad (1A)$$

$$E_B = \frac{1}{1+10^{D/400}} \quad (1B)$$

We felt that given the nature of our problem, a logistic formula similar to these would be appropriate.

Before we could approach the idea of a logistic formula, however, we needed to come up with a scoring system for our competitors. We used S-Plus to generate correlation coefficients for each exercise E_1 through E_n with respect to placement in the national competition, denoted by C_1 through C_n, and used these values to come up with the following formula:

$$BeastScore = 100 * \sum_{i=1}^{n} \left(\frac{C_i E_i}{C_{sum} \text{"Max } E_i\text{"}}\right) \quad (2)$$

where C_{sum} is the sum of all correlation coefficients.

We ran across a few issues with this scoring system. For one, it had an R^2 value ≈0.5. R^2, also known as the coefficient of determination, is a statistical measure of how well one's actual data is fitted to a model. An R^2 value of 1.0 would indicate that the model explains any and all of the variability of the

data set and fits the data extremely well[12]. An R^2 of 0.5 is not high enough to say that our model fits our data well, if at all. We observed the effects of this R^2 value through the major inconsistencies we ran across when running our model. The weakness of our initial model can be seen in Figure 1. Another issue we had was that our model was geared toward a normally distributed data set, which our data set wasn't. And finally, we had yet to remove outliers at this point, and we hypothesized that this likely had something to do with the previously mentioned inconsistencies.

It was at this point that we decided the first step to improving our model should be to remove outliers. To do this, we examined the scatter plots of our 9 variables with respect to placement, and observed a rough line of best fit for each plot. We then decided, somewhat arbitrarily which few points from each plot we would remove, shown in Figure 2 and Figure 3. We removed a total of 26 competitors, or roughly 5% of our original data set of 478.

Normalizing in S-plus

While examining our nine categories against placement to determine outliers we observed that there were no clear attributes related to a specific type of distribution. We then consulted Professor Tamas Lengyel at Occidental College [6]. After examining our data he gave us a S Plus function that he developed which generates a power-normal transformation. This function is called l9thp (See Appendix B, Program 1). First l9thp plots diagnostic graphs of the distribution using four different graphical techniques, as shown by the example in Figure 4 for workout G. The first, going for left to right, is a histogram showing the distribution in the data. The next is a boxplot showing data points in the interquartile area which is the area denoted by the shaded box [2]. This area represents the data points which make up fifty percent of the total distribution. The solid line represents the mean of the distribution. The two leafs or straight lines connected to the shaded box are the points which are within twenty five percent of the interquartile range and the lines beyond these leafs are points that lie beyond

seventy five percent of the interquartile range which makes them outliers. The third graph is a probability distribution which in Figure 4 the distribution exhibits non-normal tendencies by being skewed [1]. The last and most important graph is the empirical quantile-quantile plot between the workout and the standard normal distribution [2]. The standard normal is represented by the straight line diagonal line and the distribution of our workout is denoted by the collection of points. This specific type of graphical technique is specifically used to compare two distributions so in our case we want our data to fit the diagonal line generated in this graph. L9thp then utilizes the log likelihood function to plot observations for a data set y raised to a specific power x against that same power x. In our case y can be treated as one of our nine categories and x can be any real number between negative one and one. It then provides us with the optimal value of x to make our data resemble a normal distribution by taking the maximum value of the curve as seen in Figure 5. We then test to see if this transformation provides us with a normal distribution so l9thp generates a new set of diagnostic graphs and we can see from the fourth graph in Figure 6 that our data fits the diagonal which assures us that this distribution is now normal.

P-Statistics and BeastScore Formula

By transforming our data we then utilized a built-in function of S Plus called lm [7]. This function sets a linear fit that describes the relationship between our nine categories and placement (See Appendix B, Program 2). In addition lm provides us with the p statistic values for our nine categories. P statistics are a basis in hypothesis testing [11]. In our case there are two hypotheses. The first is that a change in one category will not change the outcome of a person's placement, which is called the null hypothesis. The second is that a change in one category will change the outcome of a person's placement, which is called the alternative hypothesis. We then chose a statistical significance value which is the likelihood that the null hypothesis will be correct. We chose this value to be five percent. To

summarize, if the p statistic is below five percent this means we can reject the null hypothesis and accept the alternative [11]. The function lm calculates these p values for each category by using the variance and sample mean with respect to placement. After reviewing our data we found five categories fit this criterion. These values of the p statistics for the five categories can be seen in Equation 3.

$$\text{Timed Workout F: } p\ value < 10^{-3}$$

$$\text{Timed Workout G: } p\ value = 0.003$$

$$\text{Timed Workout H: } p\ value < 10^{-3} \quad (3)$$

$$\text{Weightlift S: } p\ value < 10^{-3}$$

$$\text{Age: } p\ value = 0.007$$

The function lm also calculates the correlation coefficients of each of our five categories [7]. This allows us to construct a linear model to predict a person's placement in this competition given by Equation 4.

$$Placement^{.4} = Timed\ Workout\ F^{-1} + Timed\ Workout\ H^{-0.7} + Timed\ Workout\ G^{-0.6} + Weightlift\ S^{0.8} + Age \quad (4)$$

This linear model yielded an R^2 value of 0.85 which much more favorable than 0.5 which was the R^2 of our previous model. To assign individuals in our data set a BeastScore we inserted their performances in these five categories into Equation 4. This generated BeastScore values which ranged from -52 to -120. To generate a BeastScore ranging from 0 to 100 we simply transformed the placement values by using an affine transformation which changed the range to 30 to 90, with the exception of one person who had a BeastScore of 20. We figured this would be an acceptable range of BeastScores since we were dealing with data from nationally competitive athletes. We then plotted each individuals BeastScores vs

their placement to the .4 power, as seen in Figure 7 and were pleased that those with a higher BeastScore placed lower in the national competition which is what we were expecting. We then generalized Equation 4 to yield our generalized Beast Score formula, Equation 5.

$$BeastScore = \begin{pmatrix} 3777.5 * C_F^{-1} \\ +4647.6 * C_H^{-0.7} \\ -220.6 * C_G^{-0.6} \\ +0.507 * C_S^{0.8} - 0.332 * C_A \end{pmatrix} - 30 \quad (5)$$

$$C_F = Individual's\ Performance\ in\ Workout\ F$$

$$C_H = Individual's\ Performance\ in\ Workout\ H$$

$$C_G = Individual's\ Performance\ in\ Workout\ G$$

$$C_S = Individual's\ Performance\ in\ Workout\ S$$

$$C_A = Individual's\ Age$$

Calculating the Probability of "Winning"

Now that we had established a relationship between BeastScore and placement in the national competition which had a good R^2 value, our next step was to develop a probability model. As stated earlier, our goal was to determine the probability that a person with BeastScore A ranks higher in another competition than a person with BeastScore B, based on the difference A-B. To compute this probability, we decided that our best approach would be to consider comparisons between individual athletes as "games", where the winner of a game should be determined by whoever placed higher in the national competition. Therefore, if we wanted to determine the probability of an individual athlete with BeastScore A beating someone with BeastScore B, we would take the number of athletes in B that the individual A ranked higher than, and divide that by the total number of people in B. This probability can be referred to as the dominance percentage P_d. Dominance percentage is defined as "the percentage of the playing field dominated by the player in question, plus half the percentage which stands in no dominance relation with that player" [3]. In our case, one athlete dominates another if they placed

higher in the national competition. Dominance percentage can also be thought of as "the percentage score that would result from a single game against other players in the competing field" [3], or in our case, the probability that an athlete will place higher in a future competition. Our competing field is also defined by the players with BeastScores A and B. Normally, dominance percentage for a single player is given by the equation

$$P_d = \frac{W_d + .5D_d}{n} \tag{6}$$

where W_d is the number of opponents a player outperformed, D_d is the number of opponents a player tied with, and n is the total number of opponents in the given field [3]. There were no ties in our data, however, so we can remove the D_d term. Also, since this formula only considers a single athlete, we would need to average this percentage for each person of BeastScore A who "competed" against a group of people with BeastScore B.

Realistically, however, we couldn't use this idea for singular BeastScores due to the fact that our BeastScores vary over a wide range and are unevenly distributed amongst the range. For example, there are fifteen athletes with a BeastScore between 30 and 39 in our data, and over a hundred athletes with a BeastScore between 50 and 59. Therefore, we decided to divide the athletes into "buckets" of BeastScore ranges and compute the probability of one bucket placing higher than the other. This would be obtained by averaging the probability of all people in bucket A beating someone in bucket B. Thus, within our data, we evaluated the probability of someone in BeastScore bucket A ranking higher than someone with BeastScore bucket B in the national competition by the formula

$$P(A\ beats\ B) = \sum_{i=1}^{n_A} \frac{P_{d_i}}{n_A} = \frac{1}{n_A} \sum_{i=1}^{n_A} \frac{W_{d_i}}{n_B} \tag{7}$$

where W_{d_i} is the number of people in B that individual i in A ranked higher than, n_A is the number of people in A, and n_B is the number of people in B. For our data, we decided that it would be reasonable to use an interval of 5 for our buckets, so we had buckets of people with BeastScores from 30-34, 35-39, 40-44, up until the range 85-89. Also, there was only one person with a BeastScore of 20, who was the only person with a BeastScore under 30, so we omitted him from our probability calculations. In order to compute the probabilities across each bucket comparison, we used MATLAB to develop an algorithm and a program which computes all of the probabilities and then prints a 12 by 12 table of the probabilities (See Appendix C, Programs 1 and 2). The algorithm works by taking an input matrix which contains each athlete's BeastScore and their placement. Then, by defining bounds for the "winning" bucket and "losing" bucket, it counts the number of "wins" and number of "games" played to compute the probability and returns that value. The program then runs this algorithm across each bucket comparison, and it produces the table in Figure 8. As we can see from the table, our model appears to produce reasonable probabilities. Each entry (i, j) in the table represents the probability that a person belonging to the bucket at column j will beat a person belonging to the bucket at row i. We kept five decimal places of accuracy in these probabilities to keep consistency with our use of five decimal places in the power transformations of our data. Note that as the difference in BeastScores grows in the positive direction, the probability grows larger and approaches 1, and as the difference grows in the negative direction, the probability grows smaller and approaches 0. In addition, the opposite entries in the table add up to 1, so $P(A) + P(B) = 1$ for each probability comparison in our model.

The Probability Formula

With this table of probabilities, we proceeded to create a single formula to calculate the probability of ranking higher in a future competition based on the difference between two athletes' BeastScores. This new formula would be obtained by taking the average probability for each bucket

difference in our earlier table, plotting probability vs. difference into Excel, and then obtaining an equation for a line of best fit. To begin deriving this formula, note that the diagonals in the previous table represent an average difference in BeastScore. For example, the main diagonal represents an average difference of 0, since we are comparing the buckets against themselves. The super-diagonal represents an average difference of +5, the sub-diagonal represents -5, and the average difference increments by 5 each direction. Therefore, we created another program in MATLAB to average the probabilities along each diagonal to obtain probabilities corresponding to differences in BeastScore (See Appendix C, Program 3). Since we were extending our probability model beyond our given data, we decided to use four decimal places of accuracy for our calculated probabilities to account for any potential errors outside of our given system. Plugging this data into Excel, we obtained the scatterplot in Figure 9 for Probability vs. BeastScore. Clearly, this plot represents a logistic relationship. Then, using Mathematica's built-in LogitModelFit function to create a binomial logistic regression model [8], we obtained the best-fit probability formula that spans the entire BeastScore range:

$$P(x) = \frac{1}{1+e^{-.142275x}} \tag{8}$$

Here, x is the difference between two athlete's BeastScores. Figure 10 shows this Equation 8 plotted on top of the scatterplot of Probability vs. BeastScore. Originally, the exponent in the formula was given by $-7.11764*10^{-6}-.14427x$, but the constant with order of 10^{-6} was negligible. Using this formula, one could easily generate the 100x100 table requested with a computer program, which would be too large to fit in the paper presented.

The final step in developing our mathematical model was to test its robustness by conducting a sensitivity analysis. The goal of this analysis was to see how our probabilities changed when the -.144275 parameter in our probability formula was adjusted by a small amount. Using Mathematica, we

altered the parameter by 1% and 5%, and compared probabilities for varying differences in BeastScore. What we found was that altering the parameter only changed the probabilities by less than 0.3%, depending on the value of x chosen. The change in probability was closer to 0.3% as x approached zero and closer to 0% as the magnitude of x increased. This demonstrates that our model is very robust and we can be confident in our solution.

Conclusion

By utilizing several programs and our resources at Occidental College, we were able to develop a mathematical rating system and probability model from our athletic data. This model is very robust, and is simplicity makes it easy to understand and repeat in another environment. Although our model is very robust, there possible future improvements that we believe could be made to our model. For example, it would be very beneficial to our model if we had data for athletes who would have a BeastScore that was less than 30 or 90 or higher, since our model was lacking data in those regions. In addition, with a little bit more data, we could create a logistic probability function for each "bucket" based on the difference in BeastScore. This would make our probability model less generalized since we would not be averaging the probabilities for each difference in BeastScore. Finally, we believe it would be worthwhile to see how and if the linear coefficients and power transforms would change when adding more data. If enough data was added to our system, then it might be necessary to run through our process again with the collective data and obtain new linear coefficients and power transforms, which may alter our probability model as well.

References

1. "Basic Analysis in S-Plus." *Data and Statistical Services*. Princeton University Library, 2007. Web. 26 Apr. 2015. <http%3A%2F%2Fdss.princeton.edu%2Fonline_help%2Fstats_packages%2Fsplus%2Fsplus_analysis.htm>.
2. “DSS - S-Plus Analysis.” Princeton Univeristy Library. Accessed April 28, 2015. http://dss.princeton.edu/online_help/stats_packages/splus/splus_analysis.htm.
3. Jones, Royal Clifford, Jr. "Ordinal Ratings." *Back to Basics in Rating Theory*. N.p., 1 Aug. 2011. Web. 26 Apr. 2015. <http://ratingtheory.com/ordinal.htm>.
4. Jones, Royal Clifford, Jr. "Percentage Expectancy." *Back to Basics in Rating Theory*. N.p., 1 Aug. 2011. Web. 26 Apr. 2015. <http://ratingtheory.com/expectancy.htm>.
5. Larget, Bret, Prof. "Regression in S-PLUS." University of Wisconsin, 2 May 1997. Web. 26 Apr. 2015. <http://www.stat.wisc.edu/~larget/math496/regression.html>.
6. Lengyel, Tamas. l9thp. Tibco Spotfire S+, n.d.
7. Lm Linear Regression Model (version 8.2). Tibco Spotfire S+. 212 Elm Street, Somerville, MA 02144: Tibco, n.d. http://spotfire.tibco.com.
8. "LogitModelFit." *Wolfram Language Documentation*. Wolfram Research Inc., 2008. Web. 26 Apr. 2015. <https://reference.wolfram.com/language/ref/LogitModelFit.html>.
9. MATLAB. Vers. 2014. Natick, Massachusetts: MathWorks, 2014. Computer software.
10. “P Values (Calculated Probability) and Hypothesis Testing - StatsDirect.” Statistical Help. Accessed April 28, 2015. http://www.statsdirect.com/help/default.htm#basics/p_values.htm.

11. Wolfram Research, Inc. Mathematica. Vers. 10.0. Champaign, Illinois: Wolfram Research, Inc., 2014. Computer software.

Appendix A: Figures

Figure 1.

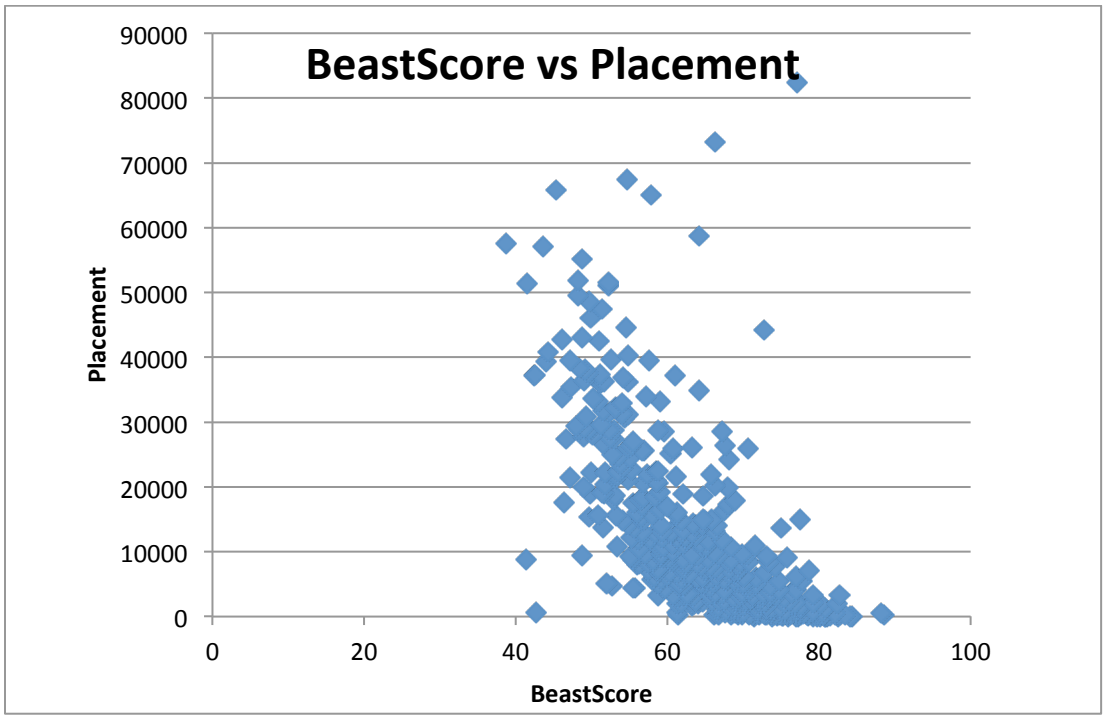


Figure 2.

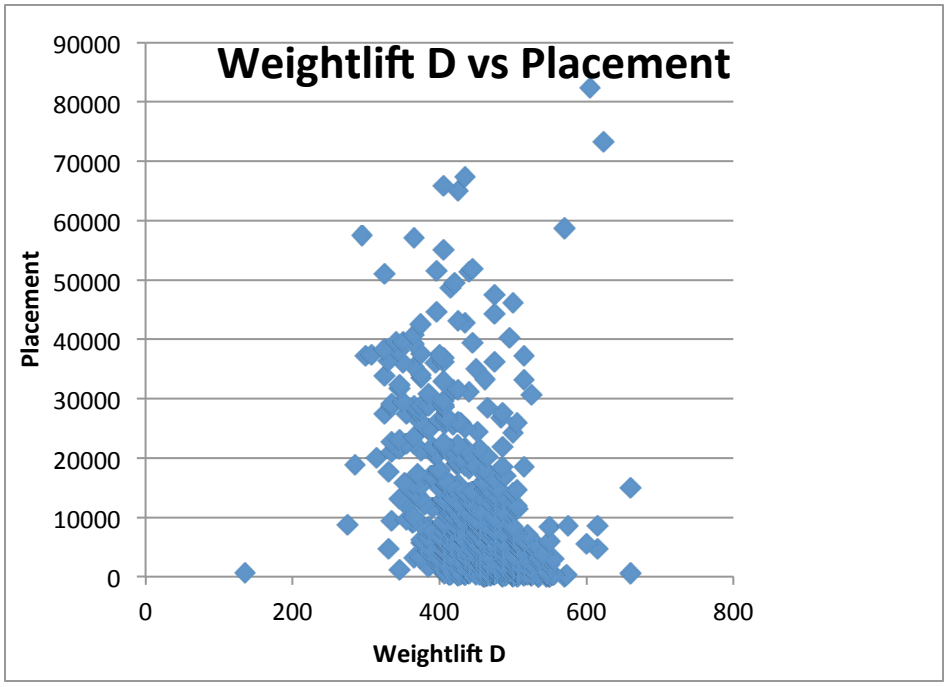


Figure 3.

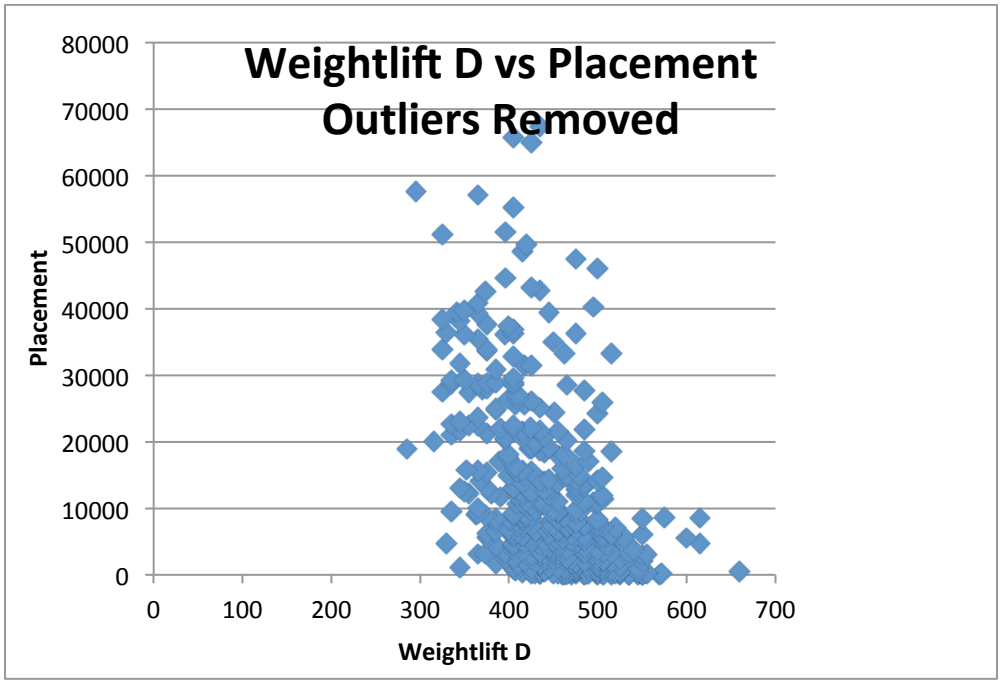


Figure 4.

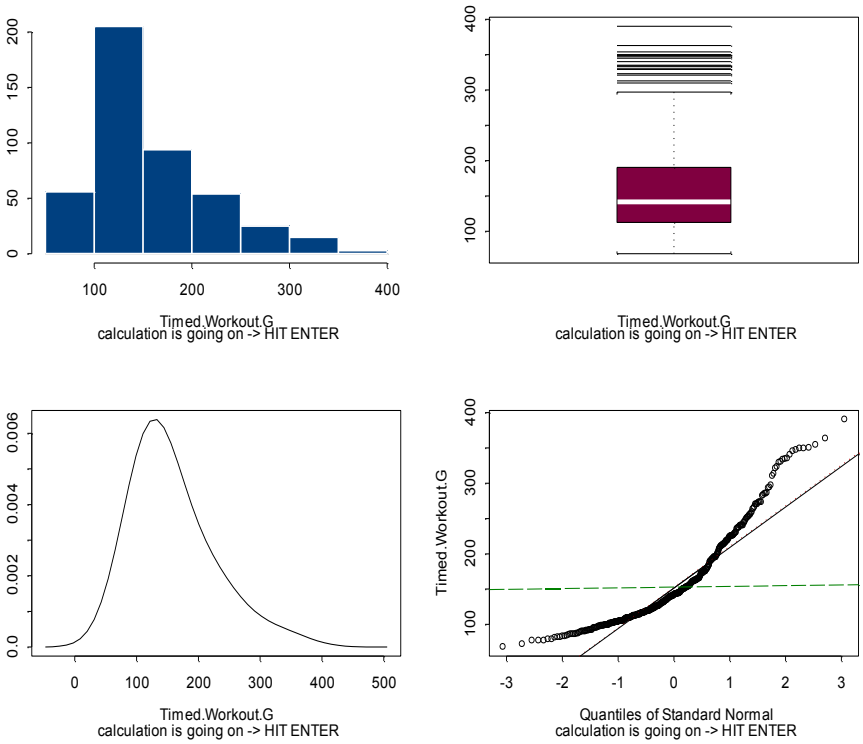


Figure 5.

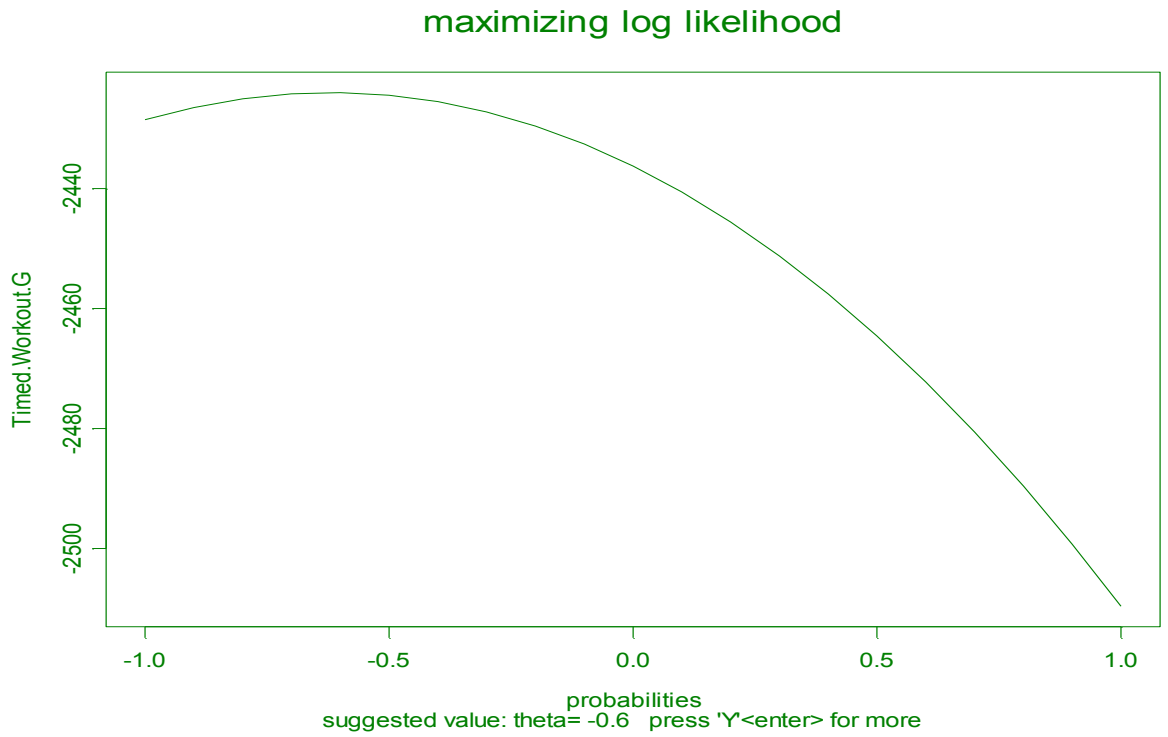


Figure 6.

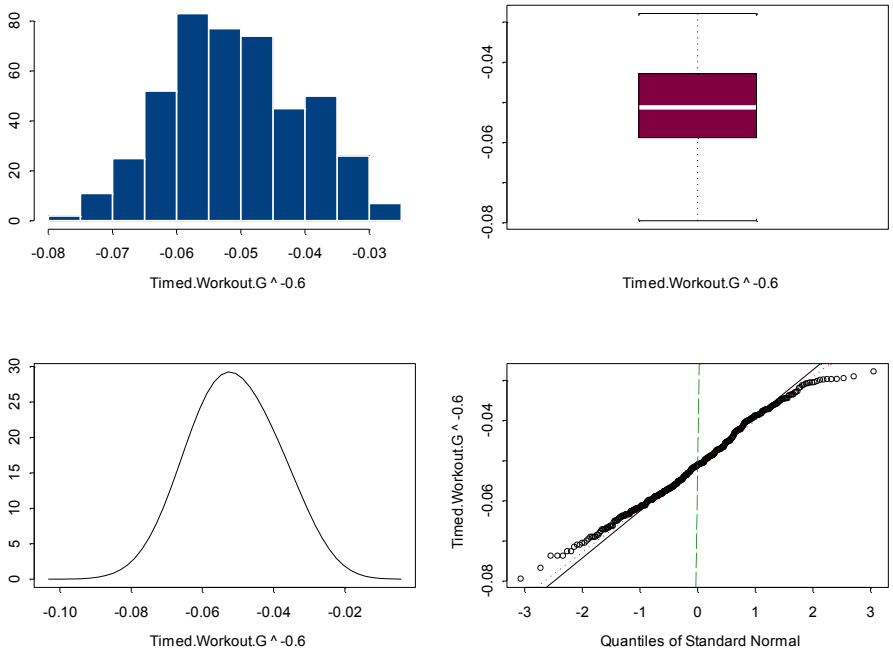


Figure 7.

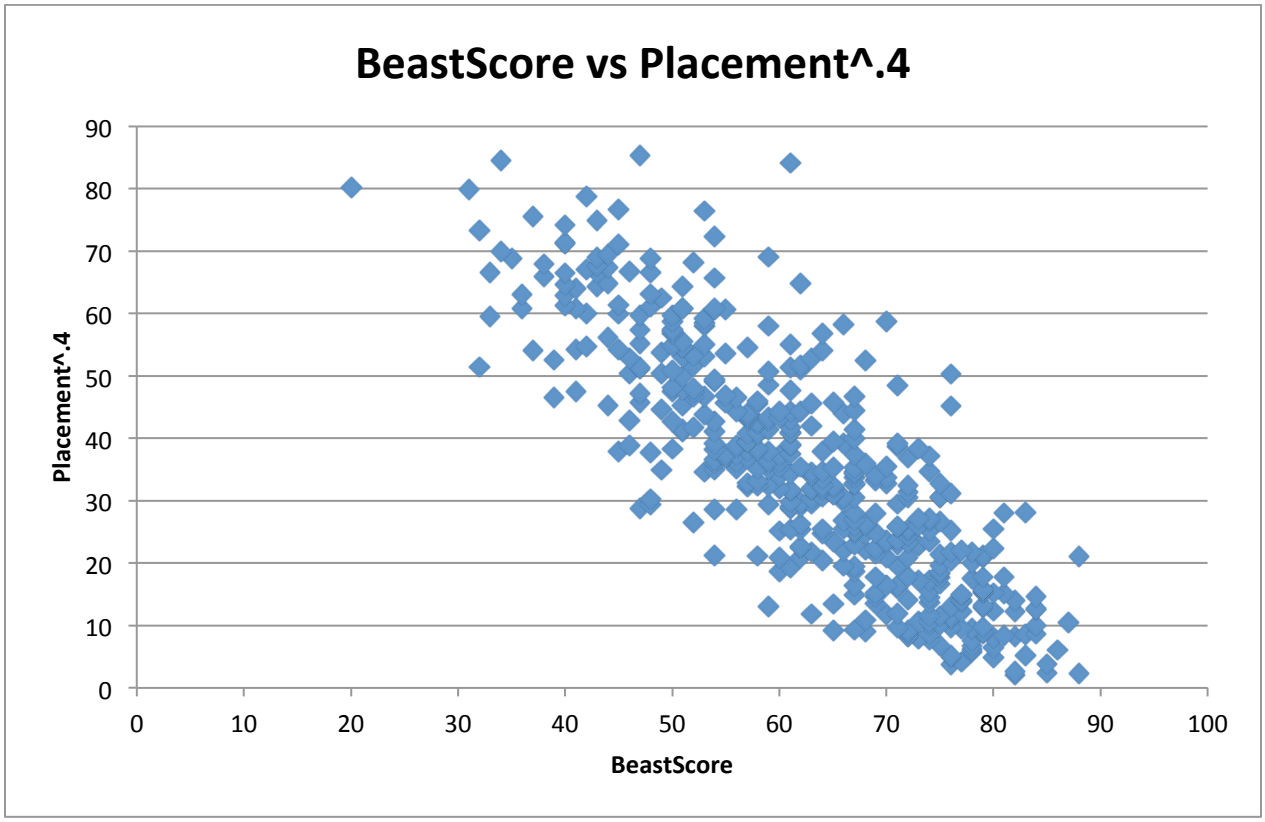


Figure 8.

```
>> DataTable2

T =
```

	W30_34	W35_39	W40_44	W45_49	W50_54	W55_59	W60_64	W65_69	W70_74	W75_79	W80_84	W85_89
30-34	0.5	0.69841	0.65143	0.81696	0.89091	0.97844	0.97186	0.99524	0.99762	1	1	1
35-39	0.30159	0.5	0.41333	0.69792	0.79192	0.95178	0.95118	0.99074	0.99259	0.99794	1	1
40-44	0.34857	0.58667	0.5	0.76875	0.85745	0.96679	0.96364	0.994	0.99533	0.99852	1	1
45-49	0.18304	0.30208	0.23125	0.5	0.57159	0.773	0.83097	0.91927	0.95729	0.98148	1	1
50-54	0.10909	0.20808	0.14255	0.42841	0.5	0.76226	0.82479	0.91727	0.95182	0.97811	0.99545	1
54-59	0.021563	0.048218	0.033208	0.227	0.23774	0.5	0.67181	0.82075	0.90283	0.95073	0.98978	0.99686
60-64	0.028139	0.048822	0.036364	0.16903	0.17521	0.32819	0.5	0.67121	0.79798	0.91611	0.96212	0.98485
65-69	0.0047619	0.0092593	0.006	0.080729	0.082727	0.17925	0.32879	0.5	0.65361	0.81204	0.88542	0.95278
70-74	0.002381	0.0074074	0.0046667	0.042708	0.048182	0.09717	0.20202	0.34639	0.5	0.68086	0.77431	0.9
75-79	0	0.0020576	0.0014815	0.018519	0.021886	0.049266	0.083895	0.18796	0.31914	0.5	0.6304	0.80556
80-84	0	0	0	0	0.0045455	0.01022	0.037879	0.11458	0.22569	0.3696	0.5	0.72917
85-89	0	0	0	0	0	0.0031447	0.015152	0.047222	0.1	0.19444	0.27083	0.5

```
fx >>
```

Figure 9.

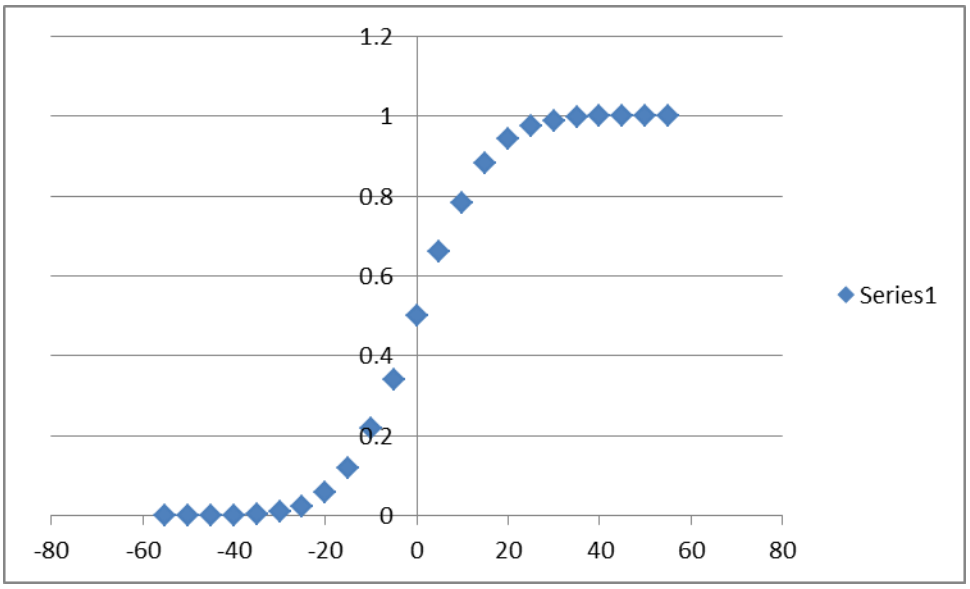
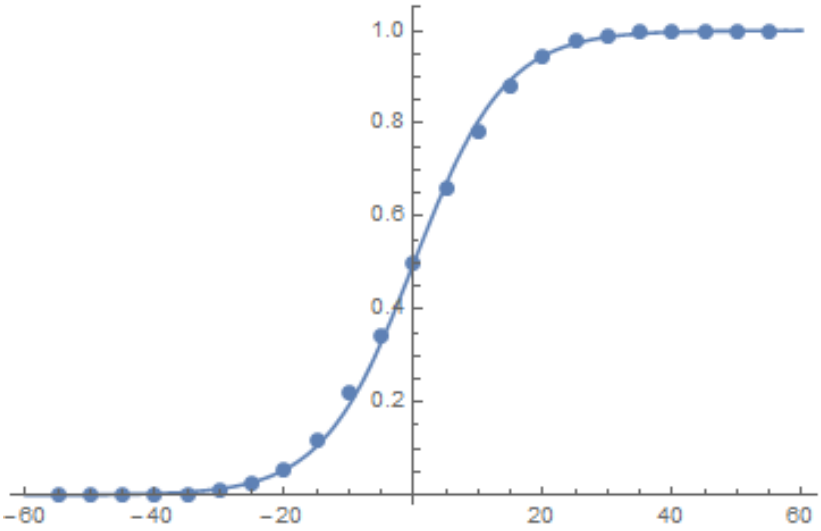


Figure 10.



Appendix B: S Plus Codes

Program 1:

```
function(x, mark = 10., from = -1., to = 1., ylab = deparse(substitute(x)))
{
        #  l9thp
        #  last modified: 120501W
        #  last modified on 042108M by tl
        ##################################################
        x <- x[!is.na(x)]
        frame()
        parold <- par()
        su <- summary(x)
        if(su[3.] - su[2.] > su[5.] - su[3.]) {
                #           from <- 1
                #           to <- 5
                {
                        l9prstars()
                        cat("I suspect that the distribution is SKEWED to the LEFT.\n")
                        cat("PLEASE CHECK BELOW IF THIS IS THE CASE INDEED.\n")
                        l9prstars()
                }
        }
        l9quickprint("THE MINIMUM VALUE IS ", su[1.], " -- if it is negative", "then you would better
watch out, e.g., use a shift, e.g.,", "    ...+min+... instead")
        out <- c()
        maxx <- -1000000000.
        sq <- seq(from, to, 1./mark)
        #######
        cat("... calculation is going on ... HIT ENTER!!!\n\n")
        l9.1(x, xlab = ylab, sub = "calculation is going on -> HIT ENTER")
        #######
        l9wait()
        #       y1 <- summary(out)[1]
        #       y2 <- summary(out)[6]
        for(i in sq) {
#                y1 <- l9thact
                  #           if(i == sq[length(sq)])
                  #                   y2 <- l9thact
                  l9thact <- l9th(x, i)
                  maxx <- max(maxx, l9thact, na.rm = T)
                  out <- c(out, l9thact)
        }
```

```
        cat("the list of values:\n")
        #, "\n\n")
        l9quickprint(out)
        l9prstars()
        ######################
        options(digits = 4.)
        where <- from + 1./mark * (time(out)[out == maxx] - 1.)
        cat("\n the MAXIMUM is =", maxx, " taken at ", format(where), "\n")
        #       l9wait()
        l9prstars()
        frame()
        par(mfrow = c(1., 1.))
        #       box(col = 1)    #2-purple)
        #       frame()
        plot(sq, out, type = "l", lty = 1., col = 4., xlim = c(from, to), ylab = ylab = ylab, xlab =
"probabilities", main = "maximizing log likelihood", sub = paste(
                "suggested value: theta=", format(where), "  press 'Y'<enter> for more"))
        cat("Would you like to check this power (N/Y)?")
        #   invisible()
        if(l9wait() == "Y") {
                cat("\n   wrapping up by checking qqnorm(y^theta) ... \n\n")
                xw <- l9power(x, where)
                #deparse(substitute(where))))
                l9.1(xw, xlab = paste(deparse(substitute(x)), "^", format(where)))
        }
        l9wait()
        par(parold)
```

Program 1:

```
function(formula, data, weights, subset, na.action, method = "qr", model = F, x = F, y = F, contrasts =
NULL, ...)
{
        call <- match.call()
        m <- match.call(expand = F)
        m$method <- m$model <- m$x <- m$y <- m$contrasts <- m$... <- NULL
        m$drop.unused.levels <- T
        m[[1]] <- as.name("model.frame")
        m <- eval(m, sys.parent())
        if(method == "model.frame")
                return(m)
        Terms <- attr(m, "terms")
        xvars <- as.character(attr(Terms, "variables"))
        if((yvar <- attr(Terms, "response")) > 0)
                xvars <- xvars[ - yvar]
```

```
        if(length(xvars) > 0) {
                xlevels <- lapply(m[xvars], levels)
                xlevels <- xlevels[!unlist(lapply(xlevels, is.null))]
                if(length(xlevels) == 0)
                        xlevels <- NULL
        }
        else xlevels <- NULL
        weights <- model.extract(m, weights)
        Y <- model.extract(m, response)
        X <- model.matrix(Terms, m, contrasts)
        na.message <- attr(m, "na.message")
        saved.na.action <- attr(m, "na.action")
        if(!model)
                unset(m)
        fit <- if(length(weights)) lm.wfit(if(x) X else unset(X), Y, weights, method, ...) else lm.fit(if(x) X
else unset(X), Y, method, ...)
        fit$terms <- Terms
        fit$call <- call
        if(model)
        fit$model <- m
        if(x)
                fit$x <- X
        if(y)
                fit$y <- Y
        if(!is.null(xlevels))
                attr(fit, "xlevels") <- xlevels
        attr(fit, "na.message") <- na.message
        fit$na.action <- saved.na.action
        fit
}
```

Appendix C: MATLAB Programs

Program 1:

```
function [ x ] = Model( A,p,q,r,o )
%Probability calculator for "buckets" of BeastScores
%   A is the input matrix of each individual's corresponding BeastScore and
%   Placement^(0.4). p and q are the bounds for the first bucket, r and o are
%   the bounds for the second bucket.

n=size(A,1); % n = total number of participants
w=0; % w = number of "wins" an individual in the upper bucket earned
t=0; % t = number of "games" played per person
b=0; % b = number of people in the second bucket
sumw=0;
sumt=0;

% calculate number of "wins" per person in second bucket
for i=1:n
   w=0;
   t=0;
   if ge(A(i,2),r) && A(i,2) < o
      b=b+1; %calculate # of people in second bucket
      for j=1:n
         if ge(A(j,2), p) && A(j,2) < q && i~=j
            t=t+1;
           if A(j,1)>A(i,1)
              w=w+1;
            end
         end
      end
      sumw=sumw+w; % total number of "wins"
      sumt=sumt+t; % total number of "games"
   end
end
x=sumw/sumt; % Probability of winning (the % of wins amongst all people in the second bucket vs.
people in first bucket)
end
```

Program 2:

```
% Probablity Table Generator
% Import our data, then create a 12*12 Table which gives us the
% probability that a person in the range in column A beats a person in the
```

```
% range in column B

num=xlsread('Placement_vs_BS.xlsx');
RowNames = {'30-34','35-39','40-44','45-49','50-54','54-59','60-64','65-69','70-74','75-79','80-84','85-89'};
ColNames = RowNames;
A=zeros(12,12);
for i=4:15
  for j = 4:15
    A(j-3,i-3)=Model(num,j*5+10,(j+1)*5+10,i*5+10,(i+1)*5+10);
  end
end
T=table(A(:,1),A(:,2),A(:,3),A(:,4),A(:,5),A(:,6),A(:,7),A(:,8),A(:,9),A(:,10),A(:,11),A(:,12),'RowNames',RowN
ames,'VariableNames',{'W30_34' 'W35_39' 'W40_44' 'W45_49' 'W50_54' 'W55_59' 'W60_64' 'W65_69'
'W70_74' 'W75_79' 'W80_84' 'W85_89'})
```

Program 3:

```
function [ x,y ] = AvgProb(A)
%Averages the probabilities for each difference in BeastScore
%   Algorithm goes along the main diagonal, and then each consecutive
%   sub-diagional to average the probability of winning if the difference
%   in BeastScores A-B = -55,-50,-45,...,-5,0,+5,...+45,+50,+55.  x gives
%   the probabilities for differences from 0 to -55, and y gives the
%   probabilities for differneces from 0 to +55.

n=size(A,1);
sum = 0;
m=0;

for i=1:n
  sum = 0;
  m = 0;
  for j=0:n-i
    sum = sum+A(j+1,i+j);
    m=m+1;
  end
  y(i)=sum/m;
end

for i=1:n
   sum = 0;
   m = 0;
   for j=0:n-i
      sum = sum+A(i+j,j+1);
      m=m+1;
   end
   x(i)=sum/m;
```

```
end
end
```