Errors in assessment of resident performance: R code

Supplemental Digital Content 1 for
"Errors in assessment of resident performance"
(RE: Baker K. et al. 2011; 115:862–875).

Johan van Schalkwyk

February 2012

Version 1.03

Contents

1 Overview

In a recent article,[1] anesthesiology residents in a large center were repeatedly assessed by multiple assessors, and the resulting Likert scores were then adjusted to compensate for assessor bias. The Z scores thus obtained were evaluated by comparing the mean score with zero, under the assumption that the individual concerned is 'reliably above average' if the mean score is more than 1.96 times the standard error of the mean above zero; a similar assessment was made on mean scores below zero.

This problematic approach is analyzed in the letter that accompanies this simulation. The following R program* simulates Baker's analysis, and demonstrates that his approach frequently misclassifies residents as above or below average, despite their underlying competence being unexceptional.

*See the R project for statistical computing, http://www.r-project.org/ Date last accessed, 4 February 2012.

2 R code

Under Baker's apparent assumption that anesthesiologists' overall competence is a continuous variable that can be normalized, I randomly select assessment scores for each resident from a normal distribution, and offset the mean for each resident by a second, randomly generated value that represents inter-individual variation in competence. Based on known individual competence and a chosen 'competence threshold', I then test Baker's analysis to determine how many residents are misclassified.

The following code initializes several variables that can be used to modify the shape of the plot that simulates Baker's Fig. 5; defines three ancillary routines; creates and plots an actual simulation of Baker's Fig. 5; and finally repeats this simulation to achieve reliable assessment of the number of residents who are falsely labelled.

2.1 Initialization & necessary subroutines

The following variables are important:

SQUEEZE The standard deviation used to determine the degree to which scores vary for each individual (intra-individual variation). Empirically, this is set at 0.5 but small variations (e.g. 0.6, 0.4) don't affect the outcomes substantially. Small values result in tiny values for the SEM 'error bars' that are quite unlike Baker's results; large values conversely result in huge error bars;

ADJUSTMENT This value is used as the standard deviation of the the *inter*-individual variation (SD_{adj}). Again, an empiric value of 0.5 gives realistic results, and small variations are acceptable. With larger values, the range of Z scores becomes far greater than those reported by Baker.

CONFIDENCE 1.96 standard deviations provide 95% confidence intervals. See usage.

THRESHOLD This value is used to determine how much an individual's mean score must vary before they are considered 'abnormal'. Convention would suggest that this should be 1.96 times the value in ADJUSTMENT, but we find that at this level, Baker's approach performs dismally, with most of the residents labelled as being 'under' or 'over' performing being falsely labelled, as they are inside the 95% confidence limits. I therefore set this value at just 1.0 SD_{adj}, but you can vary the value and assess the performance of Baker's approach.

I also define the convenience constant ‘ROW’ (see usage below).

```
SQUEEZE <- 0.5             # intra-individual variation. start with 0.5
ADJUSTMENT <- 0.5          # intER-individual variation (offset mean)
CONFIDENCE <- 1.96         # 1.96 gives 95% CIs
THRESHOLD <- 1.0 * ADJUSTMENT # anyone outside 1 SD is "good or bad" !
ROW <- 1                   # convenient constant (mnemonic)
```

2.1.1 Draw an error bar

This routine is straightforward.[†] It simply draws an error bar. Arguments accepted are:

x,y coordinates of mean

upper length of upper error bar

lower lower bar, defaults to be identical to upper length

flag a zero value results in a black graphic; under zero (-1) green, and over zero (1) red.

len length of the top/bottom horizontal line.

Note that a flag value of under zero signals that the individual whose data are summarized by the bar has a mean that lies outside the THRESHOLD value specified above in Section 2.1, but has been misclassified as ‘not an outlier’. Conversely, a value over zero says that the resident involved has been falsely labelled as being above or below the THRESHOLD.

```
Ebar <- function(x, y, upper, lower=upper, flag, len=0.1,...)
  { if(length(x) != length(y)
     | length(y) !=length(lower)
     | length(lower) != length(upper))
      stop("vectors must be same length")
    clr = 'black';
    if (flag < 0) { clr = 'green' }  # missed
    if (flag > 0) { clr = 'red' }    # false call as above/below normal
    arrows(x,y+upper, x, y-lower, angle=90,
           code=3, col=clr, length=len, ...)
    points(x,y, col=clr, pch=16)     # draw
  }
```

All the routine does is draw the error bar using the R function *arrows {graphics}*; *points {graphics}* draws the midpoint as a circle, using plotting character code = 16. There is no return value.

[†]Based on http://monkeysuncle.stanford.edu/?p=485 — code by James Holland Jones. Last accessed 4 February, 2012.

2.1.2 Has a resident been falsely labelled?

The following routine determines the 'false labelling' described in the preceding section. It accepts a cutoff value (defaulting to the previously described value in THRESHOLD), the CONFIDENCE (similarly defined above) and a vector **r** with the following components that describe properties of a particular resident:

r[1] the mean of the resident's scores;

r[2] the standard error of that mean;

r[3] the deviation of the mean from zero. A positive deviation indicates some measure of 'superior performance' and I have labelled this 'goodness'; conversely a negative value might be considered 'badness'. This value was generated using the ADJUSTMENT value described in Section 2.1.

After obtaining these values from the vector (the names used are self-explanatory), I determine the following:

1. If the resident's mean Z score is distant from zero by 1.96*SEM for that individual, Baker will classify this resident as either 'above average' or 'below average'. (Rather than hard-coding the value of 1.96, I've used CONFIDENCE, as defined above). If Baker's approach classifies the resident as 'above average' but the actual 'goodness' that we have allocated is sub-threshold (e.g. below 1 SD) then we have a false positive, and return 1; similarly, if Baker's approach says 'below average' but the allocated 'goodness' (here the term 'badness' might be more appropriate) is above the lower threshold, I similarly return 1.

2. Conversely, if Baker's criteria are not met but the individual is actually outside the cutoff limit, I return -1;

3. The default is zero (No discrepancy).

```
Isduff <- function (r, cut=THRESHOLD, conf=CONFIDENCE)
 { m <- r[1]
   sem <- r[2]
   goodness <- r[3]
   if (  ((m-conf*sem > 0) && (goodness < cut))
       ||((m+conf*sem < 0) && (goodness > -(cut)))
      ) { return (1) }              # false positive
   if (  ((m > 0) && (m-conf*sem <= 0) && (goodness >= cut ))
       ||((m < 0) && (m+conf*sem >= 0) && (goodness <= -(cut) ))
      ) { return (-1) }             # false negative
   return (0)
 }
```

2.1.3 Calculate standard error of mean

The following trivial routine determines the standard error of the mean of data **d**, calculated as the standard deviation divided by the square root of the number of items in **d**. The standard error is returned.

```
Sem <- function (d) { sd(d)/sqrt(length(d)) }
```

2.2 Main program

Here I simulate the analytic process apparently followed by Baker. I provide twenty samples for each of 100 residents, sampling from a normal distribution using the R function *rnorm{stats}*. The standard deviation is SQUEEZE as described previously.

Note that in Baker's actual study, he analyses data for just over 100 residents, and likely used more data for most residents assessed. This fact is not material to the simulation, as the value in SAMPLES can simply be adjusted upwards; note however that a small compensatory adjustment to SQUEEZE will then be required.[‡] I create the vectors **means** and **sems** to store the means and SEMs of each of the 100 residents.

```
RESIDENTS <- 100
SAMPLES <- 20  # 20 samples per resident. Can vary.
scores <- array( rnorm(n=SAMPLES*RESIDENTS, mean=0, sd=SQUEEZE),
                 dim=c(RESIDENTS,SAMPLES))
means <- apply (scores, MARGIN=ROW, mean) # or use rowMeans
sems  <- apply (scores, MARGIN=ROW, Sem)  # find standard errors
```

I then create the **goodness** vector (See also Section 2.1.2) that simulates inter-individual variation. This value determines for each resident whether they are 'above average' (positive value for goodness) or 'below average', the ADJUSTMENT being used as the standard deviation when these values are sampled from a normal distribution. The value in THRESHOLD will be used to determine whether an individual is 'actually' above or below an arbitrary threshold, initially conservatively set at just 1 SD.

In the following code, the array *all* is used to conveniently store:

1. The individual's mean, (which is now offset by the 'goodness');
2. The individual's standard error of the the mean;

[‡]It should also be noted that the variation in number of assessments from resident to resident introduces a further error in Baker's assessment, but I will not here explore this error, other than noting that appropriate use of Analysis of Means would correct for some of his erroneous assumptions.

3. The actual value of the ‘goodness’ for later comparison, see Section 2.1.2;
4. A flag derived using the Isduff function that is one of {-1,0,1}, respectively indicating a false negative label, a correct label, or a false positive label.

```
goodness <- rnorm(100, sd=ADJUSTMENT)
all <- array(NA, dim=c(100,4))  # array of all data
all[,1] <- means+goodness       # adjust mean for 'goodness'
all[,2] <- sems                 # store SEM
all[,3] <- goodness             # retain 'goodness' (-ve is 'badness')
colnames(all) <- c("mean","sem","goodness", "duff") # name columns
all[,4] <- apply(all, MARGIN=ROW, Isduff)
```

As Baker did, I next sort all individuals based on their mean Z score and count the number of individuals who are ‘reliably above / below average’, based on whether their mean is distant from zero by over 1.96 times the standard error of the mean (the value in CONFIDENCE); and finally, false positives and negatives.

```
sorted <- all[order(all[,1], decreasing=TRUE),]
# count the number where lower limit (95% CI) above zero:
goodnesscount <- length(sorted[ sorted[,1]-CONFIDENCE*sorted[,2] > 0, 1])
# similarly for upper CI below zero:
badcount <- length (sorted[ sorted[,1]+CONFIDENCE*sorted[,2] < 0, 1])
Fpos = length ( all[ all[,4] == 1,1] ) # false positives
Fneg = length ( all[ all[,4] == -1,1] )
```

These results can then be displayed, using a rather inefficient *for* loop to draw the error bars for each student, relative to a baseline of zero. The legend details those whom Baker would classify as ‘above’ or ‘below’, and the number of false positives and false negatives encountered with this approach, for a single simulation of 100 residents.

```
plot(x=0,y=0, xlim=c(0,100), ylim=c(-1.5,1.5),
     main = paste("Cutoff threshold",
     round(THRESHOLD/ADJUSTMENT, digits=2), "SD"),
     xlab="Rank Order", ylab="Mean Z score +- SEM")  # plot new figure
for (stud in 1:100)
   { Ebar(x=stud, y=sorted[stud,1],
     upper=CONFIDENCE*sorted[stud,2],
     flag=sorted[stud,4] )
   }
   segments (0,0,100,0)   # draw line through y=0
   legend (0, -1, paste ("Above = ", goodnesscount,
        "Below = ", badcount, "F+ve = ", Fpos, "F-ve = ", Fneg ))
```

2.3 Repeated testing

It should be clear that the numbers obtained for a particular simulation will vary somewhat. Results of repeated testing can be used to average out these values and determine the overall error rate (etc). The following simple code does just this.

2.3.1 Simulate an entire data set

The above process is repeated in a simplified manner within the function *Once*. This function accepts the following arguments:

sqz a SQUEEZE value, as above;

adj similarly, an ADJUSTMENT value;

samples the number of samples per resident;

residents the number of residents;

confidence the CONFIDENCE value, conventionally 1.96.[§]

In the following, in an analogous manner to the main program of Section 2.2, I:

1. create an array of residents' scores, selecting from a normal distribution with SD 'sqz';
2. determine the means of these scores;
3. determine the standard error;
4. generate the 'goodness' g, which reflects how the mean of each individual varies from zero, with a positive value considered 'above average', and a negative value 'below';
5. store the vectors in the 'all' array here termed 'A', as described previously;
6. determine the numbers 'above average' (goodc), 'below average' (badc), and false positives (fp) and negatives (fn). These last four values are returned as a vector.

```
Once <- function (sqz, adj, samples, residents, confidence)
 { # [an enhancement might be to validate the arguments here]
   s <- array( rnorm(n=samples*residents, mean=0, sd=sqz),
               dim=c(residents,samples))
   m <- rowMeans (s)
```

[§]Note however that the function Isduff defaults to the global value of CONFIDENCE, which is a bit ugly.

```
   se  <- apply (s, MARGIN=ROW, Sem)
   g <- rnorm(100, sd=adj)

   A <- array(NA, dim=c(100,4))
   A[,1] <- m+g
   A[,2] <- se
   A[,3] <- g
   A[,4] <- apply(A, MARGIN=ROW, Isduff)

   goodc <- length(A[ A[,1] - confidence*A[,2] > 0, 1])
   badc  <- length(A[ A[,1] + confidence*A[,2] < 0, 1])
   fp    <- length(A[ A[,4] == 1,1] )
   fn    <- length(A[ A[,4] == -1,1] )

   return (c(goodc, badc, fp, fn))
 }
```

2.3.2 Repeat simulation 10,000 times

I then simply invoke the function *Once* ten thousand times, storing the results in the array 'goes'. The first column of *goes* is thus the number classified as 'above average', the second those 'below', and the third and fourth the false positives and negatives.

```
repeats <- 10000
goes <- array (NA, dim=c(repeats,4))
for (n in 1:repeats) { goes[n,] <- Once(SQUEEZE,
                                        ADJUSTMENT,
                                        SAMPLES,
                                        RESIDENTS,
                                        CONFIDENCE)
                      }
print (c ( mean (goes[,1]),
           mean (goes[,2]),
           mean (goes[,3]),
           mean (goes[,4])
       )  )
```

The last line simply prints the mean values obtained.

3 References

1. Baker K. Determining Resident Clinical Performance: Getting Beyond the Noise. Anesthesiology 2011; 115:862–875.

4 Availability & Testing

This code is made available at the following URL:

http://www.anaesthetist.com/R/Baker2011/

This document is also available as R code, as a LyX document, and as a PDF LaTeX document.

4.1 Testing

This code has been tested under R version 2.10.1 running under Ubuntu Linux version 10.04. Thanks to Associate Professor Ross Ihaka (Ph.D, Department of Statistics, The University of Auckland, Auckland, New Zealand) for evaluation of and comments on the code. Note that there are several ways that the above crude code can be made more efficient.

4.2 Change log.

1. Initial version was 1.0;
2. In version 1.01, minor alterations were made to the documentation, and 'CUTOFF' was renamed 'THRESHOLD' as this may be slightly more mnemonic. The code was otherwise not altered apart from minor re-formatting.
3. In version 1.02, I made more explicit reference to SD_{adj}.
4. In version 1.03, I made amendments to the text in keeping with the requirements of the journal *Anesthesiology*, including the use of US spelling throughout, non-numeric footnotes, and a reference list. No alterations have been made to the R code.