Standardization, percentiles, and the normal curve

Andrew Johnson

Introduction

As you can imagine, it is frequently useful to convert **raw scores** to **z scores** or **T scores**. For example, if you wanted to compare two measures that have different scales (e.g., one is scored out of 40, and the other is scored out of 60), you would want to convert both scores to a common metric before conducting any comparisons.

In this demonstration, we will work through the following tasks in R:

1) Generating random numbers
2) Standardizing vectors of numbers
3) Converting z-scores to T-scores
4) Finding percentiles for z-scores

We can do all of these tasks using just the default R libraries, but I like the `describe` function in the `psych` package, and so I will load this library before we get started.

```r
library(psych)
```

Generating random numbers

The most common use for vectors of random numbers is (in my experience at least) for use within statistics demonstrations. . . but if you ever have occasion to use "computer-intensive" statistics such as *bootstrapping*, *jacknifing*, or *Monte Carlo simulation*, then you will need to become familiar with random number generation in R.

For now, let's just generate random numbers for the purposes of illustrating standardization functions in R. Imagine that you have a group of 100 students who completed two different tests of cognitive function - one of which yielded a score out of 40, and the other yielding a score out of 60. Scores on each of the tests were equal to the "total number correct", and so both variables are solely comprised of integer values.

To create our mock dataset, we need to: 1) Create an ID variable for the 100 students (i.e., a vector of numbers between 1 and 100). 2) Create a vector of plausible values for each of the tests. Let's assume that students received scores between 40% and 100% on each test, which gives us a range of 16 to 40 for the first test, and a range of 24 to 60 for the second test.

We will store all of these values in a data frame called `testdata`:

```r
testdata <- data.frame(ID = 1:100,
                       test1 = sample(16:40, 100, replace = TRUE),
                       test2 = sample(24:60, 100, replace = TRUE))
```

If we had needed real numbers, rather than integer values, we could have sampled from the uniform distribution using the following:

`runif(100, 16, 40)` and `runif(100, 24, 60)`

or we could have sampled our 100 participants from the z-distribution:

`rnorm(100, 0, 1)`

or from the T distribution:

`rnorm(100, 50, 10)`

Comparing Test #1 with Test #2

Let's take a look at the descriptives for our dataset:

```
describe(testdata[,2:3], skew=FALSE)
```

```
##       vars   n  mean    sd median trimmed   mad min max range   se
## test1    1 100 26.94  7.00   27.0   26.69  8.90  16  40    24 0.70
## test2    2 100 42.87 10.94   43.5   43.04 14.83  24  60    36 1.09
```

Looking at these descriptives, the scores are obviously higher on `test2` than they are on `test1`, but…this is probably due to the fact that the scale is different between the two measures. If we want to be sure that the variables are comparable, we need to convert them to a common scale.

One obvious method for making the tests comparable is to convert the raw scores to percentages. We can easily do this by dividing each value by the total score for the test (40 and 60 for `test1` and `test2`, respectively) and then multiplying this ratio by 100.

```
testdata$test1.percent <- (testdata$test1 / 40) * 100
testdata$test2.percent <- (testdata$test2 / 60) * 100
describe(testdata[,2:5], skew=FALSE)
```

```
##               vars   n  mean    sd median trimmed   mad min max range   se
## test1            1 100 26.94  7.00   27.0   26.69  8.90  16  40    24 0.70
## test2            2 100 42.87 10.94   43.5   43.04 14.83  24  60    36 1.09
## test1.percent    3 100 67.35 17.49   67.5   66.72 22.24  40 100    60 1.75
## test2.percent    4 100 71.45 18.24   72.5   71.73 24.71  40 100    60 1.82
```

You will note from our descriptives that our summary statistics have changed in a predictable way, with the mean, median, standard deviation, minimum, and maximum all changing by a constant (i.e., a factor of 0.40 or 0.60, for `test1` and `test2`, respectively).

Converting Raw Data to z-Scores

Converting the scores to percentages provides us with a common scale (all of the variables have a maximum score of 100%). Another possibility that we might want to explore, however, is converting our variables to z-scores. This provides us with additional information - i.e., relative standing within the group. We can standardize our variables to z-scores, using the `scale` function:

```
testdata$test1.z <- scale(testdata$test1)
testdata$test2.z <- scale(testdata$test2)
describe(testdata[,2:7], skew=FALSE)
```

```
##               vars   n  mean    sd median trimmed   mad   min    max range   se
## test1            1 100 26.94  7.00  27.00   26.69  8.90 16.00  40.00 24.00 0.70
## test2            2 100 42.87 10.94  43.50   43.04 14.83 24.00  60.00 36.00 1.09
## test1.percent    3 100 67.35 17.49  67.50   66.72 22.24 40.00 100.00 60.00 1.75
## test2.percent    4 100 71.45 18.24  72.50   71.73 24.71 40.00 100.00 60.00 1.82
## test1.z          5 100  0.00  1.00   0.01   -0.04  1.27 -1.56   1.87  3.43 0.10
## test2.z          6 100  0.00  1.00   0.06    0.02  1.35 -1.72   1.57  3.29 0.10
```

The variable is now clearly a z-score, as it has a mean of 0 and a standard deviation of 1. Note that the `scale` function defaults to creating a z-score - you can choose to just *center* your data (i.e., set the mean to zero), or *scale* your data (i.e., divide all of the values by a constant, by the standard deviation, or by the root mean square of the variable). You can read more about centering and scaling by typing `?scale`.

Converting Raw Data to Other Standard Distributions

If you prefer, you can express your data as a T-score. The T-score distribution is really just a variant on the z-score distribution, with a mean of 50 and a standard deviation of 10, rather than the mean of 0 and standard deviation of 1 that is seen in the z-score distribution.

```
testdata$test1.T <- 10*scale(testdata$test1) + 50
testdata$test2.T <- 10*scale(testdata$test2) + 50
describe(testdata[,2:9], skew=FALSE)
```

```
##               vars   n  mean    sd median trimmed   mad   min    max range   se
## test1            1 100 26.94  7.00  27.00   26.69  8.90 16.00  40.00 24.00 0.70
## test2            2 100 42.87 10.94  43.50   43.04 14.83 24.00  60.00 36.00 1.09
## test1.percent    3 100 67.35 17.49  67.50   66.72 22.24 40.00 100.00 60.00 1.75
## test2.percent    4 100 71.45 18.24  72.50   71.73 24.71 40.00 100.00 60.00 1.82
## test1.z          5 100  0.00  1.00   0.01   -0.04  1.27 -1.56   1.87  3.43 0.10
## test2.z          6 100  0.00  1.00   0.06    0.02  1.35 -1.72   1.57  3.29 0.10
## test1.T          7 100 50.00 10.00  50.09   49.64 12.72 34.36  68.67 34.30 1.00
## test2.T          8 100 50.00 10.00  50.58   50.15 13.55 32.76  65.65 32.90 1.00
```

Or, given that `test1` and `test2` are cognitive ability tests, maybe you want to convert your scores to the standard IQ distribution (i.e., a mean of 100 and a standard deviation of 15).

```
testdata$test1.IQ <- 15*scale(testdata$test1) + 100
testdata$test2.IQ <- 15*scale(testdata$test2) + 100
describe(testdata[,2:11], skew=FALSE)
```

```
##                vars   n   mean    sd median trimmed   mad   min    max range   se
## test1             1 100  26.94  7.00  27.00   26.69  8.90 16.00  40.00 24.00 0.70
## test2             2 100  42.87 10.94  43.50   43.04 14.83 24.00  60.00 36.00 1.09
## test1.percent     3 100  67.35 17.49  67.50   66.72 22.24 40.00 100.00 60.00 1.75
## test2.percent     4 100  71.45 18.24  72.50   71.73 24.71 40.00 100.00 60.00 1.82
## test1.z           5 100   0.00  1.00   0.01   -0.04  1.27 -1.56   1.87  3.43 0.10
## test2.z           6 100   0.00  1.00   0.06    0.02  1.35 -1.72   1.57  3.29 0.10
## test1.T           7 100  50.00 10.00  50.09   49.64 12.72 34.36  68.67 34.30 1.00
## test2.T           8 100  50.00 10.00  50.58   50.15 13.55 32.76  65.65 32.90 1.00
## test1.IQ          9 100 100.00 15.00 100.13   99.46 19.07 76.54 128.00 51.46 1.50
## test2.IQ         10 100 100.00 15.00 100.86  100.23 20.32 74.13 123.48 49.35 1.50
```

Calculating Percentiles

Finally, another useful conversion that you might want to make is the conversion of z-scores into percentiles. R has a built-in function for this as well: `pnorm`. This function returns a value equal to the *probability that scores that would fall below this point in the standard normal distribution*. Thus, predictably, a z-score of 1.645 would yield a probability of 0.95:

```
pnorm(1.645)
```

```
## [1] 0.9500151
```

and a z-score of -1.645 would (equally predictably) yield a probability of 0.05:

```
pnorm(-1.645)
```

```
## [1] 0.04998491
```

We can add percentiles to our data frame as follows:

```
testdata$test1.pcntile <- pnorm(testdata$test1.z) * 100
testdata$test2.pcntile <- pnorm(testdata$test2.z) * 100
describe(testdata[,2:13], skew=FALSE)
```

```
##                vars   n   mean    sd median trimmed   mad   min    max range   se
## test1             1 100  26.94  7.00  27.00   26.69  8.90 16.00  40.00 24.00 0.70
## test2             2 100  42.87 10.94  43.50   43.04 14.83 24.00  60.00 36.00 1.09
## test1.percent     3 100  67.35 17.49  67.50   66.72 22.24 40.00 100.00 60.00 1.75
## test2.percent     4 100  71.45 18.24  72.50   71.73 24.71 40.00 100.00 60.00 1.82
## test1.z           5 100   0.00  1.00   0.01   -0.04  1.27 -1.56   1.87  3.43 0.10
## test2.z           6 100   0.00  1.00   0.06    0.02  1.35 -1.72   1.57  3.29 0.10
## test1.T           7 100  50.00 10.00  50.09   49.64 12.72 34.36  68.67 34.30 1.00
## test2.T          8 100  50.00 10.00  50.58   50.15 13.55 32.76  65.65 32.90 1.00
## test1.IQ         9 100 100.00 15.00 100.13   99.46 19.07 76.54 128.00 51.46 1.50
## test2.IQ        10 100 100.00 15.00 100.86  100.23 20.32 74.13 123.48 49.35 1.50
## test1.pcntile   11 100  49.20 30.94  50.34   48.59 45.29  5.89  96.90 91.01 3.09
## test2.pcntile   12 100  50.34 31.39  52.29   50.63 47.19  4.23  94.13 89.89 3.14
```

Summary

We now have five different ways of comparing scores on these two variables:

1) percentages
2) z-scores
3) T-scores
4) IQ scores
5) percentiles

As you can see from this demonstration, comparing apples to oranges is entirely feasible…or rather, it's simply a matter of standardization.