Disattenuating Correlations for Unreliability

Andrew Johnson

Load Libraries

We will need the `psych` library for this lab.

```r
library(psych)
```

The Data

We will use the `bfi` dataset from the `psych` package for this demonstration. This dataset is made up of 25 self-report personality items from the International Personality Item Pool, gender, education level and age for 2800 subjects and used in the Synthetic Aperture Personality Assessment.

The personality items are split into 5 categories: Agreeableness (A), Conscientiousness (C), Extraversion (E), Neuroticism (N), Openness (O). Each item was answered on a six point scale: 1 Very Inaccurate, 2 Moderately Inaccurate, 3 Slightly Inaccurate, 4 Slightly Accurate, 5 Moderately Accurate, 6 Very Accurate.

```r
data("bfi")
```

The `bfi` data frame contains 3 additional variables besides the personality items (`gender`, `education`, and `age`). Let's create a data object that just has the personality items in it, to facilitate some of our later analyses. We'll call it `bfi.items`.

```r
bfi.items <- bfi[,1:25]
```

Disattenuation of a Correlation Coefficient Due to Unreliability

When we calculate a correlation between two observed variables, we are really more interested in the correlation between the latent variables that they assess. Thus, any errors in measurement (e.g., unreliability) will impact on our ability to get a true sense of the relationship between these two variables. Specifically, the correlation between two latent variables (or constructs) will be underestimated when either (or both) of the observed variables being correlated, are less than perfectly reliable.

What this means is that we will need to determine the reliabilities for each of the variables - and to do this, we need to create scoring keys for the data.

Scoring The Questionnaire Data

We'll set up the scoring key by scale. There are five scales in our data:

- *Agreeableness*
- *Conscientiousness*
- *Extraversion*
- *Neuroticism*
- *Openness*

The first step is to set up a list object that indicates the items that go with each of our scales. We can indicate negatively-keyed items with a negative sign.

```
bfi.keys.list <- list(agree=c("-A1", "A2", "A3", "A4", "A5"),
                      consc=c("C1", "C2", "C3", "-C4", "-C5"),
                      extra=c("-E1", "-E2", "E3", "E4", "E5"),
                      neuro=c("N1", "N2", "N3", "N4", "N5"),
                      open=c("O1", "-O2", "O3", "O4", "-O5"))
```

We will now take this list object, and use the `make.keys` function to create the scoring key itself.

```
bfi.keys <- make.keys(bfi.items,bfi.keys.list,item.labels=colnames(bfi.items))
```

Finally, we can use this scoring key to calculate the scale scores for each of our 5 scales.

This is also where decisions about missing data are made - I have chosen to just average the non-missing values. If we had wanted to impute missing values with mean substitution (a problematic, albeit common choice), we would use `impute = "mean"` in place of `impute = "none"`. Finally, another common (and also problematic) practice is to use listwise deletion on the data, by only analyzing complete cases. To do this, you would remove `impute = "none"` and replace it with `missing = FALSE`.

Because we have both positively and negatively keyed items within our scales, we need to tell the function the minimum and maximum on the scale. This allows it to reverse key the negatively keyed items before creating the scale scores (an item can be reversed by subtracting the observed value from the maximum scale score - in this case, "6").

```
bfi.scored <-scoreItems(bfi.keys, bfi.items, impute = "none",
                         min=1, max=6, digits=3)
```

The actual scale scores for each of our five personality variables are now available within the `scores` value of the `bfi.scored` object.

```
head(bfi.scored$scores)
```

```
##      agree consc extra neuro open
## [1,]   4.0   2.8   3.8   2.8  3.0
## [2,]   4.2   4.0   5.0   3.8  4.0
## [3,]   3.8   4.0   4.2   3.6  4.8
## [4,]   4.6   3.0   3.6   2.8  3.2
## [5,]   4.0   4.4   4.8   3.2  3.6
## [6,]   4.6   5.6   5.6   3.0  5.0
```

Reliability of the Five Scale Scores

The `scoreItems` function allows us to quickly and easily calculate Cronbach's alpha for each scale.

```
bfi.scored$alpha
```

```
##           agree    consc     extra     neuro      open
## alpha 0.7030184 0.726735 0.7617328 0.8139629 0.6001725
```

Generating a Disattenuated Correlation Matrix

The `scoreItems` function also allows us to leverage the reliabilities calculated within the function to disattenuate the correlations among all possible combinations of the variables. The `corrected` value within the `bfi.scored` object that we generated using the `scoreItems` function presents:

- the correlations of all scales (**below the diagonal**)
- Cronbach's alpha for each scale (**on the diagonal**)
- the disattenuated correlations for each correlaton (**above the diagonal**)

```
bfi.scored$corrected
```

```
##            agree      consc      extra       neuro       open
## agree  0.7030184  0.3612032  0.6324390 -0.24511039  0.2290928
## consc  0.2581802  0.7267350  0.3527045 -0.30484444  0.2956128
## extra  0.4628106  0.2624221  0.7617328 -0.28389768  0.3195135
## neuro -0.1854161 -0.2344599 -0.2235453  0.81396295 -0.1233877
## open   0.1488103  0.1952313  0.2160373 -0.08624068  0.6001725
```

Final Thoughts

Although it is tempting to view these disattenuated correlation coefficients as being the "best" estimate of the correlation between two constructs, it is important to remember three things:

1) Disattenuation of a correlation does **absolutely nothing** to the measurement quality of a questionnaire or measure.
2) Disattenuation of a correlation to remove measurement error does **absolutely nothing** about measures that are measuring a construct inaccurately (i.e., it does not correct for low **validity**).
3) Disattenuation of a correlation to remove measurement error using the methods presented in this example provides an estimate of what the correlation between two constructs would be if there was zero measurement error. In practice, this is entirely impossible (i.e., every measure contains *some* error).