r2Winsteps: An R package for interfacing between R and the Rasch Modeling Software Winsteps

Daniel Anderson

2016-03-23

The `r2Winsteps` package was developed to provide a convenient interface between *R* and the Rasch modeling software *Winsteps*. The package is not intended to encompass the full capabilities of *Winsteps*, but rather to provide a simple framework for estimating many commonly applied models. The primary features of the package include:

- Write control and data files for Winsteps with the `r2Winsteps()` function, which includes automatic detection of partial credit scoring. For partial credit scoring, either the rating scale (default; Andrich 1978) or partial credit models (Masters 1982) can be estimated.
- Run Winsteps directly from R with the `runWinsteps()` function, which writes and executes a `.bat` file to call *Winsteps*. Both item and person parameters are returned in a list, and intermediary files (control and data) can be stored or discarded (default).
- Batch run a set of models with the `batchRunWinsteps()` files. Essentially does the same thing as `runWinsteps()`, but takes as its argument a list of data frames, with a different model fit and parameter estimates returned for each data frame in the list.

Note that the package is still in development, and plotting features are next on the docket for inclusion. For now, any plotting you'd like to do will either need to be completed manually after running the models, or through Winsteps directly. The purpose of this vignette is to provide a few illustrated examples of using the package.

Installation

For the time being, the *r2Winsteps* package is housed exclusively on *github*. Installation is straightforward via the *devtools* package. If you don't have *devtools* installed, you will need to first run the following:

```
install.packages("devtools")
```

Then, you just need to load the *devtools* package and install *r2Winsteps* directly from github.

```
library(devtools)
install_github("DJAnderson07/r2Winsteps")
```

Winsteps is exclusively designed for Windows, and if you are on a Windows machine you should be good to go from here. However, if you'd like to use a Mac, you can do so through *Wine* (see https://www.winehq.org), a free, open-source "compatibility layer" for running windows programs on OS X. There are several approaches you can take here, including using the Wine Bottler app, which is also free, (http://winebottler.kronenberg.org), and using it's graphic user interface to install *Winsteps*. I have had success with Wine Bottler in the past, but have also run into some issues that are difficult to track down. The most sure fire way to install Wine is through terminal. Complete directions are available here: http://www.davidbaumgold.com/tutorials/wine-mac/ . Briefly, you need to make sure you have Xcode installed, which is a base Mac product available through the app store. Next you'll install Homebrew through terminal by copying and pasting the following code

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

You'll be asked for your password (although nothing will appear as you type). You may also need to agree to the Xcode license. Run the following code

```
sudo xcodebuild -license
```

then type *agree* and hit enter.

Next run

```
brew tap caskroom/cask
```

followed by

```
brew cask install java xquartz
```

This may take a few minutes. Finally, you can install wine (which can take quite a while to install, depending on your Internet speed).

```
brew install wine
```

Now you should have Wine installed. The last step is to install Winsteps. To do this, first locate the executable installer file (.exe). Then type `wine`, followed by a space, and then drag the executable into terminal. This should copy the path to the executable. For example, it may looks something like the below

```
wine /Applications/v3.91/WinstepsPasswordInstall3910.exe
```

You will then be taken through the Winsteps installation process, just as you would if you were on a Windows machine (note, you will need your password here). If the above doesn't work, I would recommend looking at the link above. It has the same steps, but is more comprehensive.

One last Note: Older versions of Winsteps included different (less) output from the item and person files. The *r2Winsteps* package was built with *Winsteps* Version 3.90. If you have an earlier version of Winsteps, there are workarounds you can use in the options of the `read.pfile()` and `read.ifile()` functions, and you may want to consider modifying the source code for your particular installation.

Using the package: Example 1, LSAT data

Now that everything is installed, let's start with a simple example. We'll begin by fitting a Rasch model with dichotomous, using the supplied LSAT data.

```
library(r2Winsteps)
data(LSAT)
head(LSAT)
```

```
##     ID    Sex Ethnicity Item 1 Item 2 Item 3 Item 4 Item 5
## 1 1086   Male     Black      0      0      0      0      0
## 2  978   Male     White      0      0      0      0      0
## 3  958   Male    Latino      0      0      0      0      0
## 4  987 Female     White      0      0      0      0      1
## 5 1123 Female     White      0      0      0      0      1
## 6 1004   Male     White      0      0      0      0      1
```

It's generally a good idea to inspect some preliminary data, so we can get an idea of what to expect from the model, and whether the assumptions of the model appear tenable. First, we'll estimate the proportion of respondents responding correctly to the items. Because the items are dichotomous, this is just the mean.

```
apply(LSAT[ ,4:8], 2, mean)
```

```
## Item 1 Item 2 Item 3 Item 4 Item 5
##  0.924  0.709  0.553  0.763  0.870
```

So all items appear somewhat easy, but Item 1 is clearly the easiest while Item 3 is clearly the most difficult.

Next, we can compute the point-biserial correlation, by correlating the response vector for each item with a vector of raw scores. We'll compute the raw scores, and then compute the correlations.

```
raw <- rowSums(LSAT[ ,4:8])
round( sapply(LSAT[ ,4:8], function(i) cor(i, raw)), 2)
```

```
## Item 1 Item 2 Item 3 Item 4 Item 5
##   0.36   0.57   0.62   0.53   0.44
```

These are classical test theory indicators of *item discrimination*. The Rasch model assumes essentially equivalent item discrimination, and so we're looking to see if any items appear wildly different from the others. It's also worth noting that item-fit statistics reported by *Winsteps*, such as the mean square outfit, are produced by evaluating the differences from the *average* biserial correlation (see Wu and Adams 2012). These all appear reasonable, so let's go ahead and fit the model. In this case, because the model and data are straightforward, we simply need to call the `runWinsteps()` function, which requires the data be split into a data frame of item responses and a data frame of person demographics.

```
pars_LSAT <- runWinsteps(LSAT[ ,4:8], LSAT[ ,1:3])
str(pars_LSAT)
```

```
## List of 2
##  $ Item Parameters  :List of 1
##   ..$ r2WinstepsIfile:'data.frame':  5 obs. of  22 variables:
##   .. ..$ Entry               : int [1:5] 1 2 3 4 5
##   .. ..$ Difficulty          : num [1:5] -1.55 0.56 1.63 0.16 -0.81
##   .. ..$ Status              : int [1:5] 1 1 1 1 1
##   .. ..$ Count               : num [1:5] 1000 1000 1000 1000 1000
##   .. ..$ RawScore            : num [1:5] 924 709 553 763 870
##   .. ..$ SE                  : num [1:5] 0.13 0.08 0.08 0.09 0.11
##   .. ..$ Infit               : num [1:5] 1.01 0.98 1.01 0.99 1.01
##   .. ..$ Infit_Z             : num [1:5] 0.16 -0.5 0.27 -0.28 0.16
##   .. ..$ Outfit              : num [1:5] 1.05 0.97 1.01 0.98 1.02
##   .. ..$ Outfit_Z            : num [1:5] 0.41 -0.62 0.15 -0.29 0.27
##   .. ..$ Displacement        : num [1:5] 0 0 0 0 0
##   .. ..$ PointMeasureCorr    : num [1:5] 0.35 0.56 0.63 0.53 0.42
##   .. ..$ Weight              : num [1:5] 1 1 1 1 1
##   .. ..$ ObservMatch         : num [1:5] 89.6 69.4 65.1 73 83
##   .. ..$ ExpectMatch         : num [1:5] 89.8 67.9 65.9 72.6 83
##   .. ..$ PointMeasureExpected: num [1:5] 0.36 0.56 0.64 0.52 0.43
##   .. ..$ RMSR                : num [1:5] 0.29 0.45 0.45 0.43 0.36
##   .. ..$ WMLE                : num [1:5] -1.54 0.56 1.63 0.17 -0.8
```

```
##   .. ..$ Group               : int [1:5] 1 1 1 1 1
##   .. ..$ Model               : Factor w/ 1 level " R": 1 1 1 1 1
##   .. ..$ Recoding            : Factor w/ 1 level " .": 1 1 1 1 1
##   .. ..$ ItemID              : Factor w/ 5 levels " Item 1"," Item 2",..: 1 2 3 4 5
##  $ Person Parameters:List of 1
##   ..$ r2WinstepsPfile:'data.frame':  1000 obs. of  21 variables:
##   .. ..$ Entry               : int [1:1000] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..$ Theta               : num [1:1000] -3.23 -3.23 -3.23 -1.72 -1.72 -1.72 -1.72 -1.72 -1.72 -1
##   .. ..$ Status              : int [1:1000] -1 -1 -1 1 1 1 1 1 1 1 ...
##   .. ..$ Count               : num [1:1000] 5 5 5 5 5 5 5 5 5 5 ...
##   .. ..$ RawScore            : num [1:1000] 0 0 0 1 1 1 1 1 1 1 ...
##   .. ..$ SE                  : num [1:1000] 1.93 1.93 1.93 1.21 1.21 1.21 1.21 1.21 1.21 1.21 ...
##   .. ..$ Infit               : num [1:1000] 1 1 1 1.09 1.09 1.09 1.09 1.09 1.09 1.54 ...
##   .. ..$ Infit_Z             : num [1:1000] 0 0 0 0.35 0.35 0.35 0.35 0.35 0.35 0.95 ...
##   .. ..$ Outfit              : num [1:1000] 1 1 1 0.73 0.73 0.73 0.73 0.73 0.73 1.59 ...
##   .. ..$ Outfit_Z            : num [1:1000] 0 0 0 0.18 0.18 0.18 0.18 0.18 0.18 0.82 ...
##   .. ..$ Displacement        : num [1:1000] 0 0 0 0 0 0 0 0 0 0 ...
##   .. ..$ PointMeasureCorr    : num [1:1000] 0 0 0 0.37 0.37 0.37 0.37 0.37 0.37 -0.08 ...
##   .. ..$ Weight              : num [1:1000] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..$ ObservMatch         : num [1:1000] 100 100 100 80 80 80 80 80 80 80 ...
##   .. ..$ ExpectMatch         : num [1:1000] 100 100 100 80 80 80 80 80 80 80 ...
##   .. ..$ PointMeasureExpected: num [1:1000] 0 0 0 0.37 0.37 0.37 0.37 0.37 0.37 0.37 ...
##   .. ..$ RMSR                : num [1:1000] 0 0 0 0.39 0.39 0.39 0.39 0.39 0.39 0.46 ...
##   .. ..$ WMLE                : num [1:1000] -3.23 -3.23 -3.23 -1.41 -1.41 -1.41 -1.41 -1.41 -1.41 -1
##   .. ..$ ID                  : num [1:1000] 1086 978 958 987 1123 ...
##   .. ..$ Sex                 : Factor w/ 2 levels " Female "," Male   ": 2 2 2 1 1 2 2 2 1 1 ...
##   .. ..$ Ethnicity           : Factor w/ 5 levels " Asian"," Black",..: 2 5 3 5 5 5 4 5 2 5 ...
```

The function writes a *.bat* file and executes it to call Winsteps. Other functions in the package are then called to read the item and person parameters back into R.

References

Andrich, David. 1978. "A Rating Formulation for Ordered Response Categories." *Psychometrika* 43 (4). Springer: 561–73.

Masters, Geoff N. 1982. "A Rasch Model for Partial Credit Scoring." *Psychometrika* 47 (2). Springer: 149–74.

Wu, M, and RJ Adams. 2012. "Properties of Rasch Residual Fit Statistics." *Journal of Applied Measurement* 14 (4): 339–55.