IRT in R

Doug Tommet

2/15/2018

Libraries

```
library(readr)
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 3.4.2
```

```
library(tibble)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.4.3
```

```
library(MplusAutomation)
library(mirt)
```

```
## Warning: package 'mirt' was built under R version 3.4.3
```

```
## Loading required package: stats4
```

```
## Loading required package: lattice
```

```
library(lavaan)
```

```
## This is lavaan 0.5-23.1097
```

```
## lavaan is BETA software! Please report any bugs.
```

Read in the data

Read in the tester simulation data.

```
tester <- read_csv(file = "/Users/douglastommet/Dropbox/ADNIpsychometrics/PRIVATE/RNJ/tester.csv")
```

```
## Parsed with column specification:
## cols(
##   u1 = col_integer(),
##   u2 = col_integer(),
##   u3 = col_integer(),
##   u4 = col_integer(),
##   u5 = col_integer(),
##   u6 = col_integer(),
##   u7 = col_integer(),
##   u8 = col_integer(),
##   u9 = col_integer()
## )
```

Mplus - WLSMV theta parameterization

```
mod.mplus <- " f by  u1* u2 u3 u4 u5 u6 u7 u8 u9;
               f @1; "
mod.wlsmv.t.body <- mplusObject(
  VARIABLE = "CATEGORICAL ARE all;",
  ANALYSIS = "estimator = wlsmv; parameterization = theta;",
  MODEL = mod.mplus,
  OUTPUT = "",
  SAVEDATA = "",
  usevariables = c("u1", "u2", "u3", "u4", "u5", "u6", "u7", "u8", "u9"),
  rdata = tester
)
mod.wlsmv.t.fit <- mplusModeler(mod.wlsmv.t.body,
                                modelout = "mod.wlsmv.t.inp", run = TRUE,
                                hashfilename = FALSE, writeData = 'always')
```

```
## Wrote model to: mod.wlsmv.t.inp
```

```
## Wrote data to: mod.wlsmv.t.dat
```

```
## Warning in prepareMplusData(df = data[i, , drop = FALSE], keepCols = object
## $usevariables, : The file 'mod.wlsmv.t.dat' currently exists and will be
## overwritten
```

```
##
## Running model: mod.wlsmv.t.inp
## System command: cd "." && "/Applications/Mplus/mplus" "mod.wlsmv.t.inp"
## Reading model:  mod.wlsmv.t.out
```

```
param <- readModels("mod.wlsmv.t.out")$parameters$unstandardized %>%
  as_tibble()
```

```
## Reading model:  mod.wlsmv.t.out
```

```
param.wlsmv.theta <- param %>%
  rename(est_theta = est) %>%
  mutate(param = tolower(param))
```

Mplus - WLSMV delta parameterization

```
mod.wlsmv.d.body <- mplusObject(
  VARIABLE = "CATEGORICAL ARE all;",
  ANALYSIS = "estimator = wlsmv; parameterization = delta;",
  MODEL = mod.mplus,
  OUTPUT = "",
  SAVEDATA = "",
  usevariables = c("u1", "u2", "u3", "u4", "u5", "u6", "u7", "u8", "u9"),
  rdata = tester
)
mod.wlsmv.d.fit <- mplusModeler(mod.wlsmv.d.body,
                                modelout = "mod.wlsmv.d.inp", run = TRUE,
                                hashfilename = FALSE, writeData = 'always')
```

```
## Wrote model to: mod.wlsmv.d.inp
```

```
## Wrote data to: mod.wlsmv.d.dat
```

```
## Warning in prepareMplusData(df = data[i, , drop = FALSE], keepCols = object
## $usevariables, : The file 'mod.wlsmv.d.dat' currently exists and will be
## overwritten
```

```
##
## Running model: mod.wlsmv.d.inp
## System command: cd "." && "/Applications/Mplus/mplus" "mod.wlsmv.d.inp"
## Reading model:  mod.wlsmv.d.out
```

```
param <- readModels("mod.wlsmv.d.out")$parameters$unstandardized %>%
  as_tibble()
```

```
## Reading model:  mod.wlsmv.d.out
```

```
param.wlsmv.delta <- param %>%
  rename(est_delta = est) %>%
  mutate(param = tolower(param))
```

Mplus - MLR

```
mod.mlr.body <- mplusObject(
  VARIABLE = "CATEGORICAL ARE all;",
  ANALYSIS = "estimator = mlr; ",
  MODEL = mod.mplus,
  OUTPUT = "",
  SAVEDATA = "",
  usevariables = c("u1", "u2", "u3", "u4", "u5", "u6", "u7", "u8", "u9"),
  rdata = tester
)
mod.mlr.fit <- mplusModeler(mod.mlr.body,
                            modelout = "mod.mlr.inp", run = TRUE,
                            hashfilename = FALSE, writeData = 'always')
```

```
## Wrote model to: mod.mlr.inp
```

```
## Wrote data to: mod.mlr.dat
```

```
## Warning in prepareMplusData(df = data[i, , drop = FALSE], keepCols = object
## $usevariables, : The file 'mod.mlr.dat' currently exists and will be
## overwritten
```

```
##
## Running model: mod.mlr.inp
## System command: cd "." && "/Applications/Mplus/mplus" "mod.mlr.inp"
## Reading model:  mod.mlr.out
```

```
## No PROPORTION OF DATA PRESENT sections found within COVARIANCE COVERAGE OF DATA output.
```

```
param <- readModels("mod.mlr.out")$parameters$unstandardized %>%
  as_tibble()
```

```
## Reading model:  mod.mlr.out
```

```
## No PROPORTION OF DATA PRESENT sections found within COVARIANCE COVERAGE OF DATA output.
```

```
param.mlr <- param %>%
  rename(est_mlr = est) %>%
  mutate(param = tolower(param))
```

Mplus - Bayes

```
mod.bayes.body <- mplusObject(
  VARIABLE = "CATEGORICAL ARE all;",
  ANALYSIS = "estimator = bayes; ",
  MODEL = mod.mplus,
  OUTPUT = "",
  SAVEDATA = "",
  usevariables = c("u1", "u2", "u3", "u4", "u5", "u6", "u7", "u8", "u9"),
  rdata = tester
)
# mod.bayes.fit <- mplusModeler(mod.bayes.body,
#                               modelout = "mod.bayes.inp", run = TRUE,
#                               hashfilename = FALSE, writeData = 'always')

param <- readModels("mod.bayes.out")$parameters$unstandardized %>%
  as_tibble()
```

```
## Reading model:  mod.bayes.out
```

```
param.bayes <- param %>%
  rename(est_bayes = est) %>%
  mutate(param = tolower(param))
```

Mirt

```
mod.mirt <- mirt(tester, 1)
```

```
##
Iteration: 1, Log-Lik: -192023.608, Max-Change: 0.68539
Iteration: 2, Log-Lik: -186726.777, Max-Change: 0.56274
Iteration: 3, Log-Lik: -185487.476, Max-Change: 0.30051
Iteration: 4, Log-Lik: -184966.068, Max-Change: 0.20538
Iteration: 5, Log-Lik: -184734.757, Max-Change: 0.11239
Iteration: 6, Log-Lik: -184619.378, Max-Change: 0.08752
Iteration: 7, Log-Lik: -184550.889, Max-Change: 0.05454
Iteration: 8, Log-Lik: -184487.880, Max-Change: 0.04380
Iteration: 9, Log-Lik: -184458.065, Max-Change: 0.03829
Iteration: 10, Log-Lik: -184434.485, Max-Change: 0.03671
Iteration: 11, Log-Lik: -184419.066, Max-Change: 0.04024
Iteration: 12, Log-Lik: -184405.511, Max-Change: 0.02249
Iteration: 13, Log-Lik: -184390.744, Max-Change: 0.01721
Iteration: 14, Log-Lik: -184381.806, Max-Change: 0.01177
Iteration: 15, Log-Lik: -184376.337, Max-Change: 0.00897
Iteration: 16, Log-Lik: -184372.953, Max-Change: 0.00695
Iteration: 17, Log-Lik: -184370.890, Max-Change: 0.00593
Iteration: 18, Log-Lik: -184369.581, Max-Change: 0.00458
Iteration: 19, Log-Lik: -184367.471, Max-Change: 0.00063
Iteration: 20, Log-Lik: -184367.448, Max-Change: 0.00045
Iteration: 21, Log-Lik: -184367.437, Max-Change: 0.00037
Iteration: 22, Log-Lik: -184367.412, Max-Change: 0.00007
```

```
param.mirt <- coef(mod.mirt, simplify=TRUE)$items %>%
  as.data.frame() %>%
  rownames_to_column("param") %>%
  rename(a1_mirt = a1)
```

Mirt summary

```
summary(mod.mirt)
```

```
##       F1     h2
## u1 0.757 0.5727
## u2 0.532 0.2834
## u3 0.864 0.7472
## u4 0.857 0.7344
## u5 0.722 0.5206
## u6 0.301 0.0907
## u7 0.712 0.5063
## u8 0.495 0.2448
## u9 0.703 0.4935
##
## SS loadings:  4.194
## Proportion Var:  0.466
##
## Factor correlations:
##
##    F1
## F1  1
```

Mirt coef

```
coef(mod.mirt, simplify=TRUE)
```

```
## $items
##       a1      d g u
## u1 1.970 -4.380 0 1
## u2 1.070 -2.615 0 1
## u3 2.926 -3.549 0 1
## u4 2.830 -3.212 0 1
## u5 1.774 -2.070 0 1
## u6 0.538 -1.381 0 1
## u7 1.724 -1.632 0 1
## u8 0.969 -1.185 0 1
## u9 1.680 -1.250 0 1
##
## $means
## F1
##  0
##
## $cov
##    F1
## F1  1
```

Lavaan

```
mod.lavaan.body <- 'f =~ NA*u1 + u2 + u3 + u4 + u5 + u6 + u7 + u8 + u9
f~~1*f
'
mod.lavaan.fit <- sem(mod.lavaan.body,
                      data=tester ,
                      ordered=c("u1", "u2", "u3", "u4", "u5", "u6", "u7", "u8", "u9"))

param.lavaan <- parameterestimates(mod.lavaan.fit) %>%
  rename(est_lavaan = est)
```

Lavaan summary

```
summary(mod.lavaan.fit, fit.measures=TRUE, standardized=TRUE)
```

```
## lavaan (0.5-23.1097) converged normally after  11 iterations
##
##   Number of observations                            50000
##
##   Estimator                                          DWLS      Robust
##   Minimum Function Test Statistic                  12.004      18.568
##   Degrees of freedom                                  27          27
##   P-value (Chi-square)                             0.994       0.885
##   Scaling correction factor                                    0.657
##   Shift parameter                                              0.308
##     for simple second-order correction (Mplus variant)
##
## Model test baseline model:
##
##   Minimum Function Test Statistic           131325.879  101198.182
##   Degrees of freedom                                36          36
##   P-value                                        0.000       0.000
##
## User model versus baseline model:
##
##   Comparative Fit Index (CFI)                    1.000       1.000
##   Tucker-Lewis Index (TLI)                       1.000       1.000
##
##   Robust Comparative Fit Index (CFI)                            NA
##   Robust Tucker-Lewis Index (TLI)                               NA
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.000       0.000
##   90 Percent Confidence Interval          0.000  0.000       0.000  0.002
##   P-value RMSEA <= 0.05                          1.000       1.000
##
##   Robust RMSEA                                                  NA
##   90 Percent Confidence Interval                             0.000     NA
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.005       0.005
##
## Weighted Root Mean Square Residual:
##
##   WRMR                                           0.516       0.516
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Standard Errors                           Robust.sem
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   f =~
##     u1                0.703    0.008   91.099    0.000    0.703    0.703
##     u2                0.496    0.008   62.419    0.000    0.496    0.496
##     u3                0.853    0.004  206.122    0.000    0.853    0.853
##     u4                0.846    0.004  206.358    0.000    0.846    0.846
##     u5                0.711    0.005  138.736    0.000    0.711    0.711
##     u6                0.296    0.007   39.734    0.000    0.296    0.296
##     u7                0.707    0.005  142.783    0.000    0.707    0.707
##     u8                0.496    0.006   78.923    0.000    0.496    0.496
##     u9                0.701    0.005  143.558    0.000    0.701    0.701
##
```

```
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     .u1               0.000                              0.000    0.000
##     .u2               0.000                              0.000    0.000
##     .u3               0.000                              0.000    0.000
##     .u4               0.000                              0.000    0.000
##     .u5               0.000                              0.000    0.000
##     .u6               0.000                              0.000    0.000
##     .u7               0.000                              0.000    0.000
##     .u8               0.000                              0.000    0.000
##     .u9               0.000                              0.000    0.000
##      f                0.000                              0.000    0.000
##
## Thresholds:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     u1|t1             1.645    0.009  174.042    0.000    1.645    1.645
##     u2|t1             1.283    0.008  167.700    0.000    1.283    1.283
##     u3|t1             1.036    0.007  151.276    0.000    1.036    1.036
##     u4|t1             0.961    0.007  144.354    0.000    0.961    0.961
##     u5|t1             0.835    0.006  130.952    0.000    0.835    0.835
##     u6|t1             0.793    0.006  126.001    0.000    0.793    0.793
##     u7|t1             0.669    0.006  109.995    0.000    0.669    0.669
##     u8|t1             0.614    0.006  102.201    0.000    0.614    0.614
##     u9|t1             0.520    0.006   88.309    0.000    0.520    0.520
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     f                 1.000                              1.000    1.000
##     .u1               0.506                              0.506    0.506
##     .u2               0.754                              0.754    0.754
##     .u3               0.273                              0.273    0.273
##     .u4               0.285                              0.285    0.285
##     .u5               0.494                              0.494    0.494
##     .u6               0.913                              0.913    0.913
##     .u7               0.501                              0.501    0.501
##     .u8               0.754                              0.754    0.754
##     .u9               0.509                              0.509    0.509
##
## Scales y*:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     u1                1.000                              1.000    1.000
##     u2                1.000                              1.000    1.000
##     u3                1.000                              1.000    1.000
##     u4                1.000                              1.000    1.000
##     u5                1.000                              1.000    1.000
##     u6                1.000                              1.000    1.000
##     u7                1.000                              1.000    1.000
##     u8                1.000                              1.000    1.000
##     u9                1.000                              1.000    1.000
```

Results

Merging the slope results together.

```
param.truth <- tibble(param = c("u1", "u2", "u3", "u4", "u5", "u6", "u7", "u8", "u9"),
                      truth.slope = c("1.000", "0.577", "1.614", "1.614", "1.000", "0.314", "1.000", "0.577", "1.000"))

param.wlsmv.theta.sub <- param.wlsmv.theta %>%
  filter(paramHeader=="F.BY") %>%
  select(param, est_theta) %>%
  rename(slope_theta = est_theta)

param.wlsmv.delta.sub <- param.wlsmv.delta %>%
  filter(paramHeader=="F.BY") %>%
  select(param, est_delta) %>%
  rename(slope_delta = est_delta)

param.mlr.sub <- param.mlr %>%
  filter(paramHeader=="F.BY") %>%
  select(param, est_mlr) %>%
  rename(slope_mlr = est_mlr)

param.bayes.sub <- param.bayes %>%
  filter(paramHeader=="F.BY") %>%
  select(param, est_bayes) %>%
  rename(slope_bayes = est_bayes)

param.mirt.sub <- param.mirt %>%
  select(param, a1_mirt) %>%
  rename(slope_mirt = a1_mirt)

param.lavaan.sub <- param.lavaan %>%
  filter(op=="=~") %>%
  rename(param = rhs) %>%
  select(param, est_lavaan) %>%
  rename(slope_lavaan = est_lavaan)

param.table.slope <- param.truth %>%
  left_join(param.wlsmv.theta.sub, by = "param") %>%
  left_join(param.wlsmv.delta.sub, by = "param") %>%
  left_join(param.mlr.sub, by = "param") %>%
  left_join(param.bayes.sub, by = "param") %>%
  left_join(param.mirt.sub, by = "param") %>%
  left_join(param.lavaan.sub, by = "param")
```

```
param.truth <- tibble(param = c("u1", "u2", "u3", "u4", "u5", "u6", "u7", "u8", "u9"),
                      truth.thresh = c("2.327", "2.563", "1.219", "1.123", "1.190", "2.574", "0.954", "1.226", "0.742"))

param.wlsmv.theta.sub <- param.wlsmv.theta %>%
  filter(paramHeader=="Thresholds") %>%
  separate(param, c("param", "threshold"), sep = "\\$") %>%
  select(param, est_theta) %>%
  rename(thres_theta = est_theta)

param.wlsmv.delta.sub <- param.wlsmv.delta %>%
  filter(paramHeader=="Thresholds") %>%
  separate(param, c("param", "threshold"), sep = "\\$") %>%
  select(param, est_delta) %>%
  rename(thres_delta = est_delta)

param.mlr.sub <- param.mlr %>%
  filter(paramHeader=="Thresholds") %>%
```

```
  filter(paramHeader== Thresholds ) %>%
  separate(param, c("param", "threshold"), sep = "\\$") %>%
  select(param, est_mlr) %>%
  rename(thres_mlr = est_mlr)

param.bayes.sub <- param.bayes %>%
  filter(paramHeader=="Thresholds") %>%
  separate(param, c("param", "threshold"), sep = "\\$") %>%
  select(param, est_bayes) %>%
  rename(thres_bayes = est_bayes)

param.mirt.sub <- param.mirt %>%
  select(param, d) %>%
  rename(d_mirt = d)

param.lavaan.sub <- param.lavaan %>%
  filter(op=="|") %>%
  rename(param = lhs) %>%
  select(param, est_lavaan) %>%
  rename(thres_lavaan = est_lavaan)

param.table.threshold <- param.truth %>%
  left_join(param.wlsmv.theta.sub, by = "param") %>%
  left_join(param.wlsmv.delta.sub, by = "param") %>%
  left_join(param.mlr.sub, by = "param") %>%
  left_join(param.bayes.sub, by = "param") %>%
  left_join(param.mirt.sub, by = "param") %>%
  left_join(param.lavaan.sub, by = "param")
```

```
param.table <- param.table.slope %>%
  left_join(param.table.threshold, by = "param")
```

Slope estimate table

These are the slope estimates, not standardized, under various estimator and parameterization options.

```
param.table %>%
  select(param, truth.slope, starts_with("slope_")) %>%
  kable(digits = 3)
```

param	truth.slope	slope_theta	slope_delta	slope_mlr	slope_bayes	slo
u1	1.000	0.987	0.703	1.969	0.990	
u2	0.577	0.571	0.496	1.070	0.564	
u3	1.614	1.632	0.853	2.926	1.642	
u4	1.614	1.585	0.846	2.830	1.591	
u5	1.000	1.012	0.711	1.774	1.007	
u6	0.314	0.310	0.296	0.538	0.311	
u7	1.000	0.999	0.707	1.724	1.000	
u8	0.577	0.571	0.496	0.969	0.572	
u9	1.000	0.982	0.701	1.681	0.978	

2PL slope estimate table

These are the slope estimates converted to 2PL a-parameters using **math**.

```
param.table <- param.table %>%
  mutate(a_theta = slope_theta,
         a_delta = slope_delta/sqrt(1-slope_delta^2),
         a_mlr   = slope_mlr/1.7,
         a_bayes = slope_bayes,
         a_mirt  = slope_mirt/1.7,
         a_lavaan = slope_lavaan/sqrt(1-slope_lavaan^2))
```

```
param.table %>%
  select(param, truth.slope, starts_with("a_")) %>%
  kable(digits = 3)
```

param	truth.slope	a_theta	a_delta	a_mlr	a_bayes	a_mirt	a_lavaan
u1	1.000	0.987	0.988	1.158	0.990	1.159	0.987
u2	0.577	0.571	0.571	0.629	0.564	0.630	0.571
u3	1.614	1.632	1.634	1.721	1.642	1.721	1.632
u4	1.614	1.585	1.587	1.665	1.591	1.665	1.585
u5	1.000	1.012	1.011	1.044	1.007	1.043	1.012
u6	0.314	0.310	0.310	0.316	0.311	0.316	0.309
u7	1.000	0.999	1.000	1.014	1.000	1.014	0.999
u8	0.577	0.571	0.571	0.570	0.572	0.570	0.571
u9	1.000	0.982	0.983	0.989	0.978	0.988	0.982

Threshold estimate table

These are threshold estimates, not standardized, under various estimator and parameterization options.

```
param.table %>%
  select(param, starts_with("thres_"), d_mirt) %>%
  kable(digits = 3)
```

param	thres_theta	thres_delta	thres_mlr	thres_bayes	thres_lavaan	d_
u1	2.312	1.645	4.378	2.317	1.645	
u2	1.477	1.283	2.615	1.473	1.283	
u3	1.982	1.036	3.548	1.990	1.036	
u4	1.800	0.961	3.211	1.805	0.961	
u5	1.188	0.835	2.069	1.185	0.835	
u6	0.830	0.793	1.381	0.831	0.793	
u7	0.946	0.669	1.632	0.947	0.669	
u8	0.707	0.614	1.185	0.708	0.614	
u9	0.729	0.520	1.250	0.728	0.520	

2PL location estimate

These are the threshold estimates converted to 2PL b-parameters using **math**.

```
param.table <- param.table %>%
  mutate(b_theta = thres_theta/slope_theta,
         b_delta = thres_delta/slope_delta,
         b_mlr   = thres_mlr/slope_mlr,
         b_bayes = thres_bayes/slope_bayes,
         b_lavaan = thres_lavaan/slope_lavaan,
         b_mirt   = -1 * d_mirt/slope_mirt)
```

```
param.table %>%
  select(param, truth.thresh, starts_with("b_")) %>%
  kable(digits = 3)
```

param	truth.thresh	b_theta	b_delta	b_mlr	b_bayes	b_lavaan	b_mirt
u1	2.327	2.342	2.340	2.223	2.340	2.342	2.223
u2	2.563	2.587	2.587	2.444	2.612	2.587	2.444
u3	1.219	1.214	1.215	1.213	1.212	1.215	1.213
u4	1.123	1.136	1.136	1.135	1.135	1.136	1.135
u5	1.190	1.174	1.174	1.166	1.177	1.174	1.167
u6	2.574	2.677	2.679	2.567	2.672	2.683	2.569
u7	0.954	0.947	0.946	0.947	0.947	0.947	0.947
u8	1.226	1.238	1.238	1.223	1.238	1.238	1.223
u9	0.742	0.742	0.742	0.744	0.744	0.742	0.744