Bias Reduction PML

2023-11-22

Introduction

Let $\mathbf{Y} = (Y_1, \dots, Y_p)^\top \in \{0,1\}^p$ be a vector of Bernoulli random variables. Consider a response pattern $\mathbf{y} = (y_1, \dots, y_p)^\top$, where each $y_i \in \{0,1\}$. The probability of observing such a response pattern is given by the joint distribution

$$\pi = \Pr(\mathbf{Y} = \mathbf{y}) = \Pr(Y_1 = y_1, \dots, Y_p = y_p). \tag{1}$$

Note that there are a total of $R = 2^p$ possible joint probabilities π_r corresponding to all possible two-way response patterns $\mathbf{y}_r$.

When we consider a parametric model with parameter vector $\boldsymbol{\theta} \in \mathbb{R}^m$, we write $\pi_r(\boldsymbol{\theta})$ to indicate each joint probability, and

$$\boldsymbol{\pi}(\boldsymbol{\theta}) = \begin{pmatrix} \pi_1(\boldsymbol{\theta}) \\ \vdots \\ \pi_R(\boldsymbol{\theta}) \end{pmatrix} \in [0,1]^R \tag{2}$$

for the vector of joint probabilities, with $\sum_{r=1}^R \pi_r(\boldsymbol{\theta}) = 1$.

Binary factor models

The model of interest is a factor model, commonly used in social statistics. Using an underlying variable (UV) approach, the observed binary responses y_i are manifestations of some latent, continuous variables Y_i^*, $i = 1, \dots, p$. The connection is made as follows:

$$Y_i = \begin{cases} 1 & Y_i^* > \tau_i \\ 0 & Y_i^* \le \tau_i, \end{cases}$$

where τ_i is the threshold associated with the variable Y_i^*. For convenience, Y_i^* is taken to be standard normal random variables[1]. The factor model takes the form

$$\mathbf{Y}^* = \boldsymbol{\Lambda}\boldsymbol{\eta} + \boldsymbol{\epsilon},$$

where each component is explained below:

- $\mathbf{Y}^* = (Y_1^*, \dots, Y_p^*)^\top \in \mathbf{R}^p$ are the underlying variables;
- $\boldsymbol{\Lambda} \in \mathbf{R}^{p \times q}$ is the matrix of loadings;
- $\boldsymbol{\eta} = (\eta_1, \dots, \eta_q)^\top \in \mathbf{R}^q$ is the vector of latent factors;
- $\boldsymbol{\epsilon} \in \mathbf{R}^p$ are the error terms associated with the items (aka unique variables).

We also make some distributional assumptions, namely

[1]For parameter identifiability, the location and scale of the normal distribution have to be fixed if the thresholds are to be estimated.

1. $\boldsymbol{\eta} \sim \mathrm{N}_q(\mathbf{0}, \boldsymbol{\Psi})$, where $\boldsymbol{\Psi}$ is a correlation matrix, i.e. for $k, l \in \{1, \dots, q\}$,

$$\boldsymbol{\Psi}_{kl} = \begin{cases} 1 & \text{if } k = l \\ \rho(\eta_k, \eta_l) & \text{if } k \neq l. \end{cases}$$

2. $\boldsymbol{\epsilon} \sim \mathrm{N}_p(\mathbf{0}, \boldsymbol{\Theta}_\epsilon)$, with $\boldsymbol{\Theta}_\epsilon = \mathbf{I} - \operatorname{diag}(\boldsymbol{\Lambda}\boldsymbol{\Psi}\boldsymbol{\Lambda}^\top)$.

These two assumptions, together with $\operatorname{Cov}(\boldsymbol{\eta}, \boldsymbol{\epsilon}) = 0$, implies that $\mathbf{Y}^* \sim \mathrm{N}_p(\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{y}^*})$, where

$$\boldsymbol{\Sigma}_{\mathbf{y}^*} = \operatorname{Var}(\mathbf{Y}^*) = \boldsymbol{\Lambda}\boldsymbol{\Phi}\boldsymbol{\Lambda}^\top + \boldsymbol{\Theta}_\epsilon. \tag{3}$$

The parameter vector for this factor model is denoted $\boldsymbol{\theta}^\top = (\boldsymbol{\lambda}, \boldsymbol{\psi}, \boldsymbol{\tau}) \in \mathbb{R}^m$, where it contains the vectors of the free non-redundant parameters in $\boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}$ respectively, as well as the vector of all free thresholds.

Under this factor model, the probability of response pattern $\mathbf{y}_r$ is

$$\begin{aligned} \pi_r(\boldsymbol{\theta}) &= \Pr(\mathbf{Y} = \mathbf{y}_r \mid \boldsymbol{\theta}) &(4)\\ &= \int \cdots \int_A \phi_p(\mathbf{y}^* \mid \mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{y}^*}) \, \mathrm{d}\mathbf{y}^* &(5) \end{aligned}$$

where $\phi_p(\cdot \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the density function of the p-dimensional normal distribution with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$. This integral is evaluated on the set

$$A = \{\mathbf{Y}^* \in \mathbb{R}^p \mid Y_1 = y_1, \dots, Y_p = y_p\}.$$

Pairwise likelihood estimation

In order to define the pairwise likelihood, let $\pi^{(ij)}_{y_i y_j}(\boldsymbol{\theta})$ be the probability under the model that $Y_i = y_i \in \{0, 1\}$ and $Y_j = y_j \in \{0, 1\}$ for a pair of variables Y_i and Y_j, $i, j = 1, \dots, p$ and $i < j$. The pairwise log-likelihood takes the form

$$\ell_{\mathrm{P}}(\boldsymbol{\theta}) = \sum_{i<j} \sum_{y_i} \sum_{y_j} \hat{n}^{(ij)}_{y_i y_j} \log \pi^{(ij)}_{y_i y_j}(\boldsymbol{\theta}), \tag{6}$$

where $\hat{n}^{(ij)}_{y_i y_j}$ is the observed (weighted) frequency of sample units with $Y_i = y_i$ and $Y_j = y_j$,

$$\hat{n}^{(ij)}_{y_i y_j} = \sum_h w_h [\mathbf{y}^{(h)}_i = y_i, \mathbf{y}^{(h)}_j = y_j].$$

Here the w_h refers to the design weight for any individual h in the sample. For simplicity, we may assume that these weights are normalised such that $\sum w_h = N$. In such a case, a simple random sampling design would imply all weights are equal to one, and the weighted pairwise likelihood reduces to the usual pairwise likelihood function.

Let us also define the corresponding observed pairwise proportions $p^{(ij)}_{y_i y_j} = \hat{n}^{(ij)}_{y_i y_j}/n$. There are a total of $\tilde{R} = 4 \times \binom{p}{2}$ summands, where the '4' is representative of the total number of pairwise combinations of binary choices '00', '10', '01', and '11'.

The *pairwise maximum likelihood estimator* $\hat{\boldsymbol{\theta}}_{\mathrm{PL}}$ satisfies $\hat{\boldsymbol{\theta}}_{\mathrm{PL}} = \operatorname{argmax}_{\boldsymbol{\theta}} \ell_{\mathrm{P}}(\boldsymbol{\theta})$. Under certain regularity conditions,

$$\sqrt{N}(\hat{\boldsymbol{\theta}}_{\mathrm{PL}} - \boldsymbol{\theta}) \xrightarrow{D} \mathrm{N}_m\big(\mathbf{0}, \mathcal{H}(\boldsymbol{\theta})\mathcal{J}(\boldsymbol{\theta})^{-1}\mathcal{H}(\boldsymbol{\theta})\big), \tag{7}$$

where

- $\mathcal{H}(\boldsymbol{\theta}) = -\operatorname{E} \nabla^2 \ell_{\mathrm{P}}(\boldsymbol{\theta}; \mathbf{y}^{(h)})$ is the *sensitivity matrix*; and
- $\mathcal{J}(\boldsymbol{\theta}) = \operatorname{Var}\big(\nabla \ell_{\mathrm{P}}(\boldsymbol{\theta}; \mathbf{y}^{(h)})\big)$ is the *variability matrix*.

In practice, we may estimate these matrices using the following estimators:

$$\hat{\mathbf{H}} := \mathbf{H}(\hat{\boldsymbol\theta}) = -\frac{1}{\sum_h w_h} \sum_{h=1}^{N} w_h \nabla^2 \ell_\text{P}(\boldsymbol\theta; \mathbf{y}^{(h)}) \Big|_{\boldsymbol\theta=\hat{\boldsymbol\theta}}$$

$$\hat{\mathbf{J}} := \mathbf{J}(\hat{\boldsymbol\theta}) = \frac{1}{\sum_h w_h} \sum_{h=1}^{N} w_h^2 \nabla \ell_\text{P}(\boldsymbol\theta; \mathbf{y}^{(h)}) \nabla \ell_\text{P}(\boldsymbol\theta; \mathbf{y}^{(h)})^\top \Big|_{\boldsymbol\theta=\hat{\boldsymbol\theta}}$$

That is, $\hat{\mathbf{H}}$ is the Hessian resulting from the optimisation of the pairwise likelihood function, while $\hat{\mathbf{J}}$ is the cross product of the gradient of the pairwise likelihood function–each evaluated at the maximum PLE. Note that both are considered "unit" information matrices, as they are normalised by the sum of the weights (sample size).

Bias reduction

Define

$$A(\hat{\boldsymbol\theta}) = -\frac{1}{2} \nabla \operatorname{tr}\left(\mathbf{H}(\boldsymbol\theta)^{-1} \mathbf{J}(\boldsymbol\theta)\right) \Big|_{\boldsymbol\theta=\hat{\boldsymbol\theta}}.$$

Then, an improved estimator $\tilde{\boldsymbol\theta}$ is given by

$$\tilde{\boldsymbol\theta} = \hat{\boldsymbol\theta} + \frac{1}{\sum_h w_h} \mathbf{H}(\hat{\boldsymbol\theta})^{-1} A(\hat{\boldsymbol\theta}).$$

Some computational notes:

- The Hessian $\mathbf{H}(\boldsymbol\theta)$ matrix is obtained as a byproduct of the optimisation routine in `{lavaan}` (or using my manually coded pml function and optim). There is no explicit code for it.
- The variability matrix $\mathbf{J}(\boldsymbol\theta)$ is obtained from `{lavaan}`, by tricking it into accepting starting values $\boldsymbol\theta$ as converged parameter values and extracting the $\mathbf{J}(\boldsymbol\theta)$ accordingly.
- Then form the $\mathbf{A}(\boldsymbol\theta)$ matrix and obtain the gradient using the `numDeriv` package.

Example

Consider $p = 5$ binary items generated using the UV approach as above with the following true parameter values:

- $\boldsymbol\lambda$ = 0.8, 0.7, 0.47, 0.38, 0.34
- $\boldsymbol\tau$ = -1.43, -0.55, -0.13, -0.72, -1.13

NB: This is called Model no 1 from a previous work (Jamil, Moustaki, and Skinner 2023).

```
set.seed(123)
model_no <- 1
mod <- txt_mod(model_no)
(dat <- gen_data_bin(model_no, n = 500))
```

```
## # A tibble: 500 x 5
##    y1    y2    y3    y4    y5
##    <ord> <ord> <ord> <ord> <ord>
##  1 1     1     1     1     1
##  2 1     1     1     1     1
##  3 1     0     0     1     1
##  4 1     1     0     1     1
##  5 1     1     1     0     0
##  6 1     1     0     0     1
##  7 0     0     0     0     1
##  8 1     1     1     1     1
##  9 1     1     1     1     1
## 10 1     1     1     1     1
## # i 490 more rows
```

The fit from the PL routine is obtained from {lavaan}:

```
fit_lav <- sem(mod, dat, std.lv = TRUE, estimator = "PML")
```

```
## Warning in lav_object_post_check(object): lavaan WARNING: some estimated ov variances are negative
```

```
summary(fit_lav)
```

```
## lavaan 0.6.17.9001 ended normally after 18 iterations
##
##   Estimator                                        PML
##   Optimization method                           NLMINB
##   Number of model parameters                        10
##
##   Number of observations                           100
##
## Model Test User Model:
##                                               Standard      Scaled
##   Test Statistic                                 7.497      -1.015
##   Degrees of freedom                                 5      -0.339
##   P-value (Unknown)                                 NA          NA
##   Scaling correction factor                                 -7.384
##     mean+var adjusted correction (PML)
##
## Parameter Estimates:
##
##   Standard errors                             Sandwich
##   Information bread                           Observed
##   Observed information based on                Hessian
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   eta1 =~
##     y1                1.019    0.247    4.133    0.000
##     y2                0.632    0.204    3.097    0.002
##     y3                0.387    0.251    1.542    0.123
##     y4                0.513    0.198    2.590    0.010
##     y5                0.393    0.267    1.472    0.141
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .y1                0.000
##    .y2                0.000
##    .y3                0.000
##    .y4                0.000
##    .y5                0.000
##     eta1              0.000
##
```

```
## Thresholds:
##                    Estimate  Std.Err  z-value  P(>|z|)
##     y1|t1            -1.475    0.190   -7.758    0.000
##     y2|t1            -0.523    0.132   -3.980    0.000
##     y3|t1            -0.126    0.126   -1.005    0.315
##     y4|t1            -0.705    0.137   -5.139    0.000
##     y5|t1            -1.175    0.162   -7.246    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .y1               -0.039
##    .y2                0.600
##    .y3                0.850
##    .y4                0.737
##    .y5                0.846
##     eta1              1.000
##
## Scales y*:
##                    Estimate  Std.Err  z-value  P(>|z|)
##     y1                1.000
##     y2                1.000
##     y3                1.000
##     y4                1.000
##     y5                1.000
```

We compare the fit of the coefficients and the true value:

```r
theta_hat <- coef(fit_lav)
theta_true <- c(lavaan.bingof:::get_Lambda(model_no),
                lavaan.bingof:::get_tau(model_no))
tibble(
  coef = names(theta_hat),
  theta_hat = round(theta_hat, 2),
  theta_true = theta_true
) |>
  kbl(booktabs = TRUE)
```

coef	theta_hat	theta_true
eta1=~y1	1.02	0.80
eta1=~y2	0.63	0.70
eta1=~y3	0.39	0.47
eta1=~y4	0.51	0.38
eta1=~y5	0.39	0.34
y1|t1	-1.48	-1.43
y2|t1	-0.52	-0.55
y3|t1	-0.13	-0.13
y4|t1	-0.71	-0.72
y5|t1	-1.18	-1.13

Now apply the bias reduction method

```r
# Assume HHH() and JJJ() are the functions to obtain the H and J matrices at a
# given theta
AAA <- function(.theta) {
  tmp <- function(x) {
    Hinv <- HHH(x, unit = TRUE) |> solve()
    J    <- JJJ(x)
    -0.5 * sum(diag(Hinv %*% J))
  }
  numDeriv::grad(tmp, .theta)
}

# Bias reduction
(A <- AAA(theta_hat))
```

```
##  [1]  1.2990871  1.9927353 -1.6357574  1.6324533 -0.1273744  2.6498867  0.7775249 -0.2887947  1.1881
```

```
Hinv <- solve(HHH(theta_hat, unit = FALSE))
theta_tilde <- drop(theta_hat + Hinv %*% A)
```

coef	theta_hat	theta_true	theta_tilde
eta1=~y1	1.02	0.80	1.11
eta1=~y2	0.63	0.70	0.65
eta1=~y3	0.39	0.47	0.32
eta1=~y4	0.51	0.38	0.54
eta1=~y5	0.39	0.34	0.35
y1|t1	-1.48	-1.43	-1.45
y2|t1	-0.52	-0.55	-0.52
y3|t1	-0.13	-0.13	-0.13
y4|t1	-0.71	-0.72	-0.70
y5|t1	-1.18	-1.13	-1.16

A small simulation study

```
## A dummy fit to get par names
par_names <- names(theta_hat)

## Simulation studies
set.seed(123)
n_simu <- 100
n_cores <- parallel::detectCores()
model_no <- 1
theta_true <- c(lavaan.bingof:::get_Lambda(model_no),
                lavaan.bingof:::get_tau(model_no))
mod <- txt_mod(model_no)
datasets <- replicate(n_simu, gen_data_bin(model_no, n = 500), simplify = FALSE)

failed <- function(j, n_simu) cat("Sample:", j, "/", n_simu, ": Failed\n")
done <- function(j, n_simu) cat("Sample:", j, "/", n_simu, ": Done\n")

results <- parallel::mclapply(seq.int(n_simu), function(j) {
                         fit_lav <- try(sem(mod, datasets[[j]], std.lv = TRUE, estimator = "PML"), siler
                         if (is(fit_lav, "try-error")) {
                             mpl <- eRBM <- rep(NA, length(theta_true))
                             failed(j, n_simu)
                         } else {
                              mpl <- coef(fit_lav)
                              A <- try(AAA(mpl), silent = TRUE)
                              Hinv <- try(solve(HHH(mpl, unit = FALSE)), silent = TRUE)
                              if (is(A, "try-error") | is(Hinv, "try-error")) {
                                  eRBM <- rep(NA, length(theta_true))
                                  failed(j, n_simu)
                              } else {
                                  eRBM <- drop(mpl + Hinv %*% A)
                                  done(j, n_simu)
                              }
                          }
                          data.frame(estimate = c(mpl, eRBM),
                                     method = rep(c("MCL", "eRBM"), each = length(par_names)),
                                     truth = c(theta_true, theta_true),
                                     sample = j,
                                     parameter = rep(par_names, 2))
                      }, mc.cores = n_cores)
```

Let's get some sampling distribution summaries, ignoring failed cases

```
res <- do.call("rbind", results)
res |> group_by(parameter, method) |>
    filter(!is.na(estimate)) |>
    summarize(bias = mean(estimate - truth),
              sd = sd(estimate),
              abs_bias = mean(abs(estimate - truth)),
              rmse = sqrt(mean((estimate - truth)^2)),
              min = min(estimate),
              max = max(estimate),
              n_used = n()) |>
  kbl(booktabs = TRUE, digits = 2, linesep = "")
```

```
## `summarise()` has grouped output by 'parameter'. You can override using the
## `.groups` argument.
```

parameter	method	bias	sd	abs_bias	rmse	min	max	n_used
eta1=~y1	MCL	-0.01	0.37	0.24	0.37	-0.58	2.00	100
eta1=~y1	eRBM	-253.29	2578.31	431.61	2577.04	-24024.93	4813.29	94
eta1=~y2	MCL	-0.04	0.36	0.24	0.36	-0.96	1.63	100
eta1=~y2	eRBM	1311.93	13115.42	1526.82	13111.28	-4015.66	126941.99	94
eta1=~y3	MCL	-0.05	0.30	0.21	0.31	-0.80	1.39	100
eta1=~y3	eRBM	-928.17	9025.83	1087.24	9025.54	-87193.00	5451.51	94
eta1=~y4	MCL	-0.02	0.24	0.18	0.24	-0.41	0.79	100
eta1=~y4	eRBM	-229.62	2078.49	281.25	2080.11	-19999.94	997.46	94
eta1=~y5	MCL	-0.07	0.28	0.21	0.29	-0.97	1.00	100
eta1=~y5	eRBM	-348.28	3672.09	514.96	3669.07	-35108.42	2899.78	94
y1|t1	MCL	-0.02	0.19	0.15	0.19	-2.06	-1.08	100
y1|t1	eRBM	-60.07	566.71	63.03	566.88	-5495.09	100.17	94
y2|t1	MCL	-0.01	0.15	0.12	0.15	-0.92	-0.20	100
y2|t1	eRBM	32.50	324.54	36.41	324.44	-75.14	3142.87	94
y3|t1	MCL	0.00	0.13	0.11	0.13	-0.50	0.18	100
y3|t1	eRBM	-2.29	17.64	3.85	17.69	-148.88	47.84	94
y4|t1	MCL	-0.01	0.14	0.12	0.14	-1.08	-0.41	100
y4|t1	eRBM	-5.37	49.69	6.84	49.71	-477.72	32.82	94
y5|t1	MCL	-0.02	0.16	0.13	0.16	-1.55	-0.81	100
y5|t1	eRBM	3.40	35.68	6.00	35.65	-54.58	332.00	94

and thresholding the estimates to be less than 3 (arbitrarily set here)

```r
res |> group_by(parameter, method) |>
    filter(!is.na(estimate), abs(estimate) < 3) |>
    summarize(bias = mean(estimate - truth),
              sd = sd(estimate),
              abs_bias = mean(abs(estimate - truth)),
              rmse = sqrt(mean((estimate - truth)^2)),
              min = min(estimate),
              max = max(estimate),
              n_used = n()) |>
  kbl(booktabs = TRUE, digits = 2, linesep = "")
```

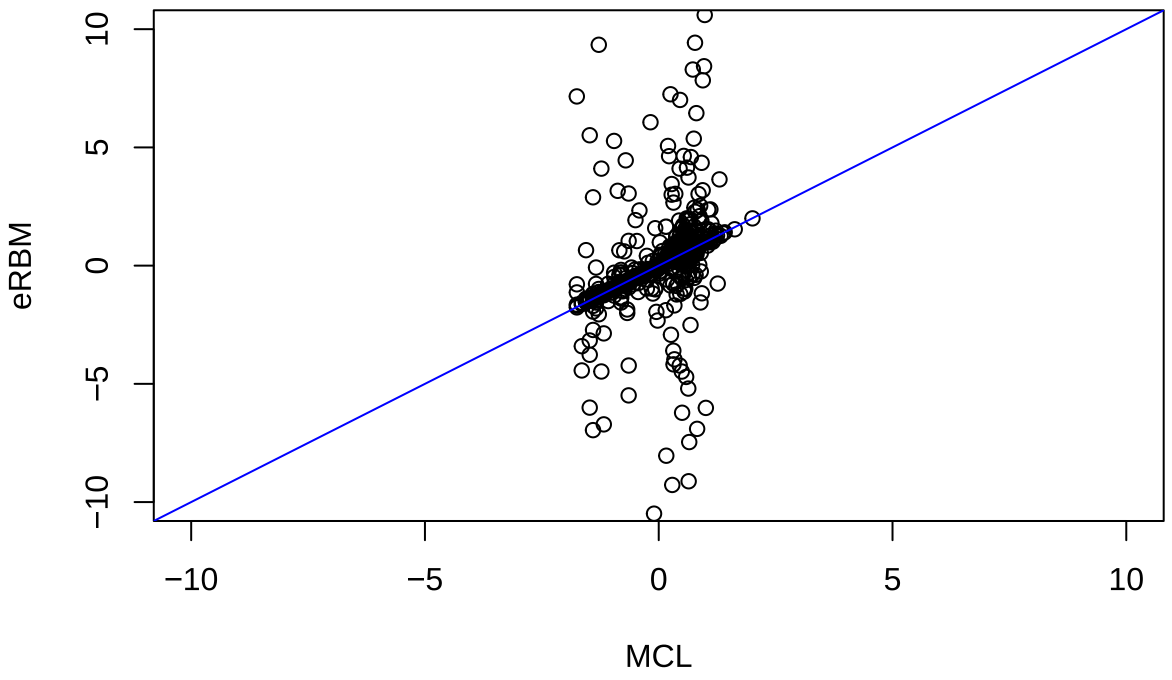
```
## `summarise()` has grouped output by 'parameter'. You can override using the
## `.groups` argument.
```

parameter	method	bias	sd	abs_bias	rmse	min	max	n_used
eta1=~y1	MCL	-0.01	0.37	0.24	0.37	-0.58	2.00	100
eta1=~y1	eRBM	0.23	0.67	0.53	0.70	-0.76	2.53	62
eta1=~y2	MCL	-0.04	0.36	0.24	0.36	-0.96	1.63	100
eta1=~y2	eRBM	0.02	0.72	0.52	0.71	-1.56	2.00	66
eta1=~y3	MCL	-0.05	0.30	0.21	0.31	-0.80	1.39	100
eta1=~y3	eRBM	-0.15	0.74	0.49	0.75	-2.51	2.67	68
eta1=~y4	MCL	-0.02	0.24	0.18	0.24	-0.41	0.79	100
eta1=~y4	eRBM	0.16	0.72	0.46	0.73	-2.92	2.45	73
eta1=~y5	MCL	-0.07	0.28	0.21	0.29	-0.97	1.00	100
eta1=~y5	eRBM	-0.19	0.47	0.33	0.51	-1.68	0.97	68
y1|t1	MCL	-0.02	0.19	0.15	0.19	-2.06	-1.08	100
y1|t1	eRBM	0.05	0.57	0.24	0.57	-2.87	2.89	78
y2|t1	MCL	-0.01	0.15	0.12	0.15	-0.92	-0.20	100
y2|t1	eRBM	0.04	0.44	0.22	0.44	-2.00	1.92	86
y3|t1	MCL	0.00	0.13	0.11	0.13	-0.50	0.18	100
y3|t1	eRBM	-0.07	0.44	0.23	0.44	-2.32	1.58	86
y4|t1	MCL	-0.01	0.14	0.12	0.14	-1.08	-0.41	100
y4|t1	eRBM	0.04	0.45	0.21	0.45	-1.85	2.33	85
y5|t1	MCL	-0.02	0.16	0.13	0.16	-1.55	-0.81	100
y5|t1	eRBM	0.04	0.32	0.18	0.32	-2.06	0.65	84

There seem to be some instabilities when computing the numerical derivatives. I am not sure if these are due to the indirect way of getting H and J, or explicit RBM (eRBM) inheriting instabilities from the MCL estimator.

For example, from the below figure I would only trust the estimates around the slope-one line through the origin.

```r
plot(res$estimate[res$method == "MCL"],
     res$estimate[res$method == "eRBM"],
     xlab = "MCL",
     ylab = "eRBM",
     xlim = c(-10, 10), ylim = c(-10, 10))
abline(0, 1, col = "blue")
```



Is there a quick way to write a function for the composite likelihood, J and H? This would certainly allow us to explore explivit and implicit RBM more.

Needless to say that it is crucial that J and H in their unit information versions are scaled by the same quantity.

References

Jamil, Haziq, Irini Moustaki, and Chris Skinner. 2023. "Pairwise Likelihood Estimation and Limited Information Goodness-of-Fit Test Statistics for Binary Factor Analysis Models Under Complex Survey Sampling." *arXiv Preprint arXiv:2311.02543*.