Improving The Success Rate Of Optimization Algorithms In Psychometric Software

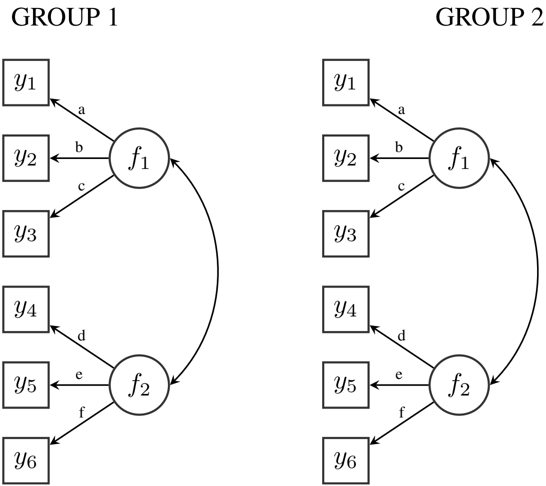
Yves Rosseel
Department of Data Analysis
Ghent University – Belgium

February 28, 2020
Psychoco – TU Dortmund University

optimization

- for many (psychometric) models, parameter estimation involves an iterative optimization algorithm
  - Newton-Raphson, Fisher scoring
  - **quasi-Newton** (eg., BFGS)
  - Expectation Maximization
  - ...
- in R, quasi-Newton optimization can be done with the functions `nlm()`, `optim()`, or **`nlminb()`**
- without care, optimization may fail (no solution is found)
- I will discuss three tricks that may help:
  1. handling linear equality constraints
  2. parameter scaling
  3. **parameter bounds**

linear equality constraints: example



- (weak) invariance model: equal factor loadings across groups

linear equality constraints in optimization

- consider the minimization of a nonlinear function subject to a set of linear equality constraints:

$$\min f(\mathbf{x}) \quad \text{subject to } \mathbf{A}\mathbf{x} = \mathbf{b}$$

- when the equality constraints are linear, you can use an 'elimination of variables' trick, ending up with an unconstrained optimization problem
- see section 15.3 of

  Nocedal, J. and Wright, S. (2006). *Numerical Optimization (2nd edition).* New York, NY: Springer

- the idea is to 'project' the full parameter vector ($\mathbf{x}$) to a reduced parameter vector ($\mathbf{x}^\star$), and send this reduced parameter vector to the optimizer
- every time we need to evaluate the objective function, we need to 'unpack' $\mathbf{x}^\star$ to form $\mathbf{x}$
- see `lav_model_estimate.R` in the lavaan package for example code

parameter scaling

- consider the standard (unconstrained) minimization problem

$$\min f(\mathbf{x})$$

  where $\mathbf{x} = \{x_1, x_2, \ldots, x_r, \ldots, x_R\}$

- in a 'well-scaled' optimization problem, the following rule holds:

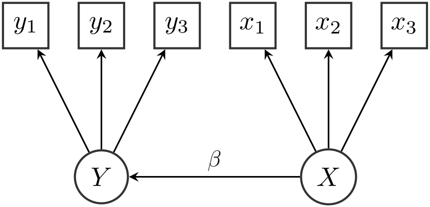
  "a one unit change in x_r results in a one unit change for $f(\mathbf{x})$"

- if this is not the case, you should rescale the model parameters until this 'rule' holds approximately
  - it may take some experimentation to find good scaling factors that work well in general (for your specific model)

- the `nlminb()` function has a `scale=` argument, where you provide a vector of scaling factors for each parameter

if the sample size is (very) small: parameter bounds may help

- consider the following SEM:



- this is a small model, with only 13 free parameters:
  - the factor loadings are set to 1, 0.8 and 0.6
  - the regression coefficient is set to $\beta = 0.25$
  - all (residual) variances are set to 1.0
- from this population model, we will generate a small sample ($N = 20$)

data generation ($N = 20$)

```
> library(lavaan)
> pop.model <- '
+     # factor loadings
+     Y =~ 1*y1 + 0.8*y2 + 0.6*y3
+     X =~ 1*x1 + 0.8*x2 + 0.6*x3
+
+     # regression part
+     Y ~ 0.25*X
+ '
> set.seed(8)
> Data <- simulateData(pop.model, sample.nobs = 20L)
```

fitting the model using ML

```
> model <- '
+     # factor loadings
+     Y =~ y1 + y2 + y3
+     X =~ x1 + x2 + x3
+
+     # regression part
+     Y ~ X
+ '
> fit.sem <- sem(model, data = Data, estimator = "ML")

lavaan WARNING: the optimizer warns that a solution has NOT been found!
```

output SEM

```
Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  Y =~
    y1                1.000
    y2                1.683       NA
    y3                1.051       NA
  X =~
    x1                1.000
    x2              302.417       NA
    x3                0.428       NA

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  Y ~
    X                -0.159       NA

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                1.706       NA
   .y2                0.763       NA
   .y3                1.066       NA
   .x1                1.408       NA
   .x2             -415.125       NA
   .x3                1.552       NA
   .Y                 0.450       NA
    X                 0.005       NA
```

R = 1000 replications: percentage of converged solutions

sample size	percentage converged
10	51.3%
15	63.0%
20	73.4%
25	78.6%
30	82.4%
40	91.7%
50	93.9%
60	97.1%
70	99.0%
80	99.1%
90	99.5%
100	99.7%

ML estimation + bounds

- given the data, we can determine ‘theoretical’ lower and upper bounds for the model parameters

- some notation:

  – s_p^2 is the observed sample variance of the p-th observed indicator
  – in scalar notation, we can write the (one-factor) measurement model as

$$y_p = \lambda_p \, f + \epsilon_p$$

  – we assume $\mathrm{Cov}(f, \epsilon_p) = 0$ and write $\mathrm{Var}(\epsilon_p) = \theta_p$ and $\mathrm{Var}(f) = \psi$, and therefore

$$\mathrm{Var}(y_p) = s_p^2 = \lambda_p^2 \, \psi + \theta_p$$

- we need bounds for the factor loadings (λ_p), the residual variances (θ_p), covariances and (optionally) regression coefficients

a few examples of lower/upper bounds

- we fix the metric of the factor f by fixing the first factor loading to 1
- the upper positive bound for λ_p is given by

$$\lambda_p^{(u)} = \sqrt{\frac{s_p^2}{\psi^{(l)}}}$$

  where $\psi^{(l)}$ is the lower bound for the variance of the factor

- the lower bound for the factor variance can be expressed as:

$$\psi^{(l)} = s_1^2 - [1 - \mathrm{REL}(y_1)]s_1^2$$

  where $\mathrm{REL}(y_1)$ is the (unknown) minimum reliability of the first (marker) indicator y_1

- we will often assume that $\mathrm{REL}(y_1) \geq 0.1$

a few examples of lower/upper bounds (2)

- residual variance θ_p
  - the lower bound for θ_p is zero
  - the upper bound for θ_p is s_p^2
  - more stricter bounds can be derived (see the EFA literature)
- a correlation (in absolute value) can not exceed 1.0; therefore

$$1 \geq \left| \frac{\text{Cov}(\theta_p, \theta_q)}{\sqrt{\text{Var}(\theta_p)}\sqrt{\text{Var}(\theta_q)}} \right|$$

$$\sqrt{\text{Var}(\theta_p)}\sqrt{\text{Var}(\theta_q)} \geq |\text{Cov}(\theta_p, \theta_q)|$$

- we will not impose bounds on the regression coefficient β

increasing/decreasing the bounds

- suppose the lower/upper bounds for a parameter θ are (0, 10)
- we can increase the upper bound with, say, 10%: (0, 11)
- similarly, we can decrease the lower bound with 10%: (-1,11)
- we have set up a simulation study to find 'optimal' bounds by using varying factors to increase/decrease the bounds (joint work with my PhD student Julie De Jonckere)
- currently, the 'best' choice seems to be:
  - minimum reliability first indicator: 0.1 (or higher)
  - increase/decrease bounds of observed variances with a factor 1.2
  - increase/decrease bounds of factor loadings with a factor 1.1
  - increase upper bounds of latent variances with a factor 1.3
- what happens to the percentage of converged solutions?

R = 1000 replications: percentage of converged solutions (with bounds)

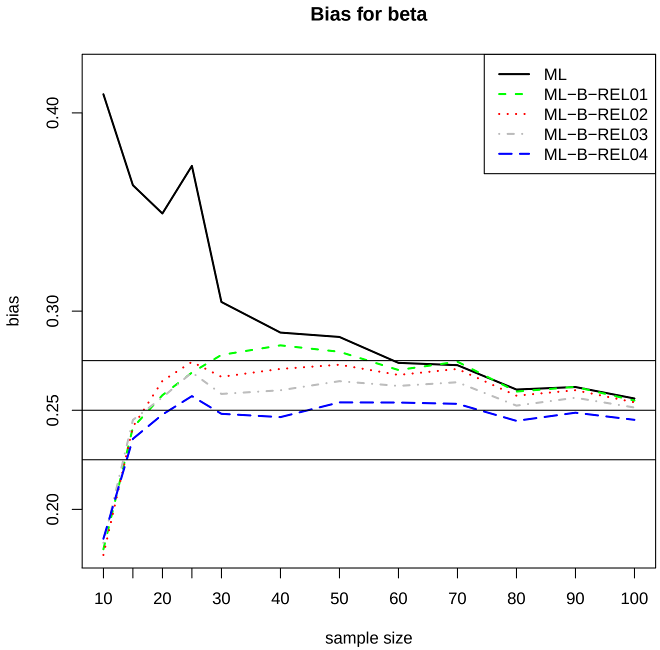
sample size	percentage converged
10	100%
15	100%
20	100%
25	100%
30	100%
40	100%
50	100%
60	100%
70	100%
80	100%
90	100%
100	100%

using these bounds with lavaan dev 0.6-6

```
> fit.semb <- sem(model, data = Data, estimator = "ML", bounds = TRUE)
> parTable(fit.semb)[,c(2,3,4,8,13,14,16)]

   lhs op rhs free  lower upper    est
1    Y =~  y1    0  1.000 1.000  1.000
2    Y =~  y2    1 -3.689 3.689  1.392
3    Y =~  y3    2 -3.231 3.231  0.977
4    X =~  x1    0  1.000 1.000  1.000
5    X =~  x2    3 -4.907 4.907  2.023
6    X =~  x3    4 -3.978 3.978  0.558
7    Y  ~   X    5   -Inf   Inf -0.104
8   y1 ~~  y1    6 -0.431 2.588  1.597
9   y2 ~~  y2    7 -0.407 2.445  0.953
10  y3 ~~  y3    8 -0.313 1.875  1.029
11  x1 ~~  x1    9 -0.283 1.695  0.715
12  x2 ~~  x2   10 -0.472 2.834 -0.472
13  x3 ~~  x3   11 -0.310 1.863  1.335
14   Y ~~   Y   12  0.000 2.803  0.552
15   X ~~   X   13  0.141 1.837  0.698
```

output SEM with bounds

```
Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  Y =~
    y1                1.000
    y2                1.392    1.013    1.374    0.170
    y3                0.977    0.652    1.498    0.134
  X =~
    x1                1.000
    x2                2.023    1.088    1.860    0.063
    x3                0.558    0.305    1.830    0.067

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  Y ~
    X                -0.104    0.226   -0.461    0.644

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                1.597    0.635    2.515    0.012
   .y2                0.953    0.795    1.198    0.231
   .y3                1.029    0.488    2.106    0.035
   .x1                0.715    0.408    1.750    0.080
   .x2          (lb) -0.472    1.401   -0.337
   .x3                1.335    0.435    3.072    0.002
   .Y                 0.552    0.590    0.935    0.350
    X                 0.698    0.514    1.358    0.175
```


Bias for beta
ML
ML−B−REL01
ML−B−REL02
ML−B−REL03
ML−B−REL04
bias
0.20
0.25
0.30
0.40
sample size
10
20
30
40
50
60
70
80
90
100

last slide

- we discussed three tricks to increase the success rate of optimization
  1. eliminating parameters (linear equality constraints)
  2. scaling parameters
  3. parameter bounds
- caveat: 1) and 3) do not mix!
- we are currently investigating the use of parameter bounds for multilevel SEMs when the number of clusters is rather small
- my examples were taken from the SEM domain, but the tricks apply to all (psychometric) models that require optimization

Thank you!

(questions?)

`http://lavaan.org`