lavaan: 構造方程式モデリングおよびその他のための R パッケージ バージョン 0.5-12（ベータ版）

Yves Rosseel
Department of Data Analysis
Ghent University (Belgium)
2012 年 12 月 19 日版

日本語訳　荒木　孝治*
岸谷　和広*
馬場　一*

2013 年 1 月 24 日

要約

本稿では，いくつかの例を用いて **lavaan**の利用の仕方を示す．**lavaan**をはじめて使おうという人は，これを最初に読んでほしい．[*1]

目次

* 関西大学　商学部．翻訳に関する意見等は arakit@kansai-u.ac.jp まで．

[*1] 訳注）原著 “lavaan: an R package for structural equation modeling and more” は，`http://users.ugent.be/~yrosseel/lavaan/lavaanIntroduction.pdf` よりダウンロード可能．

1 はじめに

lavaanの利用を開始する前に，次の諸点に注意してほしい．

- 第 1 に，R のできるだけ最新版（≥2.14.0）をインストールしておくべきである．R の最新版はサイト：`http://cran.r-project.org/` よりダウンロードできる．
- **lavaan**パッケージの開発は，まだ終わっていない．しかし，大部分のユーザーにとってはすでに，非常に役立つ状態になっている，あるいは，そうなっていると期待している．既知の重要でない問題があり（ウェブサイトでリストにしている），また，まだ実装していない機能もある．**lavaan**で現在利用できない重要な機能は，以下の通りである：
  - 階層的/マルチレベルデータセットのサポート（マルチレベル cfa，マルチレベル sem）
  - 離散潜在変数のサポート（混合モデル，潜在クラス）
  - ベイズ推定

  来年（さ来年？）くらいにはこれらの重要な機能を追加する予定である．
- 私たちは，現在のバージョンをベータ版と考えている．しかしこれは，結果が信頼できないということを意味するわけではない．結果は正確であると確信している．新しいバージョンが出たとき，状況が変わる可能性があることを意味している．例えば，関数の呼び出しにおける引数の名前を変えるかも知れない．また，ソースコードを絶えず変更する．しかし，モデルのシンタックスはほぼ安定しており，しばらく変更することはない．
- 読者には R の専門家を想定していない．実際，**lavaan**パッケージは，R を使ったことのないユーザーが利用できるように設計されている．しかし，R に少しでも慣れていると，利用しや

すくなるだろう．おそらく，学ぶべき最も重要なスキルは，自分自身のデータセット（おそらくSPSS フォーマットのもの）を R にインポートする方法である．この方法を学ぶためのチュートリアルがウェブにたくさんある．一旦データを R に読み込むと，モデルの作成を開始できる．私たちは，ユーザーが自分自身のモデルを当てはめることをできるだけ簡単にしようとしている．改善に関する提案があるときは，連絡してほしい．

- 本稿は，**lavaan**パッケージを初めて利用するユーザー（そして，ベータ版のテスター）のための文書である．参照マニュアルではないし，**lavaan**パッケージで機能がどのように実装されているかということを記す技術文書でもない．これらは現在準備中である．
- **lavaan**パッケージは，フリーのオープンソースソフトウェアである．これは，保証が全くないことを意味する．
- **lavaan**パッケージの計算結果は，商用パッケージ Mplus のものと同一でないとしても非常に近い．他の SEM パッケージの結果と比較したいときは，関数 cfa，sem，growth を利用するときに引数 `mimic="EQS"`を利用することができる（9.2 節参照）．
- （2012 年 9 月 12 日以降）手助けが必要な場合，**lavaan**ディスカッショングループで質問できる．その際、`https://groups.google.com/d/forum/lavaan/` で，グループに参加する．グループに参加した後，質問を `lavaan@googlegroups.com` に投稿することができる．バグを見つけたと思ったり，改良に関する提案があったりする場合，github で問題をオープンにすることができる（`https://github.com/yrosseel/lavaan/issues` 参照）．バグを報告するときは，問題を再現することができる例（R の簡単なスクリプトとデータ）を提供すべきである．
- 本稿は最新（2012 年 9 月 12 日）のものではない．ユーザーマニュアルの整備をウェブで行っている．‘**lavaan**に関する論文’（`http://www.jstatsoft.org/v48/i02/`）はより新しい（少なくともバージョン 0.4-14 までは）．

2 lavaanパッケージのインストール

2010 年 5 月以来，**lavaan**パッケージは CRAN で利用可能となっている．よって，**lavaan**をインストールするには，R を起動し，次を入力する．

```
> install.packages("lavaan")
```

インストールが成功したかどうかを確認するには，次を入力する．

```
> library(lavaan)

This is lavaan 0.5-12
lavaan is BETA software! Please report any bugs.
```

パッケージのロードがうまくいくと，バージョン番号や，まだベータバージョンであるという通知といったスタート時のメッセージが表示される．

3 モデル式の記法

lavaanパッケージの重要なものに，‘モデル構文’ がある．モデル構文は，推定すべきモデルの記述法である．本節では，**lavaan**のモデル構文の要素を簡単に説明する．より詳細な説明は，後に例で示す．R 環境では，回帰式は次の形で記す：

$$y \sim x1 + x2 + x3 + x4$$

このモデル式において，記号チルダ（∼）は回帰のオペレータである．このオペレータの左には目的変数（y）を記し，右には，独立変数を + オペレータで区切りながら記す．**lavaan**における典型的モデルは，回帰式の集合（あるいはシステム）である．これは，潜在変数を含んでもよい（以下の例では，潜在変数は 'f' で始まる）．例えば：

$$\begin{aligned} y &\sim f1 + f2 + x1 + x2 \\ f1 &\sim f2 + f3 \\ f2 &\sim f3 + x1 + x2 \end{aligned}$$

回帰式の中に潜在変数が含まれる場合，manifest 変数をリストすることによりそれらを定義する必要がある．このための特別なオペレータとして "=∼" があるが，これは，manifested されると読む．

例えば，3 つの潜在変数 f1，f2，f3 を定義するには，次のようにする：

$$\begin{aligned} f1 &=\sim y1 + y2 + y3 \\ f2 &=\sim y4 + y5 + y6 \\ f3 &=\sim y7 + y8 + y9 + y10 \end{aligned}$$

さらに，分散と共分散は，2 重のチルダ（∼∼）オペレータを用いて指定する．例えば：

$$\begin{aligned} y1 &\sim\sim y1 \\ y1 &\sim\sim y2 \\ f1 &\sim\sim f2 \end{aligned}$$

最後に，観測変数と潜在変数の切片は，回帰分析の記法において説明変数を切片のみとする（明示的に数字 '1' で示す）：

$$\begin{aligned} y1 &\sim 1 \\ f1 &\sim 1 \end{aligned}$$

これらの 4 つの式のタイプを利用して，多種多様な潜在変数モデルを記述することができる．しかし，新しい型を将来導入する可能性もある．現在の式のタイプを次表にまとめておく．

モデルのタイプ	オペレータ	意味
潜在変数の定義	=∼	∼により測定される
回帰	∼	∼への回帰
(残差) （共）分散	∼∼	∼と相関を持つ
切片	∼ 1	切片

3.1 文字列リテラルとしてのモデル式の指定

モデル式が短いときは，R プロンプトで，一重引用符で囲んでインタラクティブに入力すればよい．例えば：

```
> myModel <- '  # 回帰
                y1 + y2 ~ f1 + f2 + x1 + x2
                     f1 ~ f2 + f3
```

```
      f2 ~ f3 + x1 + x2
# 潜在変数の定義
 f1 =~ y1 + y2 + y3
 f2 =~ y4 + y5 + y6
 f3 =~ y7 + y8 +
       y9 + y10
# 分散と共分散
  y1 ~~ y1
  y1 ~~ y2
  f1 ~~ f2
# 切片
  y1 ~ 1
  f1 ~ 1
'
```

もちろん，インタラクティブに入力する代わりに，外部のテキスト・エディタでモデル全体を入力しておき，R コンソールにそれをコピー&ペーストしてもよい．コードのこの集合は，myModel という名前の典型的な構文オブジェクトを作る．これは後に，実際にデータセットが与えられたときに，モデルを推定する関数を呼ぶ際に利用される．モデル式をわかりやすくするために，式を複数行に分けてもよい．また，コメントを付けたり（#で始める），一重引用符の中に空行を入れたりしてもよい．

3.2 外部ファイルからのモデル式の読み込み

モデル構文がかなり長い，あるいは，何度もそれを再利用する必要がある場合，テキストファイルに入力しておく（ファイル名を仮に myModel.lav とする）方を好むかもしれない．このテキストファイルは，人間が読み取れるフォーマット（Word 文書ではない）でなければならない．R 内で，次のようにしてモデル構文を読み込むことができる：

```
> myModel <- readLines("/mydirectory/myModel.lav")
```

readLines 関数の引数は，モデル構文が保存されているファイルへのフルパスである．また，モデル構文のオブジェクトである myModel は，後で，与えられたデータセットにこのモデルを当てはめるときに利用することができる．

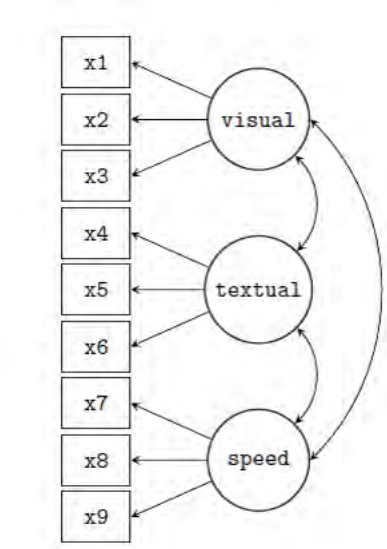
4 潜在変数モデルの当てはめ: 2 つの例

4.1 例 1: 検証的因子分析（CFA）

先ず，cfa 関数を用いて検証的因子分析（CFA）を行うという簡単な例を示す．関数 cfa は，CFA モデルを当てはめるためのユーザーフレンドリーな関数である．**lavaan**パッケージには，Holzinger-Swineford1939 という名前の組み込みデータセットがある．R プロンプト（>）の後ろに次を入力すると，このデータセットのヘルプを参照することができる．

```
> ?HolzingerSwineford1939
```

これは，構造方程式モデリング（SEM）に関する多くの論文や本（商用の SEM パッケージのマニュアルを含む）で利用されている ‘古典的な’ データセットである．データは，2 つの学校（Pasteur と Grant-White）の 7 年生と 8 年生の生徒の知能検査のスコアである．



lavaan syntax

```
visual =~ x1 + x2 + x3
textual =~ x4 + x5 + x6
speed =~ x7 + x8 + x9
```

このデータセットは，オリジナルのデータにある 26 個のうちの 9 つのテスト結果のみを抜き出したものである．9 つの変数に対して，各 3 つの指標から構成される 3 つの潜在変数（因子）を考える CFA モデルを適用することが多い．3 つの潜在因子を次に示す．

- 3 変数（$x1, x2, x3$）で計測される視覚（*visual*）因子
- 3 変数（$x4, x5, x6$）で計測される言語（*textual*）因子
- 3 変数（$x7, x8, x9$）で計測される速度（*speed*）因子

次に示す図の左のパネルは，3 因子モデルを簡単に示すグラフである．右のパネルは，本モデルを指定するための対応する **lavaan**の構文である．

このモデル構文は，3 つの ‘潜在変数の定義’ だけを含む．各式は，以下の形式になる：

$$\text{潜在変数} =\sim \text{指標 } 1 + \text{指標 } 2 + \text{指標 } 3 \tag{1}$$

このような表現を，潜在変数の定義という．なぜなら，観測（または顕在）変数（‘指標’ ということが多い）の集合によって潜在変数がどのように ‘明示される’ かを示しているからである．中央にある特殊な記号 ‘=∼’ は，割当記号（‘=’）とチルダ記号（‘∼’）から構成されている．このモデル構文がこんなに短い理由は，裏で cfa 関数が次のような状況の面倒を見ているからである．第 1 に，潜在変数の最初の指標の因子負荷量をデフォルトで 1 に固定している．これにより潜在変数のスケールが固定される．第 2 に，残差分散を自動的に加える．第 3 に，すべての外生的な潜在変数が相関を持つように，デフォルトで設定する．これらのおかげでモデル構文が簡潔になっている．一方，こうしたデフォルト値を無効にしたり，変更したりすることができるので，利用者が完全にコントロールすることも可能である．

このモデル構文を単一引用符（’）を用いて入力することができる．

```
> HS.model <- '
+  visual =~ x1 + x2 + x3
+  textual =~ x4 + x5 + x6
+  speed =~ x7 + x8 + x9
+  '
```

すると，次のようにして，CFA モデルにデータを当てはめることができる：

```
> fit <- cfa(HS.model, data = HolzingerSwineford1939)
```

lavaanの関数 cfa は，検証的因子分析モデルに当てはめるための専用の関数である．第 1 の引数は，ユーザーが指定するモデルであり，第 2 の引数は，観察された変数を含むデータセットである．モデルを当てはめた後，summary メソッドを用いて，当てはめたモデルの要約情報を表示することができる：

```
> summary(fit, fit.measures = TRUE)
lavaan (0.5-12) converged normally after 41 iterations

  Number of observations                           301

  Estimator                                         ML
  Minimum Function Test Statistic               85.306
  Degrees of freedom                                24
  P-value (Chi-square)                           0.000

Model test baseline model:

  Minimum Function Test Statistic              918.852
  Degrees of freedom                                36
  P-value                                        0.000

Full model versus baseline model:

  Comparative Fit Index (CFI)                    0.931
  Tucker-Lewis Index (TLI)                       0.896

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -3737.745
  Loglikelihood unrestricted model (H1)      -3695.092

  Number of free parameters                         21
  Akaike (AIC)                                7517.490
  Bayesian (BIC)                              7595.339
  Sample-size adjusted Bayesian (BIC)         7528.739

Root Mean Square Error of Approximation:

  RMSEA                                          0.092
  90 Percent Confidence Interval          0.071  0.114
  P-value RMSEA <= 0.05                          0.001

Standardized Root Mean Square Residual:

  SRMR                                           0.065

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
```

```
  visual =~
    x1                1.000
    x2                0.553    0.100    5.554    0.000
    x3                0.729    0.109    6.685    0.000
  textual =~
    x4                1.000
    x5                1.113    0.065   17.014    0.000
    x6                0.926    0.055   16.703    0.000
  speed =~
    x7                1.000
    x8                1.180    0.165    7.152    0.000
    x9                1.082    0.151    7.155    0.000

Covariances:
  visual ~~
    textual           0.408    0.074    5.552    0.000
    speed             0.262    0.056    4.660    0.000
  textual ~~
    speed             0.173    0.049    3.518    0.000

Variances:
    x1                0.549    0.114
    x2                1.134    0.102
    x3                0.844    0.091
    x4                0.371    0.048
    x5                0.446    0.058
    x6                0.356    0.043
    x7                0.799    0.081
    x8                0.488    0.074
    x9                0.566    0.071
    visual            0.809    0.145
    textual           0.979    0.112
    speed             0.384    0.086
```

出力は，他の SEM ソフトウェアのユーザーにとってはなじみのあるものである．これがわかりにくいとか美的に不愉快なら，知らせてほしい．改善することを試みる．この第 1 の例において，3 因子モデルを当てはめるのに必要なコード全体を次に記しておく：

R code

```
# lavaan パッケージのロード（セッション当たり 1 回のみ必要）
library(lavaan)

# モデルの設定
HS.model <- ' visual =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed =~ x7 + x8 + x9 '

# モデルへの当てはめ
fit <- cfa(HS.model, data=HolzingerSwineford1939)
```

```
# 要約情報の出力
summary(fit, fit.measures=TRUE)
```

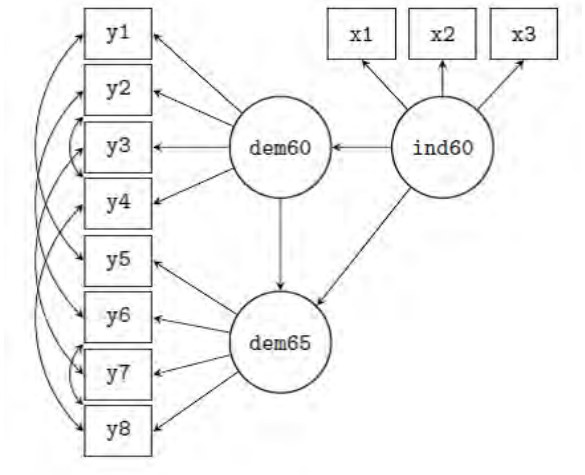
このコードは，R にコピー&ペーストするだけで実行される．この構文は，**lavaan**パッケージにおける典型的な仕事の流れを例示している：

1. **lavaan**のモデル構文を利用して，モデルを作成する．この例では，潜在変数のみが定義されている．次の例では，他のタイプの式を利用する．
2. モデルに当てはめる．これには，観測変数を含むデータセット（標本共分散行列とデータ数でもよい．9.1 節参照）が必要である．この例では，cfa 関数を使った．**lavaan**パッケージの他の関数として，構造方程式モデルと成長曲線モデルを当てはめるための sem と growth がある．これら 3 つの関数は，細々とした面倒なことを自動的に処理してくれという意味で，いわゆるユーザーフレンドリーな関数である．このおかげで，モデル構文を単純で簡潔にすることができる．標準的でないモデルを当てはめたり，自動的に処理されるのがいやだったりする場合，自分で全てをコントロールできる低レベル関数の lavaan を利用することができる．
3. 当てはめたモデルからの情報の抽出．結果の表示は，冗長な長いものにでもできるし，単一の数値（例えば，RMSEA の値）だけにでもできる．R の精神は，必要なものだけを抽出できればよいというものである．おそらく読者が無視すると思われる不要な情報は表示しない．

4.2 例 2：構造方程式モデリング

第 2 の例では，組み込みのデータセット PoliticalDemocracy を利用する．これは，構造方程式モデリングに関する Bollen 著の 1989 年の本以来利用されてきたデータセットである．このデータセットの詳細については，ヘルプページを参照．

下の図の左のパネルは，当てはめたいモデルの図である．右のパネルは，対応するモデル式である．



lavaan syntax

```
# 潜在変数の定義
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
# 回帰
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
# 残差共分散
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
```

この例では，3 つの異なる式のタイプを使う．すなわち，潜在変数の定義式，回帰式，（共）分散の定義式である．回帰式は，R の普通の式と似ている．（共）分散の定義は，次の形で行う．

変数 ~~ 変数

変数は，観測変数と潜在変数に分類することができる．左右の 2 つの変数名が同じとき，この表現は，変数の分散（または残差分散）を意味する．2 つの変数名が異なるときは，この表現は 2 変数間の（残差）共分散を意味する．**lavaan**パッケージは，自動的に分散と残差分散を区別する．

この例において，表現 `y1 ~~ y5` は，2 つの観測変数の残差分散が相関することを許容する．これは，2 変数が共通する潜在変数によって説明することができない何か別の共通の影響を受けていると考えるときに利用される．今の場合，2 つの変数は同じスコアであるが，2 つの異なる年（1960 年と 1965 年）に計測されたものである．2 つの表現 `y2 ~~ y4` と `y2 ~~ y6` を結合して，表現 `y2 ~~ y4 + y6` としていることに注意．このような略記が可能である．モデル構文を次のように入力する．

```
> model <- '
+  # measurement model
+  ind60 =~ x1 + x2 + x3
+  dem60 =~ y1 + y2 + y3 + y4
+  dem65 =~ y5 + y6 + y7 + y8
+
+ # regressions
+  dem60 ~ ind60
+  dem65 ~ ind60 + dem60
+
+ # residual correlations
+  y1 ~~ y5
+  y2 ~~ y4 + y6
+  y3 ~~ y7
+  y4 ~~ y8
+  y6 ~~ y8
+  '
```

このモデルにデータを当てはめ，結果を見るには次を入力する：

```
> fit <- sem(model, data = PoliticalDemocracy)
> summary(fit, standardized = TRUE)
lavaan (0.5-12) converged normally after 70 iterations

  Number of observations                            75

  Estimator                                         ML
  Minimum Function Test Statistic               38.125
  Degrees of freedom                                35
  P-value (Chi-square)                           0.329

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)   Std.lv  Std.all
Latent variables:
  ind60 =~
    x1                1.000                               0.670    0.920
    x2                2.180    0.139   15.742    0.000    1.460    0.973
    x3                1.819    0.152   11.967    0.000    1.218    0.872
  dem60 =~
```

```
    y1                1.000                                    2.223    0.850
    y2                1.257    0.182    6.889    0.000    2.794    0.717
    y3                1.058    0.151    6.987    0.000    2.351    0.722
    y4                1.265    0.145    8.722    0.000    2.812    0.846
  dem65 =~
    y5                1.000                                    2.103    0.808
    y6                1.186    0.169    7.024    0.000    2.493    0.746
    y7                1.280    0.160    8.002    0.000    2.691    0.824
    y8                1.266    0.158    8.007    0.000    2.662    0.828

Regressions:
  dem60 ~
    ind60             1.483    0.399    3.715    0.000    0.447    0.447
  dem65 ~
    ind60             0.572    0.221    2.586    0.010    0.182    0.182
    dem60             0.837    0.098    8.514    0.000    0.885    0.885

Covariances:
  y1 ~~
    y5                0.624    0.358    1.741    0.082    0.624    0.296
  y2 ~~
    y4                1.313    0.702    1.871    0.061    1.313    0.273
    y6                2.153    0.734    2.934    0.003    2.153    0.356
  y3 ~~
    y7                0.795    0.608    1.308    0.191    0.795    0.191
  y4 ~~
    y8                0.348    0.442    0.787    0.431    0.348    0.109
  y6 ~~
    y8                1.356    0.568    2.386    0.017    1.356    0.338

Variances:
    x1                0.082    0.019                      0.082    0.154
    x2                0.120    0.070                      0.120    0.053
    x3                0.467    0.090                      0.467    0.239
    y1                1.891    0.444                      1.891    0.277
    y2                7.373    1.374                      7.373    0.486
    y3                5.067    0.952                      5.067    0.478
    y4                3.148    0.739                      3.148    0.285
    y5                2.351    0.480                      2.351    0.347
    y6                4.954    0.914                      4.954    0.443
    y7                3.431    0.713                      3.431    0.322
    y8                3.254    0.695                      3.254    0.315
    ind60             0.448    0.087                      1.000    1.000
    dem60             3.956    0.921                      0.800    0.800
    dem65             0.172    0.215                      0.039    0.039
```

関数 sem は，関数 cfa と非常に似ている．実際，2 つの関数の機能は現在ほとんど同じであるが，将来変わる可能性はある．summary メソッドにおいて，引数 `fit.measures=TRUE` を省略した．そのため，基本的なカイ 2 乗統計量のみが表示される．引数 `standardized=TRUE` は，標準化されたパラメータ値の出力を追加する．そのため，標準化されたパラメータ値の 2 列が追加されている．第 1 列（列名

Std.lv）では，潜在変数だけが標準化されている．第 2 列（列名 Std.all）では，潜在変数と観測変数が標準化されるている．後者は，‘完全に標準化された解’ と呼ばれることがある．

本モデルに当てはめるためのコード全体を再度，次に示す：

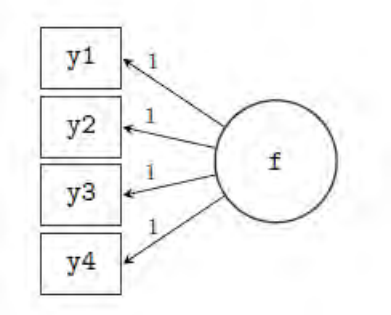
R code

```
library(lavaan) # セッション当たり 1 回のみ必要
model <- '
  # 測定モデル
    ind60 =~ x1 + x2 + x3
    dem60 =~ y1 + y2 + y3 + y4
    dem65 =~ y5 + y6 + y7 + y8
  # 回帰モデル
    dem60 ~ ind60
    dem65 ~ ind60 + dem60
  # 残差の相関
    y1 ~~ y5
    y2 ~~ y4 + y6
    y3 ~~ y7
    y4 ~~ y8
    y6 ~~ y8
'

fit <- sem(model, data=PoliticalDemocracy)
summary(fit, standardized=TRUE)
```

5　パラメータ，初期値，等式制約の指定

5.1　パラメータの指定

4 つの指標に対する 1 因子モデルを考える．デフォルトで，**lavaan**は第 1 指標の因子負荷量を常に 1 に固定する．その他の 3 つの指標の因子負荷量は自由パラメータであり，モデルによって推定される．しかし，すべての因子負荷量を 1 に設定したい場合もある．この方法を，次に示す構文で例示する．

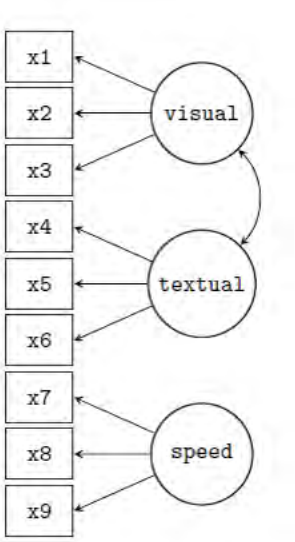


lavaan syntax

```
f =~ y1 + 1*y2 + 1*y3 + 1*y4
```

一般に，**lavaan**のモデル式でパラメータを指定するには，指定したい数値を対応する変数の左から

掛ける．これは前方乗算（pre-multiplication）方式といい，多くの目的で用いられている．別の例として，再度，Holzinger and Swineford の 3 因子 CFA モデルを取り上げる．デフォルトでは，CFA モデルの全ての外生的潜在変数が相関していることに注意．しかし，潜在変数のペアの相関（または共分散）を 0 に固定したいとき，このペアの共分散の式をモデル式に記入し，パラメータを 0 に指定する．下図では，潜在変数間の共分散を図で，式で自由パラメータとしているが，他の潜在変数間の共分散の存在は可能としている．そのうえ，speed（速度因子）の分散を 1 に固定している．したがって，第 1 指標（$x7$）の因子負荷量を 1 に指定する必要はない．この因子負荷量を自由パラメータにするには，それに左から NA を掛けておく．これにより，このパラメータの値は未知として取り扱かわれる．



lavaan syntax

```
# three-factor model
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ NA*x7 + x8 + x9
# orthogonal factors
   visual ~~ 0*speed
  textual ~~ 0*speed
# fix variance of speed factor
    speed ~~ 1*speed
```

CFA モデル内の潜在変数のすべての共分散が直交するように制約したいときには，簡単な方法がある．モデル構文で共分散の式を省略し，関数 cfa の引数として `orthogonal=TRUE` のみを与えるだけでよい．

```
> HS.model <- ' visual =~ x1 + x2 + x3
+              textual =~ x4 + x5 + x6
+              speed =~ x7 + x8 + x9 '
> fit.HS.ortho <- cfa(HS.model, data=HolzingerSwineford1939, orthogonal=TRUE)
```

同様に，CFA モデルの全ての潜在変数の分散を 1 に固定したいときにも，簡単な方法がある．関数 cfa の引数として，`std.lv=TRUE` を与えるのである．

```
> HS.model <- ' visual =~ x1 + x2 + x3
+              textual =~ x4 + x5 + x6
+              speed =~ x7 + x8 + x9 '
> fit <- cfa(HS.model, data=HolzingerSwineford1939, std.lv=TRUE)
```

引数 `std.lv=TRUE` が与えられると，各潜在変数の第 1 指標の因子負荷量は 1 に固定されない．

5.2 初期値

lavaanパッケージは，すべての自由パラメータの初期値を自動的に準備する．通常，これでうまくいく．しかし，独自の初期値を与えたいときもある．その方法は，前に利用した前方乗算方式に基づくが，数値定数は特別な関数である start() の引数として与える．次の例を見ると，この方法が簡単にわ

かる．

lavaan syntax

```
visual =~ x1 + start(0.8)*x2 + start(1.2)*x3
textual =~ x4 + start(0.5)*x5 + start(1.0)*x6
speed =~ x7 + start(0.7)*x8 + start(1.8)*x9
```

第 1 指標（$x1$，$x4$，$x7$）の因子負荷量は固定されるので，初期値は必要ない．他の全ての因子負荷量に対して，この例では初期値が与えられている．

5.3 パラメータの名前

lavaanパッケージの素晴らしい特性として，全ての自由パラメータの名前が，簡単なルールに従って自動的に与えられることがある．たとえば，等式制約が必要なとき（次項参照），これは便利である．命名のメカニズムがどのように動くかについて見るために，PolitcalDemocracy データに使ったモデルを利用する．

```
> model <- '
+   # 潜在変数の定義
+     ind60 =~ x1 + x2 + x3
+     dem60 =~ y1 + y2 + y3 + y4
+     dem65 =~ y5 + y6 + y7 + y8
+   # regressions
+     dem60 ~ ind60
+     dem65 ~ ind60 + dem60
+   # residual (co)variances
+     y1 ~~ y5
+     y2 ~~ y4 + y6
+     y3 ~~ y7
+     y4 ~~ y8
+     y6 ~~ y8
+ '
> fit <- sem(model, data=PoliticalDemocracy)
> coef(fit)
   ind60=~x2    ind60=~x3    dem60=~y2    dem60=~y3    dem60=~y4    dem65=~y6
       2.180        1.819        1.257        1.058        1.265        1.186
   dem65=~y7    dem65=~y8 dem60~ind60 dem65~ind60 dem65~dem60       y1~~y5
       1.280        1.266        1.483        0.572        0.837        0.624
      y2~~y4       y2~~y6       y3~~y7       y4~~y8       y6~~y8       x1~~x1
       1.313        2.153        0.795        0.348        1.356        0.082
      x2~~x2       x3~~x3       y1~~y1       y2~~y2       y3~~y3       y4~~y4
       0.120        0.467        1.891        7.373        5.067        3.148
      y5~~y5       y6~~y6       y7~~y7       y8~~y8 ind60~~ind60 dem60~~dem60
       2.351        4.954        3.431        3.254        0.448        3.956
dem65~~dem65
       0.172
```

coef 関数は，モデル内の自由パラメータの推定値をその名前とともに抽出する．名前は 3 つのパートから構成されており，パラメータが含まれている式の部分を反映する．最初のパートは，式の左側にあらわれる変数名である．中間パートは式のオペレータのタイプであり，第 3 のパートは，パラメータと

対応する式の右側の変数である．必要なら，カスタムメイドのパラメータ名またはラベルを変数名に対して前方乗算方式で提供することができる．この方法は，次の例を見れば明白である．

```
> model <- '
+   # 潜在変数の定義
+     ind60 =~ x1 + x2 + myLabel*x3
+     dem60 =~ y1 + y2 + y3 + y4
+     dem65 =~ y5 + y6 + y7 + y8
+   # 回帰
+     dem60 ~ ind60
+     dem65 ~ ind60 + dem60
+   # 残差（共）分散
+     y1 ~~ y5
+     y2 ~~ y4 + y6
+     y3 ~~ y7
+     y4 ~~ y8
+     y6 ~~ y8
+ '
```

ラベルは記号 a-zA-Z で始まり，数字は利用できないことに注意．例えば，'13bis' は有効なラベルではないので，lavaan の構文パーサーを混乱させる．（注意）バージョン 0.4-8 以前では，ラベルをカスタマイズするには label() を利用する必要があった．これは現在もサポートされているが，利用を推奨しない．新しいシンタックスでこれを利用すべき状況は，ラベルが" "や=を含むときのみである．

5.4 簡単な等式制約

目的によっては，等式制約を複数の自由パラメータに課したいことがある．再度，3-因子 H&S CFA モデルを考える．ある先験的な理由から，指標 $x2$ と $x3$ の因子負荷量を等しくしたいとする．このとき，2 つの自由パラメータを推定する代わりに，**lavaan**は一つの自由パラメータを推定し，両方の因子負荷量にその値を使うだけでよい．このようなタイプの（簡単な）等式制約の主要なメカニズムは，ラベルを用いることである．2 つのパラメータが同じラベルを持つと，同じだと判断し，それらのために 1 つの値のみを計算する．これらを以下の構文で例示する．

lavaan syntax

```
visual =~ x1 + v2*x2 + v2*x3
textual =~ x4 + x5 + x6
speed =~ x7 + x8 + x9
```

同じラベルを持つ全てのパラメータは，等しく制約される．他の方法として，equal() を利用することもできる．これは，カスタムラベルがなく，自動的につけられたラベルを指定するときに便利である．例えば：

lavaan syntax

```
visual =~ x1 + x2 + equal("visual=~x2")*x3
textual =~ x4 + x5 + x6
speed =~ x7 + x8 + x9
```

変数 $x2$ の因子負荷量に対応するパラメータは自動的に `visual=~x2` が呼ばれる．$x3$ に対して equal() を用いることにより，対応するパラメータ値は $x2$ の因子負荷量に等しく設定される．

5.5 非線形の等式・不等式制約

バージョン 0.4-8 において，一般的な非線形の等式・不等式制約の機能がはじめて（実験的に）追加された．例えば，次の回帰の例を考える．

lavaan syntax

```
y ~ b1*x1 + b2*x2 + b3*x3
```

次の例では，回帰係数を $b1$，$b2$，$b3$ と明示的に与えている．これらの 4 つの変数を含む簡単なデータセットを作り，回帰モデルを当てはめる．

```
> set.seed(1234)
> Data <- data.frame(y = rnorm(100), x1 = rnorm(100), x2 = rnorm(100),
+                    x3 = rnorm(100))
> model <- ' y ~ b1*x1 + b2*x2 + b3*x3 '
> fit <- sem(model, data=Data)
> coef(fit)
    b1     b2     b3   y~~y
-0.052  0.084  0.139  0.970
```

この問題に，$b1 = (b2 + b3)^2$ と $b1 > \exp(b2 + b3)$ という 2 つの（非線形の）制約を課したいとする．最初は等式の制約であり，2 番目は不等式の制約である．これらの制約を指定するために，次のシンタックスを使う：

lavaan syntax

```
model.constr <- ' # ラベル付けられたパラメータを持つモデル
                    y ~ b1*x1 + b2*x2 + b3*x3
                  # 制約
                    b1 == (b2 + b3)^2
                    b1 > exp(b2 + b3) '
```

制約の効果を見るため，再度モデルに当てはめる．

```
> model.constr <- ' # ラベル付けられたパラメータを持つモデル
+                   y ~ b1*x1 + b2*x2 + b3*x3
+                 # 制約
+                   b1 == (b2 + b3)^2
+                   b1 > exp(b2 + b3) '
> fit <- sem(model.constr, data=Data)
> coef(fit)
    b1     b2     b3   y~~y
 0.495 -0.405 -0.299  1.610
```

出力結果より，制約が成り立っているかどうかを確認することができる．等式制約は正確に成り立っている．不等式制約に関しては，左辺の値（$b1$）と右辺の値（$\exp(b2 + b3)$ が等しくなる形で成り立っている．

6 平均構造と多母集団

6.1 平均の導入

構造方程式モデルは概して，データセット内の観測変数の共分散行列をモデル化するのに用いられる．しかし，目的によっては，観測変数の平均をモデルに導入することも有益である．これに対する 1 つの方法は，**lavaan**構文に切片項を明記することである．これは，モデル構文に '切片式' を組み込むことにより実行できる．切片式は，以下の形を持つ：

$$\text{変数} \sim 1 \tag{2}$$

この式の左辺は，観察変数または潜在変数の名前である．右辺には 1 が記載されているが，これが切片をあらわす．例えば，H&S3 因子 CFA モデルにおいて，観測変数の切片を次のようにして加えることができる：

lavaan syntax

```
# 3 因子モデル
  visual =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed =~ x7 + x8 + x9
# 切片
  x1 ~ 1
  x2 ~ 1
  x3 ~ 1
  x4 ~ 1
  x5 ~ 1
  x6 ~ 1
  x7 ~ 1
  x8 ~ 1
  x9 ~ 1
```

しかし，モデル構文で切片式を省略する方がより便利であり（その値を固定したくない限り），これには関数 cfa と sem で `meanstructure=TRUE` という引数を与える．例えば，H&S3 因子 CFA モデルに再度当てはめるには次のようにする：

```
> fit <- cfa(HS.model, data = HolzingerSwineford1939, meanstructure = TRUE)
> summary(fit)
lavaan (0.5-12) converged normally after 41 iterations

  Number of observations                           301

  Estimator                                         ML
  Minimum Function Test Statistic               85.306
  Degrees of freedom                                24
  P-value (Chi-square)                           0.000

Parameter estimates:
```

```
  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  visual =~
    x1                1.000
    x2                0.553    0.100    5.554    0.000
    x3                0.729    0.109    6.685    0.000
  textual =~
    x4                1.000
    x5                1.113    0.065   17.014    0.000
    x6                0.926    0.055   16.703    0.000
  speed =~
    x7                1.000
    x8                1.180    0.165    7.152    0.000
    x9                1.082    0.151    7.155    0.000

Covariances:
  visual ~~
    textual           0.408    0.074    5.552    0.000
    speed             0.262    0.056    4.660    0.000
  textual ~~
    speed             0.173    0.049    3.518    0.000

Intercepts:
    x1                4.936    0.067   73.473    0.000
    x2                6.088    0.068   89.855    0.000
    x3                2.250    0.065   34.579    0.000
    x4                3.061    0.067   45.694    0.000
    x5                4.341    0.074   58.452    0.000
    x6                2.186    0.063   34.667    0.000
    x7                4.186    0.063   66.766    0.000
    x8                5.527    0.058   94.854    0.000
    x9                5.374    0.058   92.546    0.000
    visual            0.000
    textual           0.000
    speed             0.000

Variances:
    x1                0.549    0.114
    x2                1.134    0.102
    x3                0.844    0.091
    x4                0.371    0.048
    x5                0.446    0.058
    x6                0.356    0.043
    x7                0.799    0.081
    x8                0.488    0.074
    x9                0.566    0.071
    visual            0.809    0.145
    textual           0.979    0.112
```

```
    speed             0.384    0.086
```

出力からわかるように，モデルは観測変数と潜在変数の両方に切片パラメータ（Intercept）を含んでいる．デフォルトでは，関数 cfa と sem は，潜在変数の切片（今の場合，潜在変数の平均に相当する）を 0 に設定する．こうしないとモデルは推定可能ではない．オリジナルの（非平均構造）モデルと，カイ 2 乗統計量と自由度が同じであることに注意．なぜなら，新しいデータ（9 つの観測変数の各平均値）を追加したが，新たに 9 つのパラメータ（9 つの観測変数の各切片）もモデルに追加されたからである．だから，同一の当てはめ結果となる．実際には，ユーザーがモデル構文で切片式を加える唯一の理由は，なんらかの制約をそれらに課したい場合である．例えば，変数 $x1$，$x2$，$x3$，$x4$ の切片を 0.5 に固定したいとする．モデル構文は次のようになる：

lavaan syntax

```
# 3 因子モデル
  visual =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed =~ x7 + x8 + x9
# 固定値を持つ切片
  x1 + x2 + x3 + x4 ~ 0.5*1
```

切片の式においては，左辺の各要素に対して右辺を ‘繰り返して’ 適用するスタイル（つまり，`x1 ~ 0.5*1`，`x2 ~ 0.5*1` 等）となっている．

6.2 多母集団

lavaanパッケージでは，多母集団（グループ）の解析が可能である．この解析を行うには，関数 cfa と sem の呼び出しにおいて，データセット内のグループ変数の名前を引数 `group` に与える必要がある．デフォルトでは，全ての母集団で同じモデルが当てはめられる．次の例では，2 つの学校（Pasteur と Grant-White）のデータセットに対して H&S CFA モデルを当てはめる．

```
> HS.model <- ' visual =~ x1 + x2 + x3
+              textual =~ x4 + x5 + x6
+              speed =~ x7 + x8 + x9 '
> fit <- cfa(HS.model, data=HolzingerSwineford1939, group="school")
> summary(fit)
lavaan (0.5-12) converged normally after 63 iterations

  Number of observations per group
  Pasteur                                       156
  Grant-White                                   145

  Estimator                                      ML
  Minimum Function Test Statistic           115.851
  Degrees of freedom                             48
  P-value (Chi-square)                        0.000

Chi-square for each group:

  Pasteur                                    64.309
```

```
  Grant-White                                   51.542

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

Group 1 [Pasteur]:

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  visual =~
    x1                1.000
    x2                0.394    0.122    3.220    0.001
    x3                0.570    0.140    4.076    0.000
  textual =~
    x4                1.000
    x5                1.183    0.102   11.613    0.000
    x6                0.875    0.077   11.421    0.000
  speed =~
    x7                1.000
    x8                1.125    0.277    4.057    0.000
    x9                0.922    0.225    4.104    0.000

Covariances:
  visual ~~
    textual           0.479    0.106    4.531    0.000
    speed             0.185    0.077    2.397    0.017
  textual ~~
    speed             0.182    0.069    2.628    0.009

Intercepts:
    x1                4.941    0.095   52.249    0.000
    x2                5.984    0.098   60.949    0.000
    x3                2.487    0.093   26.778    0.000
    x4                2.823    0.092   30.689    0.000
    x5                3.995    0.105   38.183    0.000
    x6                1.922    0.079   24.321    0.000
    x7                4.432    0.087   51.181    0.000
    x8                5.563    0.078   71.214    0.000
    x9                5.418    0.079   68.440    0.000
    visual            0.000
    textual           0.000
    speed             0.000

Variances:
    x1                0.298    0.232
    x2                1.334    0.158
    x3                0.989    0.136
    x4                0.425    0.069
    x5                0.456    0.086
    x6                0.290    0.050
```

```
    x7                0.820    0.125
    x8                0.510    0.116
    x9                0.680    0.104
    visual            1.097    0.276
    textual           0.894    0.150
    speed             0.350    0.126


Group 2 [Grant-White]:

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  visual =~
    x1                1.000
    x2                0.736    0.155    4.760    0.000
    x3                0.925    0.166    5.583    0.000
  textual =~
    x4                1.000
    x5                0.990    0.087   11.418    0.000
    x6                0.963    0.085   11.377    0.000
  speed =~
    x7                1.000
    x8                1.226    0.187    6.569    0.000
    x9                1.058    0.165    6.429    0.000

Covariances:
  visual ~~
    textual           0.408    0.098    4.153    0.000
    speed             0.276    0.076    3.639    0.000
  textual ~~
    speed             0.222    0.073    3.022    0.003

Intercepts:
    x1                4.930    0.095   51.696    0.000
    x2                6.200    0.092   67.416    0.000
    x3                1.996    0.086   23.195    0.000
    x4                3.317    0.093   35.625    0.000
    x5                4.712    0.096   48.986    0.000
    x6                2.469    0.094   26.277    0.000
    x7                3.921    0.086   45.819    0.000
    x8                5.488    0.087   63.174    0.000
    x9                5.327    0.085   62.571    0.000
    visual            0.000
    textual           0.000
    speed             0.000

Variances:
    x1                0.715    0.126
    x2                0.899    0.123
    x3                0.557    0.103
    x4                0.315    0.065
```

```
    x5                0.419    0.072
    x6                0.406    0.069
    x7                0.600    0.091
    x8                0.401    0.094
    x9                0.535    0.089
    visual            0.604    0.160
    textual           0.942    0.152
    speed             0.461    0.118
```

パラメータを固定したり初期値を与えたりしたいとき，前方乗算方式を利用することができるが，一つの数の引数ではなく，各母集団に対応するベクトルで与える必要がある．ベクトルではなく 1 つの数を与えると，その数がすべての母集団に適用される（注意：これはラベルには当てはまらない．というのは，等式制約となるからである）．たとえば：

lavaan syntax

```
HS.model <- ' visual =~ x1 + 0.5*x2 + c(0.6, 0.8)*x3
              textual =~ x4 + start(c(1.2, 0.6))*x5 + a*x6
              speed =~ x7 + x8 + x9 '
```

潜在因子の visual の定義において，第 1 母集団の x3 指標の因子負荷量は ‘0.6’ に，第 2 母集団のそれは ‘0.8’ に固定されているが，両母集団とも，x2 の因子負荷量は ‘0.5’ に固定されている．textual 因子の定義においては，x5 指標に対しては，2 つの異なる初期値が与えられている．さらに，$x6$ 指標の因子負荷量に ‘a’ というラベルをつけているが，このラベルは．第 1 母集団のパラメータのみに与えられる．2 つの母集団の各々に対してラベルづけしたいなら，`c(a1,a2)*x6` のように書く．注意：`c(a,a)*x6` とすると，どちらのパラメータも同じラベルを持つことになるので，1 つのパラメータとして取り扱われる．こうした修正の効果を知るために，もう一度モデルに当てはめる。

```
> fit <- cfa(HS.model, data = HolzingerSwineford1939, group = "school")
> summary(fit)
lavaan (0.5-12) converged normally after 58 iterations

  Number of observations per group
  Pasteur                                        156
  Grant-White                                    145

  Estimator                                       ML
  Minimum Function Chi-square                118.976
  Degrees of freedom                              52
  P-value (Chi-square)                         0.000

Chi-square for each group:

  Pasteur                                     64.309
  Grant-White                                 51.542

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard
```

```
Group 1 [Pasteur]:

                 Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  visual =~
    x1              1.000
    x2              0.394    0.122    3.220    0.001
    x3              0.570    0.140    4.076    0.000
  textual =~
    x4              1.000
    x5              1.183    0.102   11.613    0.000
    x6              0.875    0.077   11.421    0.000
  speed =~
    x7              1.000
    x8              1.125    0.277    4.057    0.000
    x9              0.922    0.225    4.104    0.000

Covariances:
  visual ~~
    textual         0.479    0.106    4.531    0.000
    speed           0.185    0.077    2.397    0.017
  textual ~~
    speed           0.182    0.069    2.628    0.009

Intercepts:
    x1              4.941    0.095   52.249    0.000
    x2              5.984    0.098   60.949    0.000
    x3              2.487    0.093   26.778    0.000
    x4              2.823    0.092   30.689    0.000
    x5              3.995    0.105   38.183    0.000
    x6              1.922    0.079   24.321    0.000
    x7              4.432    0.087   51.181    0.000
    x8              5.563    0.078   71.214    0.000
    x9              5.418    0.079   68.440    0.000
    visual          0.000
    textual         0.000
    speed           0.000

Variances:
    x1              0.298    0.232
    x2              1.334    0.158
    x3              0.989    0.136
    x4              0.425    0.069
    x5              0.456    0.086
    x6              0.290    0.050
    x7              0.820    0.125
    x8              0.510    0.116
    x9              0.680    0.104
    visual          1.097    0.276
    textual         0.894    0.150
    speed           0.350    0.126
```

Group 2 [Grant-White]:

	Estimate	Std.err	Z-value	P(>|z|)
Latent variables:				
visual =~				
x1	1.000			
x2	0.736	0.155	4.760	0.000
x3	0.925	0.166	5.583	0.000
textual =~				
x4	1.000			
x5	0.990	0.087	11.418	0.000
x6	0.963	0.085	11.377	0.000
speed =~				
x7	1.000			
x8	1.226	0.187	6.569	0.000
x9	1.058	0.165	6.429	0.000
Covariances:				
visual ~~				
textual	0.408	0.098	4.153	0.000
speed	0.276	0.076	3.639	0.000
textual ~~				
speed	0.222	0.073	3.022	0.003
Intercepts:				
x1	4.930	0.095	51.696	0.000
x2	6.200	0.092	67.416	0.000
x3	1.996	0.086	23.195	0.000
x4	3.317	0.093	35.625	0.000
x5	4.712	0.096	48.986	0.000
x6	2.469	0.094	26.277	0.000
x7	3.921	0.086	45.819	0.000
x8	5.488	0.087	63.174	0.000
x9	5.327	0.085	62.571	0.000
visual	0.000			
textual	0.000			
speed	0.000			
Variances:				
x1	0.715	0.126		
x2	0.899	0.123		
x3	0.557	0.103		
x4	0.315	0.065		
x5	0.419	0.072		
x6	0.406	0.069		
x7	0.600	0.091		
x8	0.401	0.094		
x9	0.535	0.089		
visual	0.604	0.160		
textual	0.942	0.152		

```
    speed             0.461    0.118
```

6.2.1 母集団全体で 1 つのパラメータが等しいと制約する

母集団全体で、1 つまたは複数のパラメータが等しいとしたいときは，それらに同じラベルを与える．例えば，(2 つの）母集団間で $x3$ 指標の因子負荷量を等しいとするとき，次のようにする．

```
> HS.model <- ' visual =~ x1 + x2 + c(v3,v3)*x3
+              textual =~ x4 + x5 + x6
+              speed =~ x7 + x8 + x9 '
```

6.2.2 母集団全体でパラメータのグループが等しいと制約する

いくつかのパラメータに等しい制約を課すために同一のラベルを与えることは非常に柔軟な方法であるが，パラメータの集合全体（例えば，すべての因子負荷量，またはすべての切片）が等しいという制約を課すためのより便利な方法がある．このような制約をグループ等号制約といい，関数 cfa または sem の呼び出しで，group.equal 引数を用いる．例えば，母集団全体で全ての因子負荷量を等しくするには，次のようにする：

```
> HS.model <- ' visual =~ x1 + x2 + x3
+              textual =~ x4 + x5 + x6
+              speed =~ x7 + x8 + x9 '
> fit <- cfa(HS.model, data=HolzingerSwineford1939, group="school",
+            group.equal=c("loadings"))
> summary(fit)
lavaan (0.5-12) converged normally after 46 iterations

  Number of observations per group
  Pasteur                                         156
  Grant-White                                     145

  Estimator                                        ML
  Minimum Function Test Statistic             124.044
  Degrees of freedom                               54
  P-value (Chi-square)                          0.000

Chi-square for each group:

  Pasteur                                      68.825
  Grant-White                                  55.219

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

Group 1 [Pasteur]:

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  visual =~
```

```
    x1                1.000
    x2                0.599    0.100    5.979    0.000
    x3                0.784    0.108    7.267    0.000
  textual =~
    x4                1.000
    x5                1.083    0.067   16.049    0.000
    x6                0.912    0.058   15.785    0.000
  speed =~
    x7                1.000
    x8                1.201    0.155    7.738    0.000
    x9                1.038    0.136    7.629    0.000

Covariances:
  visual ~~
    textual           0.416    0.097    4.271    0.000
    speed             0.169    0.064    2.643    0.008
  textual ~~
    speed             0.176    0.061    2.882    0.004

Intercepts:
    x1                4.941    0.093   52.991    0.000
    x2                5.984    0.100   60.096    0.000
    x3                2.487    0.094   26.465    0.000
    x4                2.823    0.093   30.371    0.000
    x5                3.995    0.101   39.714    0.000
    x6                1.922    0.081   23.711    0.000
    x7                4.432    0.086   51.540    0.000
    x8                5.563    0.078   71.087    0.000
    x9                5.418    0.079   68.153    0.000
    visual            0.000
    textual           0.000
    speed             0.000

Variances:
    x1                0.551    0.137
    x2                1.258    0.155
    x3                0.882    0.128
    x4                0.434    0.070
    x5                0.508    0.082
    x6                0.266    0.050
    x7                0.849    0.114
    x8                0.515    0.095
    x9                0.658    0.096
    visual            0.805    0.171
    textual           0.913    0.137
    speed             0.305    0.078



Group 2 [Grant-White]:

                 Estimate  Std.err  Z-value  P(>|z|)
```

```
Latent variables:
  visual =~
    x1                1.000
    x2                0.599    0.100    5.979    0.000
    x3                0.784    0.108    7.267    0.000
  textual =~
    x4                1.000
    x5                1.083    0.067   16.049    0.000
    x6                0.912    0.058   15.785    0.000
  speed =~
    x7                1.000
    x8                1.201    0.155    7.738    0.000
    x9                1.038    0.136    7.629    0.000

Covariances:
  visual ~~
    textual           0.437    0.099    4.423    0.000
    speed             0.314    0.079    3.958    0.000
  textual ~~
    speed             0.226    0.072    3.144    0.002

Intercepts:
    x1                4.930    0.097   50.763    0.000
    x2                6.200    0.091   68.379    0.000
    x3                1.996    0.085   23.455    0.000
    x4                3.317    0.092   35.950    0.000
    x5                4.712    0.100   47.173    0.000
    x6                2.469    0.091   27.248    0.000
    x7                3.921    0.086   45.555    0.000
    x8                5.488    0.087   63.257    0.000
    x9                5.327    0.085   62.786    0.000
    visual            0.000
    textual           0.000
    speed             0.000

Variances:
    x1                0.645    0.127
    x2                0.933    0.121
    x3                0.605    0.096
    x4                0.329    0.062
    x5                0.384    0.073
    x6                0.437    0.067
    x7                0.599    0.090
    x8                0.406    0.089
    x9                0.532    0.086
    visual            0.722    0.161
    textual           0.906    0.136
    speed             0.475    0.109
```

さらに‘グループ等号制約’を追加することができる．因子負荷量に加えて，以下のキーワードが現在利用可能である：

- "intercepts"：観測変数の切片
- "means"：潜在変数の切片/平均
- "residuals"：観測変数の残差分散
- "residual.covariances"：観測変数の残差共分散
- "lv.variances"：潜在変数の（残差） 分散
- "lv.covariances"：潜在変数の（残差）共分散
- "regressions"：モデルの全ての回帰係数

自由パラメータとしたいものを除いたもの全て（例えば，すべての因子負荷量と切片）を母集団全体で制約したい場合はどうすればよいだろうか．このシナリオのために，引数 group.partial を使って，自由パラメータとしたいパラメータの名前を指定することができる．例えば：

```
> fit <- cfa(HS.model, data=HolzingerSwineford1939, group="school",
+            group.equal=c("loadings", "intercepts"),
+            group.partial=c("visual=~x2", "x7~1"))
```

6.2.3 測定の不変性

いくつかの母集団全体で，CFA モデルの測定の不変性の検定に関心があるとき，measurementInvariance 関数を使うことができる．これは，パラメータに対して次第に制約を多くしながら，特定の順序でいくつかの複数の母集団の分析を行う．（注意：バージョン 0.5 からは，measurementInvariance() 関数は，semTools パッケージに移された.）各モデルは，カイ 2 乗の差の検定を用いて，ベースラインのモデルおよび前のモデルと比較される．さらに，cfi 当てはめ測度の差も表示される．この関数の現在の機能はまだ少し原始的であるが，**lavaan**パッケージのいろいろな構成要素を用いて高次関数（例えば，measurementInvariance 関数）を作るのに利用する方法について例示する．これを商用のソフトウェアでは実行するのは難しい．

```
> library(semTools)
> measurementInvariance(HS.model, data=HolzingerSwineford1939, group="school")

Measurement invariance tests:

Model 1: configural invariance:
   chisq       df   pvalue      cfi    rmsea      bic
 115.851   48.000    0.000    0.923    0.097 7706.822

Model 2: weak invariance (equal loadings):
   chisq       df   pvalue      cfi    rmsea      bic
 124.044   54.000    0.000    0.921    0.093 7680.771

[Model 1 versus model 2]
  delta.chisq      delta.df delta.p.value     delta.cfi
        8.192         6.000         0.224         0.002

Model 3: strong invariance (equal loadings + intercepts):
   chisq       df   pvalue      cfi    rmsea      bic
 164.103   60.000    0.000    0.882    0.107 7686.588

[Model 1 versus model 3]
```

```
  delta.chisq      delta.df delta.p.value     delta.cfi
       48.251        12.000         0.000         0.041

[Model 2 versus model 3]
  delta.chisq      delta.df delta.p.value     delta.cfi
       40.059         6.000         0.000         0.038

Model 4: equal loadings + intercepts + means:
   chisq       df   pvalue      cfi    rmsea      bic
 204.605   63.000    0.000    0.840    0.122 7709.969

[Model 1 versus model 4]
  delta.chisq      delta.df delta.p.value     delta.cfi
       88.754        15.000         0.000         0.083

[Model 3 versus model 4]
  delta.chisq      delta.df delta.p.value     delta.cfi
       40.502         3.000         0.000         0.042
```

group.partial 引数を使い，いくつかのパラメータを自由にしておくことによって，測定不変性の検定を行うことができる．

7 成長曲線モデル

別の重要な種類の潜在変数モデルに，潜在的な成長曲線モデルがある．成長モデリングは，縦断的（経時的）データの分析に用いられることが多い．この種のデータでは，特性値が数回計測され，時間経過における変化を調査したい．多くの場合，時間変化による軌道は 1 次または 2 次の曲線としてモデル化できる．変量効果を考えることにより，個体差を捕えることができる．変量効果を，成長因子と呼ばれる（連続的な）潜在変数によってうまく表現することができる．下記の例では，Demo.growth という名前の簡単な人工データを利用する．これは，読解力尺度に関する標準化された得点のデータで，4 時点で計測したものとしている．この 4 時点データに対して線形成長モデルを当てはめるためには，2 つの潜在変数，つまり，ランダムな切片とランダムな傾きを持つモデルを指定する必要がある：

lavaan syntax

```
# 4 時点の線形成長曲線
# 母数の切片と傾き
 i =~ 1*t1 + 1*t2 + 1*t3 + 1*t4
 s =~ 0*t1 + 1*t2 + 2*t3 + 3*t4
```

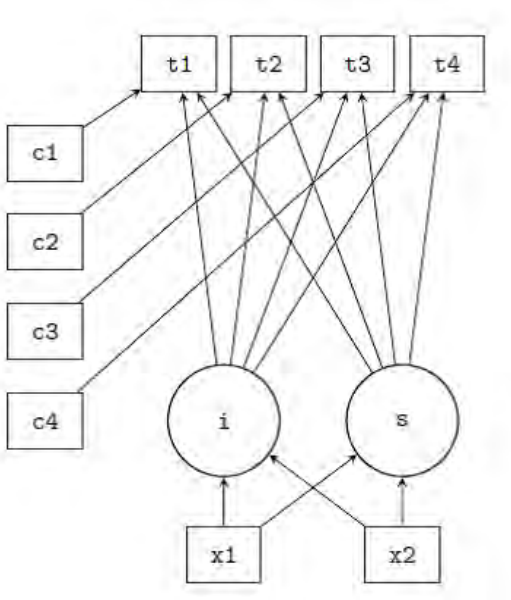
このモデルでは，成長関数のすべての係数を固定している．**lavaan**パッケージは，このモデルを当てはめるための特別な関数 growth を提供する．

```
> model <- ' i =~ 1*t1 + 1*t2 + 1*t3 + 1*t4
+            s =~ 0*t1 + 1*t2 + 2*t3 + 3*t4 '
> fit <- growth(model, data=Demo.growth)
> summary(fit)
lavaan (0.5-12) converged normally after 44 iterations

  Number of observations                           400
```

```
  Estimator                                         ML
  Minimum Function Test Statistic                8.069
  Degrees of freedom                                 5
  P-value (Chi-square)                           0.152

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  i =~
    t1                1.000
    t2                1.000
    t3                1.000
    t4                1.000
  s =~
    t1                0.000
    t2                1.000
    t3                2.000
    t4                3.000

Covariances:
  i ~~
    s                 0.618    0.071    8.686    0.000

Intercepts:
    t1                0.000
    t2                0.000
    t3                0.000
    t4                0.000
    i                 0.615    0.077    8.007    0.000
    s                 1.006    0.042   24.076    0.000

Variances:
    t1                0.595    0.086
    t2                0.676    0.061
    t3                0.635    0.072
    t4                0.508    0.124
    i                 1.932    0.173
    s                 0.587    0.052
```

growth 関数は、技術的には sem 関数とほとんど同一である．しかし，平均構造が自動的に仮定され，デフォルトで切片は 0 に固定されるが，潜在変数である切片/平均は自由に推定される．もう少し複雑なモデルとして，潜在的な成長因子に影響する 2 つの説明変数（x1 と x2）を加える．さらに，4 時点の結果測度に影響する時間で変動する共変量もモデルに加える．これに対応するモデルのグラフ表現と**lavaan**構文を次に示す．

コピー&ペーストが簡単にできるように，時間で変動する共変量を持つ線形成長モデルを指定し，当てはめるための R の完全なコードを次に示す．



lavaan syntax

```
# 固定した係数を持つ
# 切片と傾き
  i =~ 1*t1 + 1*t2 + 1*t3 + 1*t4
  s =~ 0*t1 + 1*t2 + 2*t3 + 3*t4
# 回帰
  i ~ x1 + x2
  s ~ x1 + x2
# 時間で変動する共変量
  t1 ~ c1
  t2 ~ c2
  t3 ~ c3
  t4 ~ c4
```

R code

```
# 時間で変動する共変量を持つ線形成長モデル
model <- '
# 固定された切片と傾き
 i =~ 1*t1 + 1*t2 + 1*t3 + 1*t4
 s =~ 0*t1 + 1*t2 + 2*t3 + 3*t4
# 回帰
 i ~ x1 + x2
 s ~ x1 + x2
# 時間で変動する共変量
 t1 ~ c1
 t2 ~ c2
 t3 ~ c3
 t4 ~ c4
'
fit <- growth(model, data=Demo.growth)
summary(fit)
```

8　カテゴリカル変数の利用

2 値変数，順序尺度変数，名目変数は，カテゴリカル（連続的ではない）変数として考えることができる．モデルの中で，これらの変数が外生（独立）変数か内生（従属）変数かということは非常に大きな違いをもたらす．

8.1 外生カテゴリカル変数

2 値の外生共変量（例えば，性別）があるときは，それをダミー変数に再コード化（0/1）するだけでよい．古典的な回帰モデルで行うのと同じことである．外生の順序尺度変数の場合，順序を反映することができるコーディング（例えば，$1, 2, 3, \ldots$）を用いて他の（数値）変数と同じように取り扱えば良い．K（$K > 2$）水準の名目カテゴリカル変数の場合，それを $K - 1$ 個のダミー変数に置き換える必要がある．これも古典的な回帰分析の場合と同じである．

8.2 内生カテゴリカル変数

lavaanのバージョン 0.5 シリーズは，2 値または順序尺度（名目ではない）の内生変数を取り扱うことができる．現在，3 段階 WLS 法のみが可能である（あるロバストな改良版を含む）．2 値または順序尺度データに対しては，2 つの方法を選択できる．

1. 分析を行う前に，データフレームにあるそれらの変数を ‘順序付き’ 変数であると宣言する（R の基本パッケージにある ordered() 関数を利用）．例えば，データフレーム（‘Data’ とする）にある 4 つの変数（例えば，item1, item2, item3, item4) を順序尺度であると宣言するとき，次のようにする．
```
> Data[,c("item1","item2","item3","item4")] <-
+     lapply(Data[,c("item1","item2","item3","item4")], ordered)
```
2. モデルへの当てはめの関数（例えば，`cfa`，`sem`，`growth`，`lavaan` など）の 1 つを用いるときに引数 `ordered=`を利用する．例えば，4 つの 2 値または順序尺度の変数（例えば，item1, item2, item3, item4）があるとき，次のようにする．
```
> fit <- cfa(myModel, data=myData, ordered=c("item1","item2","item3","item4"))
```

どちらの場合も，**lavaan**は自動的に WLSMV に自動的に切り替えられる：モデルのパラメータを推定する際には対角上での重み付き最小 2 乗法（DWLS）を利用するが，ロバストな標準誤差，平均と分散で調整した検定統計量を計算する場合には，完全重み付き行列を用いる．いくつかの例（多母集団の例を含む）を付録で示している．

9 追加情報

9.1 入力として共分散行列を利用する

完全なデータセットがなくても，標本共分散行列があれば，モデルに当てはめることができる．平均構造を必要とするときは，平均ベクトルも提供する必要がある．統計量の標本値のみを提供するときは，標本積率を計算するのに用いたデータ数を指定する必要があることが重要である．標本共分散行列を入力とする利用法を次に例示する：

```
> lower <- '
+  11.834
+   6.947    9.364
+   6.819    5.091   12.532
+   4.783    5.028    7.495    9.986
+  -3.839   -3.889   -3.841   -3.625   9.610
+ -21.899  -18.831  -21.748  -18.775  35.522  450.288 '
> # classic wheaton et al model
```

```
> wheaton.cov <- getCov(lower, names=c("anomia67","powerless67", "anomia71",
+ "powerless71","education","sei"))
> wheaton.model <- '
+   # latent variables
+     ses =~ education + sei
+     alien67 =~ anomia67 + powerless67
+     alien71 =~ anomia71 + powerless71
+
+   # regressions
+     alien71 ~ alien67 + ses
+     alien67 ~ ses
+
+   # correlated residuals
+     anomia67 ~~ anomia71
+     powerless67 ~~ powerless71
+ '
> fit <- sem(wheaton.model, sample.cov=wheaton.cov, sample.nobs=932)
> summary(fit, standardized=TRUE)
lavaan (0.5-12) converged normally after 82 iterations

  Number of observations                           932

  Estimator                                         ML
  Minimum Function Test Statisti                 4.735
  Degrees of freedom                                 4
  P-value (Chi-square)                           0.316

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)   Std.lv  Std.all
Latent variables:
  ses =~
    education         1.000                               2.607    0.842
    sei               5.219    0.422   12.364    0.000   13.609    0.642
  alien67 =~
    anomia67          1.000                               2.663    0.774
    powerless67       0.979    0.062   15.895    0.000    2.606    0.852
  alien71 =~
    anomia71          1.000                               2.850    0.805
    powerless71       0.922    0.059   15.498    0.000    2.628    0.832

Regressions:
  alien71 ~
    alien67           0.607    0.051   11.898    0.000    0.567    0.567
    ses              -0.227    0.052   -4.334    0.000   -0.207   -0.207
  alien67 ~
    ses              -0.575    0.056  -10.195    0.000   -0.563   -0.563

Covariances:
```

```
  anomia67 ~~
    anomia71          1.623    0.314    5.176    0.000    1.623    0.356
  powerless67 ~~
    powerless71       0.339    0.261    1.298    0.194    0.339    0.121

Variances:
    education         2.801    0.507                      2.801    0.292
    sei             264.597   18.126                    264.597    0.588
    anomia67          4.731    0.453                      4.731    0.400
    powerless67       2.563    0.403                      2.563    0.274
    anomia71          4.399    0.515                      4.399    0.351
    powerless71       3.070    0.434                      3.070    0.308
    ses               6.798    0.649                      1.000    1.000
    alien67           4.841    0.467                      0.683    0.683
    alien71           4.083    0.404                      0.503    0.503
```

共分散行列の下半分の要素しか持たないとき（おそらく，教科書や論文からのデータの場合），getCov() 関数を用いると，完全な共分散行列を簡単に作ることができる（変数名を含んで）．共分散行列の下三角部分は 2 つのシングルコーテーションマークで囲まれていることに注意．よって，さらなる追加的な柔軟性がある．例えば，コメントや空行をいれることができる．数字がコンマやセミコロンで区切られていてもよい．getCov() 関数についてもっと知りたい場合，オンラインヘルプを参照のこと．

複数の母集団（グループ）があるとき，`sample.cov` 引数は，各母集団の標本分散共分散行列を別々のリストの要素として持つリストでなければならない．平均構造が必要ならば，`sample.mean` 引数は，各母集団の標本平均を含むリストでなければならない．最後に，`sample.nobs` 引数は，各母集団のデータ数を含むリストか整数のベクトルのどちらかでよい．

9.2 推定量，標準誤差，欠測値

9.2.1 推定量

lavaanパッケージのデフォルトの推定法は，最尤法である（`estimator = "ML"`）．**lavaan**で現在利用できる他の推定法を次に示す：

- `"GLS"` 一般化最小 2 乗法．完全データに対してのみ利用可．
- `"WLS"` 重み付き最小 2 乗法（ADF 推定ともいう）．完全データに対してのみ利用可．
- `"MLM"` ロバスト標準誤差と Satorra-Bentler scaled test statistic を持つ最尤法．完全データに対してのみ利用可．
- `"MLF"` 1 次導関数に基づく標準誤差と伝統的な検定統計量を持つ最尤法．完全データと欠測データに対して利用可．
- `"MLR"` ロバスト（Huber-White）標準誤差と，漸近的に Yuan-Bentler 検定統計量と等しい標準化検定統計量を持つ最尤法．完全データと欠測データに対して利用可．

最尤法（`"ML"`，`"MLM"`，`"MLF"`，`"MLR"`）を用いたとき，**lavaan**のデフォルトでは，いわゆる偏りのある標本共分散行列（要素は，$n-1$ の代わりに n によって割られた）に基づく分析が行われる．これは内部的に処理され，ユーザーが行う必要はない．これに加え，カイ 2 乗統計量は，最小の関数値に因子 n（$n-1$ の代わりに）を掛けることによって計算される．これは，Mplus プログラムと同じである．不偏共分散を用い，カイ 2 乗統計量を計算する乗数として $n-1$ を用いたいならば，関数を利用するとき

に，引数 likelihood="wishart"を指定する必要がある．例えば：

```
> fit <- cfa(HS.model, data = HolzingerSwineford1939, likelihood = "wishart")
> fit
lavaan (0.5-12) converged normally after 41 iterations

  Number of observations                           301

  Estimator                                         ML
  Minimum Function Chi-square                   85.022
  Degrees of freedom                                24
  P-value (Chi-square)                           0.000
```

検定統計量の値は，EQ や LISREL，AMOS のようなプログラムが出力する値に近いであろう．なぜなら，それらのプログラムは全て，最尤法を使うとき ‘Wishart’ アプローチを用いているからである．一方，Mplus は，最尤法への ‘通常の” アプローチを取る．

9.2.2 欠測値

データに欠測値があるとき，デフォルトでの処理はリスト単位での削除となる．欠測のメカニズムが MCAR（無作為に完全に欠測する）または MAR（無作為に欠測する）のとき，**lavaan**パッケージはケースワイズ（または ‘完全情報’）最尤推定を行う．モデル当てはめの関数の呼び出しにおいて，引数 missing="ml" を用いて，特徴をオンにすることができる．制約のない（h1）モデルが自動的に推定され，全ての一般の当てはめの尺度が利用可能である．

9.2.3 標準誤差

標準誤差は，（デフォルトでは）期待情報行列に基づく．唯一の例外は，欠測値があり，完全情報 ML（missing="ml"の指定により）を用いるときである．この場合，観測された情報行列を用いて標準誤差が計算される．ユーザーは information 引数を用いて，"expected"または "observed"を指定することにより，これを変更することができる．推定量が単に”ML”のときは，se 引数を用いて"robust.mlm"や"robust.mlr"，"first.order"を指定することにより，ロバスト標準誤差に変更することもできる．これらが必要でないならば，単に"none"とすればよい．これは，検定統計量には影響しない．実際，あなたは"test"引数を用いて，独立して検定統計量を選ぶことができる．指定できるのは，"standard"または"Satorra-Bentler"，"Yuan-Bentler"である．

9.2.4 ブートストラッピング

lavaanでブートストラップを利用する方法は 2 つある．モデルを当てはめるとき，se="boot"または test="boot"を指定するか（この場合，ブートストラップ標準誤差またはブートストラップに基づく p 値を得る），bootstrapLavaan() 関数を用いるかである．後者の場合，既に当てはめた **lavaan**オブジェクトが必要である．

9.2.5 間接効果と媒介分析

応答変数 Y，予測変数 X，媒介変数 M がある 3 変数の古典的な媒介状況を考える．例として，これら 3 変数を含む簡単なデータセットを作り，X が Y に直接影響を与え，X が M を媒介して Y に影響を与えるパス解析モデルを当てはめる．

```
> set.seed(1234)
```

```
> X <- rnorm(100)
> M <- 0.5*X + rnorm(100)
> Y <- 0.7*M + rnorm(100)
> Data <- data.frame(X = X, Y = Y, M = M)
> model <- ' # 直接効果
+             Y ~ c*X
+           # 媒介
+             M ~ a*X
+             Y ~ b*M
+           # 間接効果 (a*b)
+             ab := a*b
+           # 全体の効果
+             total := c + (a*b)
+ '
> fit <- sem(model, data=Data)
> summary(fit)
lavaan (0.5-12) converged normally after 13 iterations

  Number of observations                           100

  Estimator                                         ML
  Minimum Function Test Statistic                0.000
  Degrees of freedom                                 0
  P-value (Chi-square)                           1.000

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)
Regressions:
  Y ~
    X         (c)     0.036    0.104    0.348    0.728
  M ~
    X         (a)     0.474    0.103    4.613    0.000
  Y ~
    M         (b)     0.788    0.092    8.539    0.000

Variances:
    Y                 0.898    0.127
    M                 1.054    0.149

Defined parameters:
    ab                0.374    0.092    4.059    0.000
    total             0.410    0.125    3.287    0.001
```

この例は，**lavaan**モデル構文で ":=" オペレータを利用する方法を示している．このオペレータは，オリジナルのモデルパラメータの任意の関数値を取る新しいパラメータを '定義' する．しかし，この関数は，モデル構文で明示されたパラメータのラベルにより指定さなければならない．デフォルトでは，これらの定義されたパラメータの標準誤差は，デルタ法を用いて計算される．他のモデルと同様，当て

はめの関数の中で se="bootstrap"と単に指定することにより，ブートストラップ標準誤差を要求することができる．

9.3 修正インデックス

修正インデックスは，summary の呼び出しにおいて modindices=TRUE 引数を加えるか，直接，modindices 関数を利用することによって要求することができる．modindices 関数は，データフレームを返す．例えば，因子負荷量に関する修正インデックスだけを参照するには，次のようにすればよい：

```
> fit <- cfa(HS.model, data=HolzingerSwineford1939)
> mi <- modindices(fit)
> mi[mi$op == "=~",] #$
       lhs op rhs     mi    epc sepc.lv sepc.all sepc.nox
1   visual =~  x1     NA     NA      NA       NA       NA
2   visual =~  x2  0.000  0.000   0.000    0.000    0.000
3   visual =~  x3  0.000  0.000   0.000    0.000    0.000
4   visual =~  x4  1.211  0.077   0.069    0.059    0.059
5   visual =~  x5  7.441 -0.210  -0.189   -0.147   -0.147
6   visual =~  x6  2.843  0.111   0.100    0.092    0.092
7   visual =~  x7 18.631 -0.422  -0.380   -0.349   -0.349
8   visual =~  x8  4.295 -0.210  -0.189   -0.187   -0.187
9   visual =~  x9 36.411  0.577   0.519    0.515    0.515
10 textual =~  x1  8.903  0.350   0.347    0.297    0.297
11 textual =~  x2  0.017 -0.011  -0.011   -0.010   -0.010
12 textual =~  x3  9.151 -0.272  -0.269   -0.238   -0.238
13 textual =~  x4     NA     NA      NA       NA       NA
14 textual =~  x5  0.000  0.000   0.000    0.000    0.000
15 textual =~  x6  0.000  0.000   0.000    0.000    0.000
16 textual =~  x7  0.098 -0.021  -0.021   -0.019   -0.019
17 textual =~  x8  3.359 -0.121  -0.120   -0.118   -0.118
18 textual =~  x9  4.796  0.138   0.137    0.136    0.136
19   speed =~  x1  0.014  0.024   0.015    0.013    0.013
20   speed =~  x2  1.580 -0.198  -0.123   -0.105   -0.105
21   speed =~  x3  0.716  0.136   0.084    0.075    0.075
22   speed =~  x4  0.003 -0.005  -0.003   -0.003   -0.003
23   speed =~  x5  0.201 -0.044  -0.027   -0.021   -0.021
24   speed =~  x6  0.273  0.044   0.027    0.025    0.025
25   speed =~  x7     NA     NA      NA       NA       NA
26   speed =~  x8  0.000  0.000   0.000    0.000    0.000
27   speed =~  x9  0.000  0.000   0.000    0.000    0.000
```

修正インデックスは，自由ではない（あるいは冗長ではない）パラメータに対して表示される．

修正インデックスは，期待されるパラメータ変化の値（epc 列）という補足情報を持つ．最後の 2 列は，それぞれ標準化された，完全に標準化された EPC 値である．

9.4 当てはめたモデルからの情報の抽出

summary 関数は，当てはめたモデルの洗練された概要を与えるが，表示するだけである．さらに処理を続けるために数値データが必要なときには，‘抽出’ のための関数を用いる方がよいだろう．当てはめたモデルの推定母数を抽出するための coef 関数についてはすでに見ている．

9.4.1 parameterEstimates

parameterEstimates 関数は，母数の推定値のみならず，標準誤差の値，z 値，標準化されたパラメーター値をデータフレームとして抽出する．例えば：

```
> parameterEstimates(fit)
       lhs op     rhs   est    se      z pvalue ci.lower ci.upper
1   visual =~      x1 1.000 0.000     NA     NA    1.000    1.000
2   visual =~      x2 0.553 0.100  5.554      0    0.358    0.749
3   visual =~      x3 0.729 0.109  6.685      0    0.516    0.943
4  textual =~      x4 1.000 0.000     NA     NA    1.000    1.000
5  textual =~      x5 1.113 0.065 17.014      0    0.985    1.241
6  textual =~      x6 0.926 0.055 16.703      0    0.817    1.035
7    speed =~      x7 1.000 0.000     NA     NA    1.000    1.000
8    speed =~      x8 1.180 0.165  7.152      0    0.857    1.503
9    speed =~      x9 1.082 0.151  7.155      0    0.785    1.378
10      x1 ~~      x1 0.549 0.114  4.833      0    0.326    0.772
11      x2 ~~      x2 1.134 0.102 11.146      0    0.934    1.333
12      x3 ~~      x3 0.844 0.091  9.317      0    0.667    1.022
13      x4 ~~      x4 0.371 0.048  7.779      0    0.278    0.465
14      x5 ~~      x5 0.446 0.058  7.642      0    0.332    0.561
15      x6 ~~      x6 0.356 0.043  8.277      0    0.272    0.441
16      x7 ~~      x7 0.799 0.081  9.823      0    0.640    0.959
17      x8 ~~      x8 0.488 0.074  6.573      0    0.342    0.633
18      x9 ~~      x9 0.566 0.071  8.003      0    0.427    0.705
19  visual ~~  visual 0.809 0.145  5.564      0    0.524    1.094
20 textual ~~ textual 0.979 0.112  8.737      0    0.760    1.199
21   speed ~~   speed 0.384 0.086  4.451      0    0.215    0.553
22  visual ~~ textual 0.408 0.074  5.552      0    0.264    0.552
23  visual ~~   speed 0.262 0.056  4.660      0    0.152    0.373
24 textual ~~   speed 0.173 0.049  3.518      0    0.077    0.270
```

9.4.2 standardizedSolution

standardizedSolution 関数は，parameterEstimates 関数と似ているが，標準化されない，および標準化されたパラメータ推定値をのみを示す．

9.4.3 fitted.values

fitted 関数と fitted.values 関数は，当てはめたモデルの意味された（当てはめた）共分散行列（および平均ベクトル）を返す．

```
> fit <- cfa(HS.model, data = HolzingerSwineford1939)
> fitted(fit)
$cov
   x1    x2    x3    x4    x5    x6    x7    x8    x9
x1 1.358
x2 0.448 1.382
x3 0.590 0.327 1.275
x4 0.408 0.226 0.298 1.351
x5 0.454 0.252 0.331 1.090 1.660
x6 0.378 0.209 0.276 0.907 1.010 1.196
```

```
x7 0.262 0.145 0.191 0.173 0.193 0.161 1.183
x8 0.309 0.171 0.226 0.205 0.228 0.190 0.453 1.022
x9 0.284 0.157 0.207 0.188 0.209 0.174 0.415 0.490 1.015

$mean
x1 x2 x3 x4 x5 x6 x7 x8 x9
 0  0  0  0  0  0  0  0  0
```

9.4.4 residuals

resid，または residuals 関数は，当てはめたモデルの（標準化されていない）残差を返す．これは単に，観測された共分散と推定された共分散の差，および観測された平均と推定された平均値のベクトルの差である．推定量が最尤推定量ならば，normalized 残差，standardized 残差を得ることもできる．

```
> fit <- cfa(HS.model, data=HolzingerSwineford1939)
> resid(fit, type="standardized")
$cov
   x1     x2     x3     x4     x5     x6     x7     x8     x9
x1     NA
x2 -2.196     NA
x3 -1.199  2.692  0.000
x4  2.465 -0.283 -1.948     NA
x5 -0.362 -0.610 -4.443  0.856     NA
x6  2.032  0.661 -0.701     NA  0.633     NA
x7 -3.787 -3.800 -1.882  0.839 -0.837 -0.321  0.000
x8 -1.456 -1.137 -0.305 -2.049 -1.100 -0.635  3.804     NA
x9  4.062  1.517  3.328  1.237  1.723  1.436 -2.772     NA     NA

$mean
x1 x2 x3 x4 x5 x6 x7 x8 x9
 0  0  0  0  0  0  0  0  0
```

9.4.5 vcov

vcov 関数は，推定されたパラメータの共分散行列の推定値を返す．

9.4.6 AIC と BIC

当てはめたモデルの AIC と BIC の値は，AIC，BIC 関数により知ることができる．

9.4.7 fitMeasures

fitMeasures 関数は，名前付き数値ベクトルとして **lavaan**により計算された適合の測度すべてを返す．1 つの値のみ，例えば CFI のみが必要な場合は，第 2 引数としてその名前を（小文字で）与えればよい：

```
> fit <- cfa(HS.model, data = HolzingerSwineford1939)
> fitMeasures(fit, "cfi")
  cfi
0.931
```

9.4.8 inspect

当てはめた **lavaan**オブジェクト（関数 cfa，sem，growth の呼び出しによって返されたオブジェクト）の中を見たいとき，いろいろなオプションをつけて inspect 関数を用いれば良い．

```
> inspect(fit)
$lambda
   visual textul speed
x1      0      0     0
x2      1      0     0
x3      2      0     0
x4      0      0     0
x5      0      3     0
x6      0      4     0
x7      0      0     0
x8      0      0     5
x9      0      0     6

$theta
   x1 x2 x3 x4 x5 x6 x7 x8 x9
x1  7
x2  0  8
x3  0  0  9
x4  0  0  0 10
x5  0  0  0  0 11
x6  0  0  0  0  0 12
x7  0  0  0  0  0  0 13
x8  0  0  0  0  0  0  0 14
x9  0  0  0  0  0  0  0  0 15

$psi
        visual textul speed
visual  16
textual 19     17
speed   20     21     18
```

各モデル行列におけるパラメータの初期値を知るには，次を入力する．

```
> inspect(fit, what="start")
$lambda
   visual textul speed
x1  1.000  0.000 0.000
x2  0.778  0.000 0.000
x3  1.107  0.000 0.000
x4  0.000  1.000 0.000
x5  0.000  1.133 0.000
x6  0.000  0.924 0.000
x7  0.000  0.000 1.000
x8  0.000  0.000 1.225
x9  0.000  0.000 0.854

$theta
```

```
   x1    x2    x3    x4    x5    x6    x7    x8    x9
x1 0.679
x2 0.000 0.691
x3 0.000 0.000 0.637
x4 0.000 0.000 0.000 0.675
x5 0.000 0.000 0.000 0.000 0.830
x6 0.000 0.000 0.000 0.000 0.000 0.598
x7 0.000 0.000 0.000 0.000 0.000 0.000 0.592
x8 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.511
x9 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.508

$psi
        visual textul speed
visual  0.05
textual 0.00   0.05
speed   0.00   0.00   0.05
```

lavaanが内部でモデルをどう表現しているかは，次を入力することにより知ることができる．

```
> inspect(fit, what="list")
   id     lhs op     rhs user group free ustart exo label eq.id unco
1   1  visual =~      x1    1     1    0      1   0           0    0
2   2  visual =~      x2    1     1    1     NA   0           0    1
3   3  visual =~      x3    1     1    2     NA   0           0    2
4   4 textual =~      x4    1     1    0      1   0           0    0
5   5 textual =~      x5    1     1    3     NA   0           0    3
6   6 textual =~      x6    1     1    4     NA   0           0    4
7   7   speed =~      x7    1     1    0      1   0           0    0
8   8   speed =~      x8    1     1    5     NA   0           0    5
9   9   speed =~      x9    1     1    6     NA   0           0    6
10 10      x1 ~~      x1    0     1    7     NA   0           0    7
11 11      x2 ~~      x2    0     1    8     NA   0           0    8
12 12      x3 ~~      x3    0     1    9     NA   0           0    9
13 13      x4 ~~      x4    0     1   10     NA   0           0   10
14 14      x5 ~~      x5    0     1   11     NA   0           0   11
15 15      x6 ~~      x6    0     1   12     NA   0           0   12
16 16      x7 ~~      x7    0     1   13     NA   0           0   13
17 17      x8 ~~      x8    0     1   14     NA   0           0   14
18 18      x9 ~~      x9    0     1   15     NA   0           0   15
19 19  visual ~~  visual    0     1   16     NA   0           0   16
20 20 textual ~~ textual    0     1   17     NA   0           0   17
21 21   speed ~~   speed    0     1   18     NA   0           0   18
22 22  visual ~~ textual    0     1   19     NA   0           0   19
23 23  visual ~~   speed    0     1   20     NA   0           0   20
24 24 textual ~~   speed    0     1   21     NA   0           0   21
```

inspect 関数のオプションの詳細については，次を入力することにより表示される **lavaan**クラスのヘルプページを参照．

```
> class?lavaan
```

10 バグレポート，フィードバック

バグを見つけたり，何か変なことが生じたりしたとき，知らせてください．email アドレスは，Yves.Rosseel@UGent.be です．email のサブジェクトが [lavaan] で始まっていると，**lavaan**に関することだとわかります．生じた問題に対処（そして，バグを修正する）際，2 つのタイプの情報が必要です：

1. 問題の詳しい説明：起こったこと，表示されたエラーメッセージや警告，いつそれが生じたか．可能なら，エラーを引き起こした構文の再生可能な例の提供．
2. R での次のコマンドの出力:

   ```
   > sessionInfo()
   ```

   これは，あなたのプラットホームについての不可欠な情報，つまり，R のバージョン等の問題を特定するのに必要な情報を示す．

ソフトウェアと文書に関する提案も全て歓迎する．

.1 第 3 章：回帰とパス解析

```
# ex3.1
Data <- read.table("ex3.1.dat")
names(Data) <- c("y1","x1","x2")

model.ex3.1 <- ' y1 ~ x1 + x2 '
fit <- sem(model.ex3.1, data=Data)
summary(fit, standardized=TRUE, fit.measures=TRUE)

# ex3.4
Data <- read.table("ex3.4.dat")
names(Data) <- c("u1", "x1", "x3")
Data$u1 <- ordered(Data$u1)

model <- ' u1 ~ x1 + x3 '
fit <- sem(model, data=Data)
summary(fit, fit.measures=TRUE)

# ex3.11
Data <- read.table("ex3.11.dat")
names(Data) <- c("y1","y2","y3",
                 "x1","x2","x3")

model.ex3.11 <- ' y1 + y2 ~ x1 + x2 + x3
y3 ~ y1 + y2 + x2 '
fit <- sem(model.ex3.11, data=Data)
summary(fit, standardized=TRUE, fit.measures=TRUE)

# ex3.12
Data <- read.table("ex3.12.dat")
names(Data) <- c("u1","u2","u3","x1","x2","x3")
```

```
Data$u1 <- ordered(Data$u1)
Data$u2 <- ordered(Data$u2)
Data$u3 <- ordered(Data$u3)

model <- ' u1 + u2 ~ x1 + x2 + x3
               u3 ~ u1 + u2 + x2 '

fit <- sem(model, data=Data)
summary(fit, fit.measures=TRUE)

# Mplus の例 3.14
Data <- read.table("ex3.14.dat")
names(Data) <- c("y1","y2","u1","x1","x2","x3")
Data$u1 <- ordered(Data$u1)

model <- ' y1 + y2 ~ x1 + x2 + x3
                u1 ~ y1 + y2 + x2 '

fit <- sem(model, data=Data)
summary(fit, fit.measures=TRUE)
```

.2 第 5 章：確認的因子分析と構造方程式モデリング

```
# ex5.1
Data <- read.table("ex5.1.dat")
names(Data) <- paste("y", 1:6, sep="")
model.ex5.1 <- ' f1 =~ y1 + y2 + y3
                 f2 =~ y4 + y5 + y6 '

fit <- cfa(model.ex5.1, data=Data)
summary(fit, standardized=TRUE, fit.measures=TRUE)

# ex5.2
Data <- read.table("ex5.2.dat")
names(Data) <- c("u1","u2","u3","u4","u5","u6")
# 全ての変数を順序付き因子として宣言:
Data <- as.data.frame(lapply(Data, ordered))

model <- ' f1 =~ u1 + u2 + u3; f2 =~ u4 + u5 + u6 '
fit <- cfa(model, data=Data)
summary(fit, fit.measures=TRUE)

# ex5.3
Data <- read.table("ex5.3.dat")
names(Data) <- c("u1","u2","u3","y4","y5","y6")
Data$u1 <- ordered(Data$u1)
Data$u2 <- ordered(Data$u2)
Data$u3 <- ordered(Data$u3)

model <- ' f1 =~ u1 + u2 + u3
           f2 =~ y4 + y5 + y6 '
```

```
fit <- cfa(model, data=Data)
summary(fit, fit.measures=TRUE)

# ex5.6
Data <- read.table("ex5.6.dat")
names(Data) <- paste("y", 1:12, sep="")

model.ex5.6 <- ' f1 =~ y1 + y2 + y3
                 f2 =~ y4 + y5 + y6
                 f3 =~ y7 + y8 + y9
                 f4 =~ y10 + y11 + y12
                 f5 =~ f1 + f2 + f3 + f4 '

fit <- cfa(model.ex5.6, data=Data, estimator="ML")
summary(fit, standardized=TRUE, fit.measures=TRUE)

# ex5.8
Data <- read.table("ex5.8.dat")
names(Data) <- c(paste("y", 1:6, sep=""), paste("x", 1:3, sep=""))

model.ex5.8 <- ' f1 =~ y1 + y2 + y3
                 f2 =~ y4 + y5 + y6
                 f1 + f2 ~ x1 + x2 + x3 '

fit <- cfa(model.ex5.8, data=Data, estimator="ML")
summary(fit, standardized=TRUE, fit.measures=TRUE)

# ex5.9
Data <- read.table("ex5.9.dat")
names(Data) <- c("y1a","y1b","y1c","y2a","y2b","y2c")

model.ex5.9 <- ' f1 =~ 1*y1a + 1*y1b + 1*y1c
                 f2 =~ 1*y2a + 1*y2b + 1*y2c
                 y1a + y1b + y1c ~ i1*1
                 y2a + y2b + y2c ~ i2*1 '

fit <- cfa(model.ex5.9, data=Data)
summary(fit, standardized=TRUE, fit.measures=TRUE)

# ex5.11
Data <- read.table("ex5.11.dat")
names(Data) <- paste("y", 1:12, sep="")

model.ex5.11 <- ' f1 =~ y1 + y2 + y3
                  f2 =~ y4 + y5 + y6
                  f3 =~ y7 + y8 + y9
                  f4 =~ y10 + y11 + y12
                  f3 ~ f1 + f2
                  f4 ~ f3 '

fit <- sem(model.ex5.11, data=Data, estimator="ML")
summary(fit, standardized=TRUE, fit.measures=TRUE)
```

```
# ex5.14
Data <- read.table("ex5.14.dat")
names(Data) <- c("y1","y2","y3","y4","y5","y6", "x1","x2","x3", "g")

model.ex5.14 <- ' f1 =~ y1 + y2 + y3
                  f2 =~ y4 + y5 + y6
                  f1 + f2 ~ x1 + x2 + x3 '

fit <- cfa(model.ex5.14, data=Data, group="g", meanstructure=FALSE,
           group.equal=c("loadings"), group.partial=c("f1=~y3"))
summary(fit, standardized=TRUE, fit.measures=TRUE)

# ex5.15
Data <- read.table("ex5.15.dat")
names(Data) <- c("y1","y2","y3","y4","y5","y6", "x1","x2","x3", "g")

model.ex5.15 <- ' f1 =~ y1 + y2 + y3
                  f2 =~ y4 + y5 + y6
                  f1 + f2 ~ x1 + x2 + x3 '

fit <- cfa(model.ex5.15, data=Data, group="g", meanstructure=TRUE,
           group.equal=c("loadings", "intercepts"),
           group.partial=c("f1=~y3", "y3~1"))
summary(fit, standardized=TRUE, fit.measures=TRUE)

# ex5.16
Data <- read.table("ex5.16.dat")
names(Data) <- c("u1","u2","u3","u4","u5","u6","x1","x2","x3","g")
Data$u1 <- ordered(Data$u1)
Data$u2 <- ordered(Data$u2)
Data$u3 <- ordered(Data$u3)
Data$u4 <- ordered(Data$u4)
Data$u5 <- ordered(Data$u5)
Data$u6 <- ordered(Data$u6)

model <- ' f1 =~ u1 + u2 + c(l3,l3b)*u3
           f2 =~ u4 + u5 + u6
           # mimic
           f1 + f2 ~ x1 + x2 + x3
           # 同じ閾値であるが，第 2 グループの u3|1 に関しては自由
           u3 | c(u3,u3b)*t1
           # 第 2 グループの u3*のスケールを 1 に固定
           u3 ~*~ c(1,1)*u3
         '

fit <- cfa(model, data=Data, group="g", group.equal=c("loadings","thresholds"))
summary(fit, fit.measures=TRUE)

# ex5.20
Data <- read.table("ex5.20.dat")
names(Data) <- paste("y", 1:6, sep="")
```

```
model.ex5.20 <- ' f1 =~ y1 + lam2*y2 + lam3*y3
                  f2 =~ y4 + lam5*y5 + lam6*y6
                  f1 ~~ vf1*f1 + start(1.0)*f1 ## otherwise, neg vf2
                  f2 ~~ vf2*f2 + start(1.0)*f2 ##
                  y1 ~~ ve1*y1
                  y2 ~~ ve2*y2
                  y3 ~~ ve3*y3
                  y4 ~~ ve4*y4
                  y5 ~~ ve5*y5
                  y6 ~~ ve6*y6
                  # 制約
                  lam2^2*vf1/(lam2^2*vf1 + ve2) ==
                      lam5^2*vf2/(lam5^2*vf2 + ve5)
                  lam3*sqrt(vf1)/sqrt(lam3^2*vf1 + ve3) ==
                      lam6*sqrt(vf2)/sqrt(lam6^2*vf2 + ve6)
                  ve2 > ve5
                  ve4 > 0
                '

fit <- cfa(model.ex5.20, data=Data, estimator="ML")
summary(fit, standardized=TRUE, fit.measures=TRUE)
```

.3 第 6 章: 成長モデリング

```
# 6.1
Data <- read.table("ex6.1.dat")
names(Data) <- c("y11","y12","y13","y14")

model.ex6.1 <- ' i =~ 1*y11 + 1*y12 + 1*y13 + 1*y14
                 s =~ 0*y11 + 1*y12 + 2*y13 + 3*y14 '

fit <- growth(model.ex6.1, data=Data)
summary(fit, standardized=TRUE, fit.measures=TRUE)

#6.8
Data <- read.table("ex6.8.dat")
names(Data) <- c("y11","y12","y13","y14")

model.ex6.8 <- ' i =~ 1*y11 + 1*y12 + 1*y13 + 1*y14
                 s =~ 0*y11 + 1*y12 + start(2)*y13 + start(3)*y14 '

fit <- growth(model.ex6.8, data=Data)
summary(fit, standardized=TRUE, fit.measures=TRUE)

#6.9
Data <- read.table("ex6.9.dat")
names(Data) <- c("y11","y12","y13","y14")

model.ex6.9 <- ' i =~ 1*y11 + 1*y12 + 1*y13 + 1*y14
                 s =~ 0*y11 + 1*y12 + 2*y13 + 3*y14
```

```
                q =~ 0*y11 + 1*y12 + 4*y13 + 9*y14 '

fit <- growth(model.ex6.9, data=Data)
summary(fit, standardized=TRUE, fit.measures=TRUE)

#6.10
Data <- read.table("ex6.10.dat")
names(Data) <- c("y11","y12","y13","y14","x1","x2","a31","a32","a33","a34")

model.ex6.10 <- ' i =~ 1*y11 + 1*y12 + 1*y13 + 1*y14
                  s =~ 0*y11 + 1*y12 + 2*y13 + 3*y14
                  i + s ~ x1 + x2
                  y11 ~ a31
                  y12 ~ a32
                  y13 ~ a33
                  y14 ~ a34 '

fit <- growth(model.ex6.10, data=Data)
summary(fit, standardized=TRUE, fit.measures=TRUE)

#6.11
Data <- read.table("ex6.11.dat")
names(Data) <- c("y1","y2","y3","y4","y5")

modelex6.11 <- ' i =~ 1*y1 + 1*y2 + 1*y3 + 1*y4 + 1*y5
                 s1 =~ 0*y1 + 1*y2 + 2*y3 + 2*y4 + 2*y5
                 s2 =~ 0*y1 + 0*y2 + 0*y3 + 1*y4 + 2*y5 '

fit <- growth(modelex6.11, data=Data)
summary(fit, standardized=TRUE, fit.measures=TRUE)
```