

lavaanチュートリアル

Yves Rosseel
Department of Data Analysis
Ghent University (Belgium)
2017年1月24日

関西大学商学部 荒木孝治*

2017年7月14日

要約

本チュートリアルは、**lavaan**をはじめて使おうという人が最初に読むべき文献である。**lavaan**の基本的な構成要素、つまり、モデル記法、適合関数 (cfa, sem, growth), および主要な抽出関数 (summary, coef, fitted, inspect) の利用法を記している。2つの簡単な例を示した後、重要なトピックである平均構造、多母集団、成長曲線モデル、媒介分析、質的データに関して簡単に述べる。この中で、**lavaan**の利用を開始する（これ以上ではない）ための十分な情報をえることができる。^{*1}

目次

1	はじめに	2
2	lavaan パッケージのインストール	3
3	モデル式の記法	3
4	例 1: 検証的因子分析 (CFA)	5
5	例 2: 構造方程式モデリング	10
6	構文の追加	14
7	平均の導入	18
8	多母集団	20

* 翻訳に関する意見等は arakit@kansai-u.ac.jp まで。

*1 訳注) 原著 “The lavaan tutorial” は、<http://lavaan.ugent.be/tutorial/tutorial.pdf> よりダウンロード可能。

9	成長曲線モデル	31
10	カテゴリカル変数の利用	34
11	共分散行列を入力として利用	35
12	推定量, 標準誤差, 欠測値	37
13	間接効果と媒介分析	39
14	修正指標	40
15	当てはめたモデルからの情報の抽出	41

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(3.0.0以降)をインストールしておくこと. 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. 既知の重要な問題があり(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の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)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.

- 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、私に直接メールを送るか（警告するために）、ディスカッショングループに提示するか（議論するために）、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)。後者は、それがバグであり、修正すべきである私たちが認定した場合、有益な方法である。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。

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", dependencies = TRUE)
```

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)
```

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*2。

```
This is lavaan 0.5-13  
lavaan is BETA software! Please report any bugs.
```

このメッセージが表示されると、スタートできる。

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 ~ 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

*2 訳注) 2019年10月20日時点で、CRANで公開されている**lavaan**のバージョンは、0.5-19である。第3節以降、原著の出力では、バージョン0.5-13と0.5-17.693のものが混在している。翻訳でも基本的に原著のままとしている。

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：

```
y ~ f1 + f2 + x1 + x2
f1 ~ f2 + f3
f2 ~ f3 + x1 + x2
```

回帰式の中に潜在変数が含まれる場合、（顕在または潜在）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、manifest 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：

```
f1 =~ y1 + y2 + y3
f2 =~ y4 + y5 + y6
f3 =~ y7 + y8 + y9 + y10
```

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：

```
y1 ~~ y1 # 分散
y1 ~~ y2 # 共分散
f1 ~~ f2 # 共分散
```

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（数字‘1’で明示する）：

```
y1 ~ 1
f1 ~ 1
```

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

lavaanの完全なモデル構文は、これらのタイプを結合し、一重引用符で括ったものである。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
            ,

```

この構文を R プロンプトでインタラクティブに入力してもよいが、外部のテキストエディタでモデル全体を入力する方が便利である。その後、R コンソールにそれをコピー&ペーストする。RStudio を利用している場合、新しい‘R スクリプト’を開き、RStudio のソースエディタにモデル構文（と R のセッションに必要な他の全てのコマンド）を入力する。そして、スクリプトを保存しておく、後にそれを再利用することができる。上記のコードの見本は、`myModel` という名前の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。

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保存しておいてもよい（ファイル名を仮に `myModel.lav` とする）。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 例 1: 検証的因子分析 (CFA)

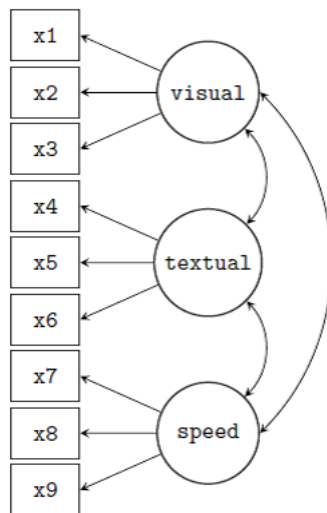
まず、関数 `cfa()` を用いて検証的因子分析 (CFA) を行う簡単な例を示す。これは、CFA モデルを当てはめるためのユーザーフレンドリーな関数である。`lavaan` パッケージには、`HolzingerSwineford1939` という名前の組み込みデータセットがある。R プロンプト (>) の後ろに次を入力すると、このデータセットのヘルプを参照することができる。

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。

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 (x_1, x_2, x_3) で計測される視覚 (*visual*) 因子
- 3 変数 (x_4, x_5, x_6) で計測される言語 (*textual*) 因子
- 3 変数 (x_7, x_8, x_9) で計測される速度 (*speed*) 因子

この 3 因子モデルを図示すると、次図のようになる。



このモデルを指定する **lavaan** の構文を次に示す

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

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：

潜在変数 = ~ 指標 1 + 指標 2 + 指標 3

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。中程にある特殊な記号 “=~” 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、関数 `cfa()` が次のような状況の面倒を裏で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設定している。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。

このモデル構文を一重引用符 (') を用いて入力する。

```
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。最初の引数は、ユーザーが指定するモデルであり、2番目の引数は、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、関数 `summary()` 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)
```

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。これがわかりにくくいと美的に不愉快なら、知らせてほしい。改善することを試みる。

```
lavaan (0.5-13) 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

```
User 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	0.554	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3つのパートからなる。最初の6行はヘッダーである。ヘッダーは次の情報を持つ。

- lavaan のバージョン番号
- lavaan が正常に収束したかどうか、繰り返し計算は何回必要だったか
- 分析で有効に利用された観測数
- パラメータ値を得るのに利用された推定量（ここでは ML）
- モデルの検定統計量の値、自由度、対応する P 値

次のパートは適合度の指標を含むが、それは、`fit.measures = TRUE` 引数を指定したときのみ表示される。これは `Model test baseline model` の行から始まり、`SRMR` の値で終わる。最後のパートでは、推定されたパラメータの値が示されている。それは、標準誤差に関する情報から始まる（情報行列が期待値か観測値か、標準誤差が標準のものか、ロバストなものか、ブートストラップによるものか）。そして、モデルに含まれる自由（そして固定された）パラメータを全て表にまとめて表示する。通常、まず潜在変数（Latent Variables）が、次いで、共分散（Covariances）と（残差）分散（Variances）が表示される。第1列（Estimate）は、各モデルパラメータに対する（推定された、または固定された）パラメータ値を示す。第2列（Std.err）は、推定された各パラメータに対する標準誤差である。第3列（Z-value）には Wald 統計量（これは単に、パラメータ値をその標準誤差で割ったもの）が、最終列（ $P(>|z|)$ ）には、母集団においてパラメータは0であるという帰無仮説を検定するときの p 値が表示される。第1例のまとめとして、3因子モデルを当てはめるのに必要な完全なコードを次に示す。

```
# 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)

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

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

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

このコードを R にコピー&ペーストするだけで実行できる。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：

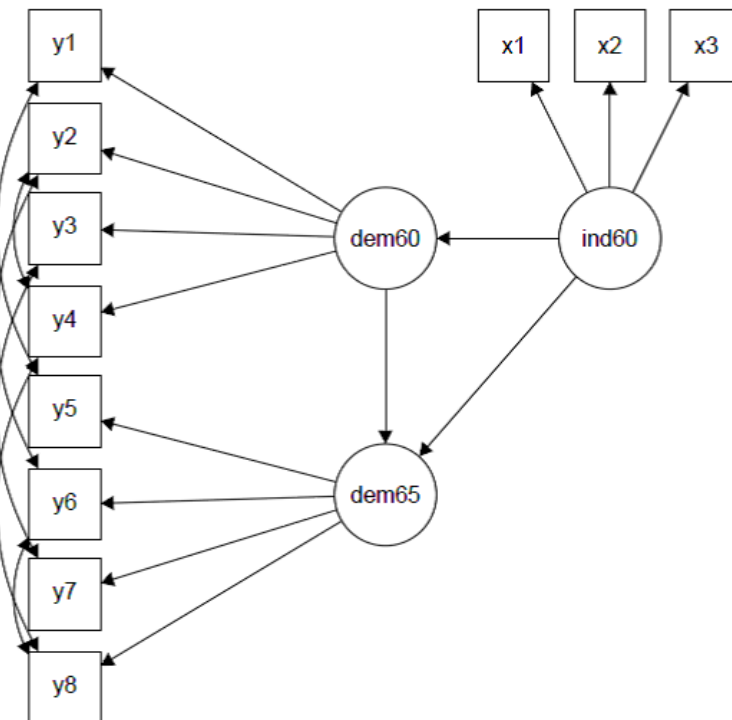
1. `lavaan` のモデル構文を利用して、モデルを作成する。この例では、潜在変数のみが定義されている。次の例では、他のタイプの式を利用する。
2. モデルに当てはめる。これには、観測変数を含むデータセット（標本共分散行列とデータ数でもよい。9.1 節参照）が必要である。この例では、関数 `cfa()` を使った。`lavaan` パッケージの他の関数として、構造方程式モデルと成長曲線モデルを当てはめるための関数 `sem()` と `growth()` が

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、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である `lavaan()` を代わりに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。

5 例 2：構造方程式モデリング

第 2 の例では、組み込みデータセット `PoliticalDemocracy` を利用する。これは、構造方程式モデリングに関する Bollen の本（1989 年出版）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。



対応する `lavaan` のモデル式を次に示す。

```

model <- '
# 測定モデル
ind60 =~ x1 + x2 + x3
dem60 =~ y1 + y2 + y3 + y4
dem65 =~ y5 + y6 + y7 + y8
# 回帰

```

```

dem60 ~ ind60
dem65 ~ ind60 + dem60
# 残差の相関
y1 ~~ y5
y2 ~~ y4 + y6
y3 ~~ y7
y4 ~~ y8
y6 ~~ y8
,

```

この例では、3つの式のタイプ、すなわち、潜在変数の定義式（オペレータ= \sim を利用）、回帰式（オペレータ \sim を利用）、（共）分散の定義式（オペレータ $\sim\sim$ を利用）を使う。回帰式は、Rの通常の式と似ている。（共）分散の定義は、次のようにする。

変数 $\sim\sim$ 変数

変数には、観測変数と潜在変数がある。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 $y1 \sim\sim y5$ 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）に計測されたものとなっている。表現 $y2 \sim\sim y4 + y6$ は、2つの表現 $y2 \sim\sim y4$ と $y2 \sim\sim y6$ を結合したものであることに注意（このような略記が可能）。モデル構文を次のように入力する。

```

model <- '
# 測定モデル
ind60 =~ x1 + x2 + x3
dem60 =~ y1 + y2 + y3 + y4
dem65 =~ y5 + y6 + y7 + y8
# 回帰
dem60 ~ ind60
dem65 ~ ind60 + dem60
# 残差の相関
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 (0.5-19.867) converged normally after 68 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()` 関数において、引数 `fit.measures=TRUE` を省略した。そのため、基本的なカイ 2 乗検定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 (`Std.lv`) では、潜在変数だけが標準化されている。第 2 列 (`Std.all`) では、潜在変数と観測変数が標準化されている。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：

```
library(lavaan) # セッションごとに 1 回必要
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
,
fit <- sem(model, data = PoliticalDemocracy)
summary(fit, standardized = TRUE)
```

6 構文の追加

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。これは次のようにする。

```
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。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。下記の構文では、潜在変数 visual と textual 間に共分散があるとしているが、他の潜在変数間の共分散は0としている。そのうえ、因子 speed 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われる。

```
# 3 因子モデル
visual =~ x1 + x2 + x3
textual =~ x4 + x5 + x6
speed =~ NA*x7 + x8 + x9
# 直交因子
visual ~~ 0*speed
textual ~~ 0*speed
# speed 因子の分散を固定
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するための簡単な方法がある。それは、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 のみを与える方法である。

```
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。

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、数値定数は関数 `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。

```
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。

パラメータのラベル `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。これは例えば、等式制約が必要なとき便利である（次項参照）。命名のメカニズムがどのように機能するかについて見るために、Political Democracy データに対して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
# 回帰
dem60 ~ ind60
dem65 ~ ind60 + dem60
# 残差 (共) 分散
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
2.180	1.819	1.257	1.058	1.265
dem65=~y6	dem65=~y7	dem65=~y8	dem60~ind60	dem65~ind60
1.186	1.280	1.266	1.483	0.572
dem65~dem60	y1~~y5	y2~~y4	y2~~y6	y3~~y7
0.837	0.624	1.313	2.153	0.795

```

y4~~y8      y6~~y8      x1~~x1      x2~~x2      x3~~x3
0.348       1.356       0.082       0.120       0.467
y1~~y1      y2~~y2      y3~~y3      y4~~y4      y5~~y5
1.891       7.373       5.067       3.148       2.351
y6~~y6      y7~~y7      y8~~y8      ind60~~ind60 dem60~~dem60
4.954       3.431       3.254       0.448       3.956
dem65~~dem65
0.172
>

```

関数 `coef()` 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。

特定のパラメータに対して利用者独自のラベルを選択できると便利であることが多い。この方法は、パラメータを固定することに似ているが、定数の代わりに、文字列（ラベル）を前方乗算に用いる。次に示す例は、指標 `x3` の因子負荷量に `myLabel` を‘ラベル’づける。

```

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。新しい構文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。

修飾子 これまで何度も（* オペレータを用いる）前方乗算方式の利用法を見てきた。これは、パラメータを固定したり、初期値を与えたり、パラメータにラベルづけたりする方法であった。こうしたオペレーションを修飾子と呼ぶことにする。別の修飾子を後に紹介する。式の右辺の各項は、1つのみ修飾子を持つことができる。同じパラメータに複数の修飾子を利用したい場合、同じ式の中で複数回項をリストする必要がある。例えば次のようにする。

```
f =~ y1 + y2 + myLabel*y3 + start(0.5)*y3 + y4
```


指標 y_3 は 2 回リストされており、おのおの異なる修飾子を持つ。パラメータは全ての異なる修飾子を累積するが、 y_3 を 1 つの指標として取り扱う。

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 x_2 と x_3 の因子負荷量を等しくしたいとする。lavaan では、このとき、2 つの自由パラメータを推定する代わりに、1 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）等式制約を行うには、ラベルを利用することができる。2 つのパラメータが同じラベルを持つと、lavaan は同じだと判断し、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。

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

注意 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。例えば次のようにする。

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

変数 x_2 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。 x_3 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 x_2 の因子負荷量と同じ値に設定される。

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実験的に追加された。例えば、次の回帰の例を考える。

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

次の例では、回帰係数を b_1 , b_2 , b_3 として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、 $b_1 = (b_2 + b_3)^2$ と $b_1 > \exp(b_2 + b_3)$ 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。これらの制約を課すには、次の構文を使う：

```

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。

7 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、次式の形となる：

$$\text{変数} \sim 1 \tag{1}$$

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&S 3 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。

```

# 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し（その値を固定したくない限り）、適合関数で引数 `meanstructure=TRUE` を与える方が便利である。例えば、H&S 3 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 (0.5-13) 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 =~					
		1.000			
		0.554	0.100	5.554	0.000
		0.729	0.109	6.685	0.000
textual =~					
		1.000			
		1.113	0.065	17.014	0.000
		0.926	0.055	16.703	0.000
speed =~					
		1.000			
		1.180	0.165	7.152	0.000
		1.082	0.151	7.155	0.000

Covariances:

visual ~~					
		0.408	0.074	5.552	0.000
		0.262	0.056	4.660	0.000
textual ~~					
		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			

Variiances:

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（Intercept）パラメータを含んでいる。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 乗検定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 x_1 , x_2 , x_3 , x_4 の切片を 0.5 に固定したいとすると、モデル構文は次のようになる：

```
# 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、 $x_1 \sim 0.5*1$, $x_2 \sim 0.5*1$ 等）となっている。

8 多母集団

`lavaan` パッケージは、多母集団（多グループ）の解析を完全にサポートしている。多母集団の解析を行うには、データセット内のグループ変数の名前を適合関数の引数 `group` として与える必要がある。

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モデルを当てはめている。

```
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 (0.5-17.693) converged normally after 57 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：

```

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 `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' に固定されるが、両母集団とも、`x2` の因子負荷量は '0.5' に固定される。`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。こうした修飾子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 (0.5-17.693) converged normally after 48 iterations

Number of observations per group	
Pasteur	156
Grant-White	145
Estimator	
	ML
Minimum Function Test Statistic	118.976
Degrees of freedom	52
P-value (Chi-square)	0.000

Chi-square for each group:

Pasteur	64.901
Grant-White	54.075

Parameter estimates:

Information	Expected
Standard Errors	Standard

Group 1 [Pasteur]:

	Estimate	Std.err	Z-value	P(> z)
Latent variables:				
visual =~				
x1	1.000			
x2	0.500			
x3	0.600			
textual =~				
x4	1.000			
x5	1.185	0.102	11.598	0.000
x6 (a)	0.876	0.077	11.409	0.000
speed =~				
x7	1.000			
x8	1.129	0.279	4.055	0.000
x9	0.931	0.227	4.103	0.000

Covariances:

visual ~~				
textual	0.460	0.103	4.479	0.000
speed	0.182	0.076	2.408	0.016
textual ~~				
speed	0.181	0.069	2.625	0.009

Intercepts:

x1	4.941	0.094	52.379	0.000
x2	5.984	0.100	59.945	0.000
x3	2.487	0.092	26.983	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	0.388	0.129
x2	1.304	0.155
x3	0.965	0.120
x4	0.427	0.069
x5	0.454	0.086
x6	0.289	0.050
x7	0.824	0.124
x8	0.510	0.116
x9	0.677	0.105
visual	1.001	0.172
textual	0.892	0.150
speed	0.346	0.125

Group 2 [Grant-White]:

	Estimate	Std.err	Z-value	P(> z)
Latent variables:				
visual =~				
x1	1.000			
x2	0.500			
x3	0.800			
textual =~				
x4	1.000			
x5	0.990	0.087	11.425	0.000
x6	0.963	0.085	11.374	0.000
speed =~				
x7	1.000			
x8	1.228	0.188	6.539	0.000
x9	1.081	0.168	6.417	0.000

```

Covariances:
  visual ~~
    textual      0.454    0.099    4.585    0.000
    speed        0.315    0.079    4.004    0.000
  textual ~~
    speed        0.222    0.073    3.049    0.002

```

```

Intercepts:
  x1      4.930    0.097    50.688    0.000
  x2      6.200    0.089    69.616    0.000
  x3      1.996    0.086    23.223    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      0.637    0.115
  x2      0.966    0.120
  x3      0.601    0.091
  x4      0.316    0.065
  x5      0.418    0.072
  x6      0.407    0.069
  x7      0.609    0.091
  x8      0.411    0.094
  x9      0.522    0.089
  visual  0.735    0.132
  textual 0.942    0.152
  speed   0.453    0.117

```

必ずしも全てではない複数の母集団でパラメータが等しいと制約する。パラメータの値を特定の母集団を除いた全ての母集団で等しいと制約したいときもある。特定のグループでは、そのパラメータの値を自由に推定する。このパラメータの modifier は、各グループのパラメータに対しては固定された値、特定の（複数でもよい）母集団に対しては NA を含むベクトルとする。例として、4つの母集団があるとする。3つの指標を持つ潜在変数（fとする）を定義する。第2母手段を除く全ての母集団に対して、指標 item2 の因子負荷量を 1.0 に固定したいとする。このとき、次のようにする。

```
f = ~ item1 + c(1, NA, 1, 1)*item2 + item3
```

複数の母集団で 1つのパラメータが等しいと制約する。母集団間で 1つのパラメータを等しくしたいとき、同じラベルを与えればよい。例えば、(2つの)母集団間で指標 x3 の因子負荷量を等しくしたい

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 '
```

ここでも、同じラベルは、母集団内および母集団間で同一のパラメータであることを意味する。

複数の母集団間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が等しいという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の場合には、より便利な方法がある。このような制約を‘母集団等値制約’といい、適合関数の呼び出しで引数 `group.equal` 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 (0.5-17.693) converged normally after 41 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   71.088    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 '母集団等値制約' 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

引数 `group.equal` を省略すると、全てのパラメータは各母集団で自由に推定される。

しかし、自由パラメータとしたいものいくつかを除いて、パラメータのグループ全体(例えば、因子負荷量と切片)を母集団間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` がある。これを使って、自由としたいパラメータの名前を指定する。例えば：

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

測定の不変性 いくつかの母集団で、CFA モデルの測定の不変性の検定に関心があるとき、`measurementInvariance()` 関数を利用することができる。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。

9 成長曲線モデル

別の種類の重要な潜在変数モデルとして、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：

```
# 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 (0.5-13) 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:

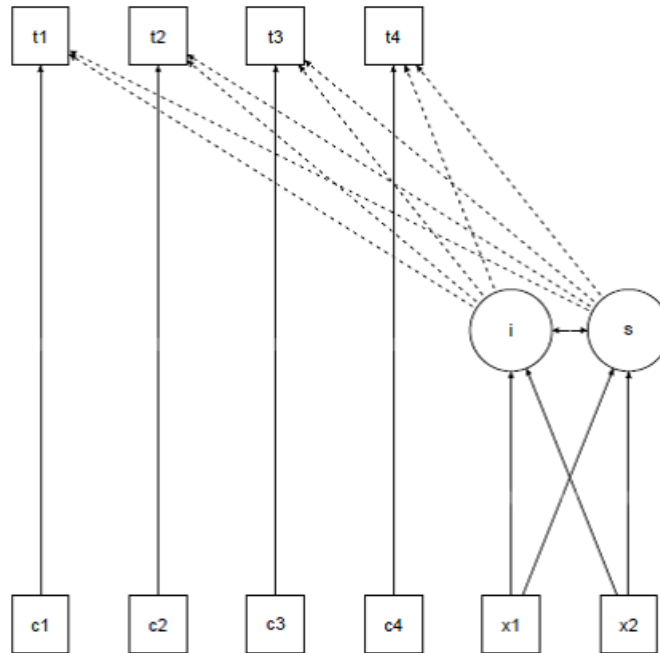
Information	Expected
Standard Errors	Standard

	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

関数 `growth()` は、技術的には関数 `sem()` 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。



この図に対応する構文は次である。

```
# 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
```

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。

```

# 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)

```

10 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。

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, ...）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。

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`）が順序尺度を持つと宣言するとき、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 に切り替える: 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。

11 共分散行列を入力として利用

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。まず, 共分散行列の下三角行列 (対角要素を含む) を読み込む。

```
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 '

wheaton.cov <-
  getCov(lower, names = c("anomia67", "powerless67",
                        "anomia71", "powerless71",
                        "education", "sei"))
```

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), 関数 `getCov()` を用いると, (変数名を含む) 完全な共分散行列を簡単に作ることができる。共分散行列の下三角部分は 2 つの一重引用符で囲まれていることに注意。これにより, さらなる軟性を得ることになる。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。

```

# 古典的 Wheaton et al モデル
wheaton.model <- '
  # 潜在変数
  ses =~ education + sei
  alien67 =~ anomia67 + powerless67
  alien71 =~ anomia71 + powerless71

  # 回帰
  alien71 ~ alien67 + ses
  alien67 ~ ses

  # 相関を持つ残差
  anomia67 ~~ anomia71
  powerless67 ~~ powerless71
,

fit <- sem(wheaton.model,
           sample.cov=wheaton.cov,
           sample.nobs=932)
summary(fit, standardized=TRUE)

```

lavaan (0.5-13) converged normally after 82 iterations

Number of observations	932
Estimator	ML
Minimum Function Test Statistic	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

複数の母集団があるとき、`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。`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。

12 推定量, 標準誤差, 欠測値

推定量 全てのデータが連続型のとき、`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。
- "DWLS" : 対角重み付き最小 2 乗法
- "ULS" : 重み付けなし最小 2 乗法

多くの推定量はロバストバージョンを持つ。これは、ロバストな標準誤差と尺度変換された検定統計量を提供することを意味する。例えば、最尤推定量に対して、`lavaan`は次のロバストバージョンを提供する。

- "MLM" : ロバスト標準誤差と Satorra-Bentler 尺度変換検定統計量を持つ最尤法。完全データに対してのみ利用可。
- "MLMVS" : ロバストな標準誤差と、平均一分散で調整された検定統計量 (Satterthwaite の方法として知られる) を持つ最尤推定
- "MLMV" : ロバストな標準誤差と、平均一分散で調整された検定統計量を持つ最尤推定 (scale-shifted 法を利用)。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.

lavaanは DWLS と ULS に対してもロバストバージョンである WLSM, WLSMVS, WLSMV, ULSM, ULSMVS, ULSMV を持つ. ロバストな WLS バージョンにおいて, 推定に対して重み付け行列の対角成分を利用するが, 標準誤差を修正し, 検定統計量を計算するために完全な重み付け行列を利用することに注意.

最尤推定: ウィシャート (Wishert) 対正規 最尤法 ("ML" またはそのロバスト版) 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 (要素を $n-1$ の代わりに n で割ったもの) 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, 最小の閾値に因子 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 プログラムに似ている. 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, 適合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 (0.5-18) converged normally after 35 iterations

Number of observations	301
Estimator	ML
Minimum Function Test Statistic	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.

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単位 (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, この特性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.

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, 変更することができる.

推定量が単に "ML" のとき, `se` 引数を用いて "robust.sem" や "robust.huber.white", "first.order", "bootstrap" 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、"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", "Satorra-Bentler", "Yuan-Bentler", "bootstrap"である。

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="bootstrap"またはtest="bootstrap"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、任意の統計量（あるいは、統計量のベクトル）をブートストラップすることに利用できるが、既に当てはめたlavaanオブジェクトが必要である。

13 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 (0.5-13) 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)	0.000

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、いわゆる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。

14 修正指標

修正指標は、`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。関数 `modindices()` は、データフレームを返すので、それをソートしたり、必要な情報を抽出したりすることができる。例えば、因子負荷量に関する修正指標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

```

修正指標は、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。修正指標は、期待されるパラメータ変化（EPC）の値（epc 列）という補足情報を持つ。最後の 3 列は、それぞれ標準化された EPC 値である（sepc.lv：潜在変数を標準化したもの；sepc.all：全ての変数を標準化したもの；sepc.nox：外生観測変数以外全てを標準化したもの）。

15 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情報が必要なときには、‘抽出’のための関数を用いるとよい。当てはめたモデルの母数の推定値を抽出するための関数 coef() についてはすでに見た。

parameterEstimates 関数 parameterEstimates() 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値、返された値をデータフレームとして抽出する。例えば：

```

fit <- cfa(HS.model, data = HolzingerSwineford1939)
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	0.554	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

standardizedSolution 関数 `standardizedSolution()` は、関数 `parameterEstimates()` 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。

fitted.values 関数 `fitted()` と `fitted.values()` は、当てはめたモデルの暗黙の（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
```

残差 関数 `resid()` または `residuals()` 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 残差を得ることができる（注意：NA が表示される可能性があるが、無視してよい）。

```
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  0.000
x3 -1.199  2.692  0.000
x4  2.465 -0.283 -1.948  0.000
x5 -0.362 -0.610 -4.443  0.856  0.000
x6  2.032  0.661 -0.701    NA  0.633  0.000
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  0.000
x9  4.062  1.517  3.328  1.237  1.723  1.436 -2.771    NA  0.000

```

```

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

```

vcov 関数 `vcov()` は、推定されたパラメータの共分散行列の推定値を返す。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。

fitMeasures 関数 `fitMeasures()` は、lavaanにより計算された適合の測度すべてを名前付き数値ベクトルとして返す。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)

```

fmin	chisq	df	pvalue
0.142	85.306	24.000	0.000
baseline.chisq	baseline.df	baseline.pvalue	cfi
918.852	36.000	0.000	0.931
tli	nnfi	rfi	nfi
0.896	0.896	0.861	0.907
pnfi	ifi	rni	logl
0.605	0.931	0.931	-3737.745
unrestricted.logl	npar	aic	bic
-3695.092	21.000	7517.490	7595.339
ntotal	bic2	rmsea	rmsea.ci.lower
301.000	7528.739	0.092	0.071
rmsea.ci.upper	rmsea.pvalue	rmr	rmr_nomean
0.114	0.001	0.082	0.082
srmr	srmr_bentler	srmr_bentler_nomean	srmr_bollen
0.065	0.065	0.065	0.065
srmr_bollen_nomean	srmr_mplus	srmr_mplus_nomean	cn_05
0.065	0.065	0.065	129.490
cn_01	gfi	agfi	pgfi
152.654	0.943	0.894	0.503
mfi	ecvi		
0.903	0.423		

1つの当てはめの測度のみが必要な場合（例えば、CFI）、第2の引数としてその名前を与える（小文字で）。

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

```
cfi
0.931
```

当てはめの測度のベクトルを次のようにして与えることもできる。

```
fit <- cfa(HS.model, data = HolzingerSwineford1939)
fitMeasures(fit, c("cfi", "rmsea", "srmr"))
```

```
cfi rmsea srmr
0.931 0.092 0.065
```

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。デフォルトでは、当てはめた **lavaan** オブジェクトに対して `inspect()` を適用すると、モデルを表現するのに内的に利用されたモデルのメトリックスのリストを返す。自由パラメータはゼロではない整数である。

```
fit <- cfa(HS.model, data = HolzingerSwineford1939)
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 textual 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 textual 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 textual 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を知るには、次を入力する.

```
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

これは、`parTable(fit)` 関数と同じ機能を持つ。ここで返された表を、‘パラメータ表’という。

`inspect` 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
```