lavaan: internals

Yves Rosseel
Department of Data Analysis
Ghent University – Belgium

Utrecht – April 24, 2012

Overview

1. model, data, and discrepancy functions
2. from model syntax to model matrices
3. the lavaan class for representing a fitted model
4. how to extend lavaan

Model fitting: model + data + discrepancy function

The model

- the user needs to be able to specify/describe his/her model
- the default approach: lavaan model syntax
- internal representation: the **parameter table**
  - complete description of the model
  - does not depend on the traditional 'matrix' representation (LISREL, Bentler-Weeks, RAM, ...)
  - computing on the model
- future plans:
  - specify model matrices directly
  - read/parse/write LISREL/EQS/Mplus syntax
  - graphical/GUI interface

The data

- default input: a regular R data.frame
  - @Data slot describes the data: which observed variables are used, number of groups, missing data, ...
  - @Data@X contains a copy of the raw data (a list of matrices; one per group)
- alternative input: sample statistics (covariance matrix, mean vector)
  - no robust standard errors, no scaled test statistics
  - no missing data
- partially implemented: simulated data
  - `simulateData()`
  - `bootstrapLavaan()`

The discrepancy function

- currently all discrepancy function are hard-wired in the code
- focus on covariance structure and mean structure analysis
  - traditional ML

$$F_{ML}(\boldsymbol{\theta}) = \log|\boldsymbol{\Sigma}| + \mathsf{tr}(\mathbf{S}\boldsymbol{\Sigma}^{-1}) - \log|\mathbf{S}| - p$$

  - FIML (direct ML, missing data, pattern-based)
  - least-squares: GLS, WLS
- future plans:
  - user-specified discrepancy functions
  - not only covariance/mean structure analysis, but any arbitrary model fitting function

The optimizer

- default without constraints:
  - `nlminb` (builtin R optimizer)
  - quasi-newton with analytical gradients
  - manual parameter scaling
- with constraints:
  - `nlminb.constr` (lavaan specific)
  - augmented Lagrangian multiplier method
  - slow for simple problems
  - we need better strategies for specific cases
- we need better optimizers (in R)!

From model syntax to model matrices

The model syntax

```
HS.model <- '
    # some random comment
    visual  =~ x1 + a*x2 + a*x3
    textual =~ x4 + 0.8*x5 + 1.2*x6

    # some blank lines
    speed   =~ x7 + start(0.5)*x8 + x9
'
```

The lavaanify() function

1. phase I: parses the model syntax and 'flattens' the model equations
2. phase II: creates a complete parameter table, representing the model as specified by the user

phase I: parseModelString()

```
FLAT <- parseModelString(HS.model)
as.data.frame(FLAT)
```

```
      lhs op rhs mod.idx group fixed start label
1  visual =~  x1       0     0                  
2  visual =~  x2       1     0                 a
3  visual =~  x3       2     0                 a
4 textual =~  x4       0     0                  
5 textual =~  x5       3     0   0.8            
6 textual =~  x6       4     0   1.2            
7   speed =~  x7       0     0                  
8   speed =~  x8       5     0         0.5      
9   speed =~  x9       0     0                  
```

phase II: lavaanify()

```
ParTable <- lavaanify(FLAT, auto.var=TRUE, auto.cov.lv.x=TRUE)
ParTable
```

The parameter table

	id	lhs	op	rhs	user	group	free	ustart	exo	label	eq.id	unco
1	1	visual	=~	x1	1	1	1	NA	0		0	1
2	2	visual	=~	x2	1	1	2	NA	0	a	2	2
3	3	visual	=~	x3	1	1	2	NA	0	a	2	3
4	4	textual	=~	x4	1	1	3	NA	0		0	4
5	5	textual	=~	x5	1	1	0	0.8	0		0	0
6	6	textual	=~	x6	1	1	0	1.2	0		0	0
7	7	speed	=~	x7	1	1	4	NA	0		0	5
8	8	speed	=~	x8	1	1	5	0.5	0		0	6
9	9	speed	=~	x9	1	1	6	NA	0		0	7
10	10	x1	~~	x1	0	1	7	NA	0		0	8
11	11	x2	~~	x2	0	1	8	NA	0		0	9
12	12	x3	~~	x3	0	1	9	NA	0		0	10
13	13	x4	~~	x4	0	1	10	NA	0		0	11
14	14	x5	~~	x5	0	1	11	NA	0		0	12
15	15	x6	~~	x6	0	1	12	NA	0		0	13
16	16	x7	~~	x7	0	1	13	NA	0		0	14
17	17	x8	~~	x8	0	1	14	NA	0		0	15
18	18	x9	~~	x9	0	1	15	NA	0		0	16
19	19	visual	~~	visual	0	1	16	NA	0		0	17
20	20	textual	~~	textual	0	1	17	NA	0		0	18
21	21	speed	~~	speed	0	1	18	NA	0		0	19
22	22	visual	~~	textual	0	1	19	NA	0		0	20
23	23	visual	~~	speed	0	1	20	NA	0		0	21
24	24	textual	~~	speed	0	1	21	NA	0		0	22

lavaanify() arguments

```
lavaanify(model = NULL, meanstructure = FALSE, int.ov.free = FALSE,
    int.lv.free = FALSE, orthogonal = FALSE, std.lv = FALSE,
    fixed.x = TRUE, constraints = NULL, auto = FALSE, model.type = "sem
    auto.fix.first = FALSE, auto.fix.single = FALSE, auto.var = FALSE,
    auto.cov.lv.x = FALSE, auto.cov.y = FALSE, ngroups = 1L,
    group.equal = NULL, group.partial = NULL, debug = FALSE,
    warn = TRUE, as.data.frame. = TRUE)
```

extracting variable names from a parameter table

```
# observed variables
lavaanNames(ParTable, type="ov")
```

```
[1] "x1" "x2" "x3" "x4" "x5" "x6" "x7" "x8" "x9"
```

```
# latent variables
lavaanNames(ParTable, type="lv")
```

```
[1] "visual"  "textual" "speed"
```

Computing on the parameter table

- get number of free parameters

```
> lavaan:::getNPAR(ParTable)
[1] 21
```

- get number of datapoints

```
> lavaan:::getNDAT(ParTable)
[1] 45
```

- get degrees of freedom

```
> lavaan:::getDF(ParTable)
[1] 24
```

- check if the model is identified
- plot the model
- export the model to LISREL/EQS/Mplus/... syntax
- ...

From parameter table to model matrices

```
LisrelMM <- as.data.frame(lavaan:::representation.LISREL(ParTable))
cbind(ParTable[,1:7], LisrelMM)
```

	id	lhs	op	rhs	user	group	free	mat	row	col
1	1	visual	=~	x1	1	1	1	lambda	1	1
2	2	visual	=~	x2	1	1	2	lambda	2	1
3	3	visual	=~	x3	1	1	2	lambda	3	1
4	4	textual	=~	x4	1	1	3	lambda	4	2
5	5	textual	=~	x5	1	1	0	lambda	5	2
6	6	textual	=~	x6	1	1	0	lambda	6	2
7	7	speed	=~	x7	1	1	4	lambda	7	3
8	8	speed	=~	x8	1	1	5	lambda	8	3
9	9	speed	=~	x9	1	1	6	lambda	9	3
10	10	x1	~~	x1	0	1	7	theta	1	1
11	11	x2	~~	x2	0	1	8	theta	2	2
12	12	x3	~~	x3	0	1	9	theta	3	3
13	13	x4	~~	x4	0	1	10	theta	4	4
14	14	x5	~~	x5	0	1	11	theta	5	5
15	15	x6	~~	x6	0	1	12	theta	6	6
16	16	x7	~~	x7	0	1	13	theta	7	7
17	17	x8	~~	x8	0	1	14	theta	8	8
18	18	x9	~~	x9	0	1	15	theta	9	9
19	19	visual	~~	visual	0	1	16	psi	1	1
20	20	textual	~~	textual	0	1	17	psi	2	2
21	21	speed	~~	speed	0	1	18	psi	3	3

...

Model matrices: free parameters

```
$lambda
   visual textul speed
x1      0      0     0
x2      1      0     0
x3      1      0     0
x4      0      0     0
x5      0      0     0
x6      0      0     0
x7      0      0     0
x8      0      0     2
x9      0      0     3

$theta
   x1 x2 x3 x4 x5 x6 x7 x8 x9
x1  4
x2  0  5
x3  0  0  6
x4  0  0  0  7
x5  0  0  0  0  8
x6  0  0  0  0  0  9
x7  0  0  0  0  0  0 10
x8  0  0  0  0  0  0  0 11
x9  0  0  0  0  0  0  0  0 12
```

```
$psi
        visual textul speed
visual  13
textual 16     14
speed   17     18     15
```

- version 0.4-13: no self-contained function (yet!)

Model matrices: starting values

```
$lambda
   visual textul speed
x1      1    0.0   0.0
x2      1    0.0   0.0
x3      1    0.0   0.0
x4      0    1.0   0.0
x5      0    0.8   0.0
x6      0    1.2   0.0
x7      0    0.0   1.0
x8      0    0.0   0.5
x9      0    0.0   1.0
```

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

```
$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
```

From model matrices to a parameter vector . . .

- the optimizer needs a parameter vector (not a matrix list)
- we extract the 'free' parameters from each matrix and place the unduplicated ones in a vector

```
> fit@Model@x.free.idx
[[1]]
[1] 1 1 2 3

[[2]]
[1]  4  5  6  7  8  9 10 11 12

[[3]]
[1] 13 16 17 16 14 18 17 18 15
```

- internal function:

```
> x <- lavaan:::getModelParameters(Model)
> x
 [1] 1.0000000 0.5000000 1.0000000 0.6791849 0.6908919 0.6374324
 [7] 0.6753323 0.8298929 0.5981792 0.5915697 0.5109914 0.5075019
[13] 0.0500000 0.0500000 0.0500000 0.0000000 0.0000000 0.0000000
```

. . . and back

- each iteration, we need to (re)compute the model-implied covariance matrix

```
> fit@Model@m.free.idx
[[1]]
[1]  2  3 26 27

[[2]]
[1]  1 11 21 31 41 51 61 71 81

[[3]]
[1] 1 2 3 4 5 6 7 8 9
```

- objective function (simplified):

```
minimize.this.function <- function(x) {

    # update model matrices
    GLIST <- x2GLIST(object, x=x)

    # compute function value
    fx <- computeObjective(object, GLIST=GLIST, sample, ...)

    fx
}
```

The 'lavaan' class

- all fitting functions (`sem`, `cfa`, `growth`, `lavaan`) create an object of class 'lavaan'
- lavaan 0.4-13 currently uses S4 classes
- future releases may replace this partially/entirely with reference classes
- many methods for this 'lavaan' class exist; see `class?lavaan` for an overview
- a 'lavaan' object contains the following slots:

```
> example(cfa)
> slotNames(fit)
[1] "call"    "timing"  "Options" "User"    "Data"    "Sample"
[7] "Model"   "Fit"
```

- warning: the names are likely to change in the (near) future

slots `@call` and `@timing`

- `@call` contains the function call as returned by `match.called()`

```
lavaan(model = HS.model, model.type = "cfa", int.ov.free = TRUE,
    int.lv.free = FALSE, auto.fix.first = TRUE, auto.fix.single =
    auto.var = TRUE, auto.cov.lv.x = TRUE, auto.cov.y = TRUE,
    data = HolzingerSwineford1939)
```

- `@timing` contains the elapsed time (user+system) for various parts of the program as a list, including the total time

```
> unlist(fit@timing)
InitOptions.elapsed    InitData.elapsed        User.elapsed
              0.002               0.002               0.003
     Sample.elapsed       Start.elapsed       Model.elapsed
              0.002               0.001               0.003
   Estimate.elapsed        VCOV.elapsed        TEST.elapsed
              0.036               0.002               0.000
      total.elapsed
              0.054
```

slot `@Options`

- program-wide options
- in future releases, we may remove this slot
- example:

```
              model         model.type   meanstructure      int.ov.free
      "    ...    "              "cfa"         "FALSE"           "TRUE"
      int.lv.free          fixed.x      orthogonal          std.lv
          "FALSE"            "TRUE"         "FALSE"          "FALSE"
  auto.fix.first  auto.fix.single        auto.var    auto.cov.lv.x
          "TRUE"            "TRUE"         "TRUE"           "TRUE"
       auto.cov.y           std.ov        missing     constraints
          "TRUE"           "FALSE"      "listwise"              ""
        estimator      likelihood     information              se
            "ML"         "normal"      "expected"      "standard"
            test        bootstrap           mimic  representation
       "standard"           "1000"       "lavaan"         "LISREL"
          do.fit          verbose            warn           debug
          "TRUE"           "FALSE"         "TRUE"          "FALSE"
       data.type
          "full"
```

slot `@User`

- the parameter table
- the name refers to the 'user'-specified model
- is a list, but can be coerced to a data.frame

```
$id
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
[23] 23 24

$lhs
 [1] "visual"  "visual"  "visual"  "textual" "textual" "textual"
 [7] "speed"   "speed"   "speed"   "x1"      "x2"      "x3"
[13] "x4"      "x5"      "x6"      "x7"      "x8"      "x9"
[19] "visual"  "textual" "speed"   "visual"  "visual"  "textual"

$op
 [1] "=~" "=~" "=~" "=~" "=~" "=~" "=~" "=~" "=~" "~~" "~~" "~~" "~~"
[14] "~~" "~~" "~~" "~~" "~~" "~~" "~~" "~~" "~~" "~~" "~~"
...
```

slot @Data

- a description of the data

```
Formal class 'lavaanData' [package "lavaan"] with 9 slots
  ..@ ngroups    : int 1
  ..@ group.label: chr(0)
  ..@ nobs       :List of 1
  .. ..$ : int 301
  ..@ norig      :List of 1
  .. ..$ : int 301
  ..@ ov.names   :List of 1
  .. ..$ : chr [1:9] "x1" "x2" "x3" "x4" ...
  ..@ ov.idx     :List of 1
  .. ..$ : int [1:9] 7 8 9 10 11 12 13 14 15
  ..@ case.idx   :List of 1
  .. ..$ : int [1:301] 1 2 3 4 5 6 7 8 9 10 ...
  ..@ Mp         :List of 1
  .. ..$ : NULL
  ..@ X          :List of 1
  .. ..$ : num [1:301, 1:9] 3.33 5.33 4.5 5.33 4.83 ...
```

slot `@Sample`: sample statistics

```
Formal class 'SampleStats' [package "lavaan"] with 12 slots
  ..@ cov          :List of 1
  .. ..$ : num [1:9, 1:9] 1.358 0.407 0.58 0.505 0.441 ...
  ..@ mean         :List of 1
  .. ..$ : num [1:9] 4.94 6.09 2.25 3.06 4.34 ...
  ..@ nobs         :List of 1
  .. ..$ : int 301
  ..@ ntotal       : int 301
  ..@ ngroups      : int 1
  ..@ icov         :List of 1
  .. ..$ : num [1:9, 1:9] 1.1474 -0.1051 -0.3267 -0.2723 0.0344 ...
  ..@ cov.log.det :List of 1
  .. ..$ : num -0.989
  ..@ cov.vecs     :List of 1
  .. ..$ : num [1:45] 1.358 0.407 0.58 0.505 0.441 ...
  ..@ WLS.V        :List of 1
  .. ..$ : NULL
  ..@ missing.flag: logi FALSE
  ..@ missing      :List of 1
  .. ..$ : NULL
  ..@ missing.h1   :List of 1
  .. ..$ : NULL
```

slot `@Model`: model matrices and more

```
Formal class 'Model' [package "lavaan"] with 30 slots
  ..@ GLIST            :List of 3
  .. ..$ lambda: num [1:9, 1:3] 1 0.553 0.729 0 0 ...
  .. ..$ theta : num [1:9, 1:9] 0.549 0 0 0 0 ...
  .. ..$ psi   : num [1:3, 1:3] 0.809 0.408 0.262 0.408 0.979 ...
  ..@ dimNames         :List of 3
  .. ..$ :List of 2
  .. .. ..$ : chr [1:9] "x1" "x2" "x3" "x4" ...
  .. .. ..$ : chr [1:3] "visual" "textual" "speed"
  .. ..$ :List of 2
  .. .. ..$ : chr [1:9] "x1" "x2" "x3" "x4" ...
  .. .. ..$ : chr [1:9] "x1" "x2" "x3" "x4" ...
  .. ..$ :List of 2
  .. .. ..$ : chr [1:3] "visual" "textual" "speed"
  .. .. ..$ : chr [1:3] "visual" "textual" "speed"
  ..@ isSymmetric      : logi [1:3] FALSE TRUE TRUE
  ..@ mmSize           : int [1:3] 27 45 6
  ..@ representation   : chr "LISREL"
  ..@ meanstructure    : logi FALSE
  ..@ ngroups          : int 1
  ..@ nmat             : int 3
  ..@ nvar             : int 9
```

```
..@ nx.free          : int 21
..@ nx.unco          : int 21
..@ nx.user          : int 24
..@ m.free.idx       :List of 3
.. ..$ : int [1:6] 2 3 14 15 26 27
.. ..$ : int [1:9] 1 11 21 31 41 51 61 71 81
.. ..$ : int [1:9] 1 2 3 4 5 6 7 8 9
..@ x.free.idx       :List of 3
.. ..$ : int [1:6] 1 2 3 4 5 6
.. ..$ : int [1:9] 7 8 9 10 11 12 13 14 15
.. ..$ : int [1:9] 16 19 20 19 17 21 20 21 18
..@ m.unco.idx       :List of 3
.. ..$ : int [1:6] 2 3 14 15 26 27
.. ..$ : int [1:9] 1 11 21 31 41 51 61 71 81
.. ..$ : int [1:9] 1 2 3 4 5 6 7 8 9
..@ x.unco.idx       :List of 3
.. ..$ : int [1:6] 1 2 3 4 5 6
.. ..$ : int [1:9] 7 8 9 10 11 12 13 14 15
.. ..$ : int [1:9] 16 19 20 19 17 21 20 21 18
..@ m.user.idx       :List of 3
.. ..$ : int [1:9] 1 2 3 13 14 15 25 26 27
.. ..$ : int [1:9] 1 11 21 31 41 51 61 71 81
.. ..$ : int [1:9] 1 2 3 4 5 6 7 8 9
```

```
..@ x.user.idx       :List of 3
.. ..$ : int [1:9] 1 2 3 4 5 6 7 8 9
.. ..$ : int [1:9] 10 11 12 13 14 15 16 17 18
.. ..$ : int [1:9] 19 22 23 22 20 24 23 24 21
..@ x.def.idx        : int(0)
..@ x.ceq.idx        : int(0)
..@ x.cin.idx        : int(0)
..@ eq.constraints   : logi FALSE
..@ eq.constraints.K: num[0 , 0 ]
..@ def.function     :function ()
..@ ceq.function     :function ()
..@ ceq.jacobian     :function ()
..@ cin.function     :function ()
..@ cin.jacobian     :function ()
..@ con.jac          : num[0 , 0 ]
..@ fixed.x          : logi FALSE
```

slot `@Fit`: model fit results

```
Formal class 'Fit' [package "lavaan"] with 13 slots
  ..@ npar      : int 21
  ..@ x         : num [1:21] 0.553 0.729 1.113 0.926 1.18 ...
  ..@ start     : num [1:24] 1 0.778 1.107 1 1.133 ...
  ..@ est       : num [1:24] 1 0.553 0.729 1 1.113 ...
  ..@ se        : num [1:24] 0 0.0997 0.1091 0 0.0654 ...
  ..@ fx        : num 0.142
  ..@ fx.group  : num 0.142
  ..@ iterations: int 41
  ..@ converged : logi TRUE
  ..@ control   :List of 7
  .. ..$ eval.max: int 20000
  .. ..$ iter.max: int 10000
  .. ..$ trace   : int 0
  .. ..$ abs.tol : num 2.22e-15
  .. ..$ rel.tol : num 1e-10
  .. ..$ x.tol   : num 1.5e-08
  .. ..$ step.min: num 2.2e-14
  ..@ Sigma.hat :List of 1
  .. ..$ : num [1:9, 1:9] 1.358 0.448 0.59 0.408 0.454 ...
  ..@ Mu.hat    :List of 1
  .. ..$ : num [1:9] 0 0 0 0 0 0 0 0 0
```

```
..@ test      :List of 1
.. ..$ :List of 5
.. .. ..$ test      : chr "standard"
.. .. ..$ stat      : num 85.3
.. .. ..$ stat.group: num 85.3
.. .. ..$ df        : int 24
.. .. ..$ pvalue    : num 8.5e-09
```

How to extend lavaan?

Some general guidelines

- does your code need to go into lavaan, or in a separate package?
  - soon on CRAN: the `semTools` package
- use github!
  - `https://github.com/yrosseel/lavaan/`
- try to use 'accessor' functions (e.g. `coef`, `inspect()`,...) to 'extract' information from a fitted lavaan object
- try to avoid using internal functions that are not exported
- if you must use them, contact me
- keep your contributed code in a separate file
- will your code work with multiple groups, missing data, ...?