Additional Lavaan Options

Jim Grace

U.S. Department of the Interior
U.S. Geological Survey

In this module I provide a few illustrations of options within lavaan for handling various situations.

An appropriate citation for this material is

Yves Rosseel (2012). lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2), 1-36. URL http://www.jstatsoft.org/v48/i02/

Notes: IP-056512; Support provided by the USGS Climate & Land Use R&D and Ecosystems Programs. I would like to acknowledge formal review of this material by Jesse Miller and Phil Hahn, University of Wisconsin. Many helpful informal comments have contributed to the final version of this presentation. The use of trade names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

Last revised 17.02.15.

Source: https://www.usgs.gov/centers/wetland-and-aquatic-research-center/science/quantitative-analysis-using-structural-equation

A variety of special modeling issues are automated in lavaan.

Outline:

- Missing data
- Robust estimators for non-normal endogenous variables
- Bootstrapping
- Categorical responses
- Input data as a covariance matrix
- Simulating data

Additional capabilities of lavaan are presented in modules on other specific topics (e.g., Multi-Group Modeling and Adjusting for Nested Data).

USGS 2

This module illustrates automated procedures in lavaan that relate to some common modeling challenges.

Missing Data

3

Important issue – dealing with missing data. Traditionally "listwise deletion" is used when packages and functions encounter an empty data cell. This discards lots of information (all non-missing information in the rows deleted). Not only is this wasteful, it is also not proper because there may be particular conditions where missing cells are more likely. lavaan has simply automated procedures that use all the data even when some cells are missing.

1. Lavaan options for working with missing data.

Two degrees of assumptions about the pattern of missingness:
(a) MCAR – missing completely at random
(b) MAR – missing at random (pattern of missingness not correlated with model predictors.)

Lavaan default is listwise deletion if you do not using "missing=".

You can invoke FIML (full-information maximum likelihood) in lavaan by declaring '**missing = FIML**' in the fitting command.

```
# specific model
mod1 <- 'cover    ~ firesev
         firesev ~ age'

# Fit model
mod1.fit <- sem(mod1, data=k.dat, missing="FIML")
```

USGS

4

So-called "full-information maximum likelihood" is a very powerful option for performing analyses in the presence of missing data. It can be justified in two different situations, MCAR and MAR.

A useful source of information is

http://williammurrah.com/fiml-for-missing-data-in-lavaan-part-1-descriptive-statistics-and-correlations/

and

http://williammurrah.com/fiml-for-missing-data-in-lavaan-part-2-regression-analysis/

Results from missing="FIML".

```
> summary(mod1.fit, rsq=T)
lavaan (0.5-20) converged normally after  22 iterations

  Number of observations                            90

  Number of missing patterns                         4

  Estimator                                         ML
  Minimum Function Test Statistic                3.018
  Degrees of freedom                                 1
  P-value (Chi-square)                           0.082

Parameter Estimates:

  Information                                 Observed
  Standard Errors                             Standard
```

USGS

lavaan FIML methods first examine the patterns of missingness in the data. There is actually minimal reporting for the method (just the "Number of missing patterns" shown here).

Results from missing="FIML" (continued).

```
Regressions:
                   Estimate  Std.Err  Z-value  P(>|z|)
  cover ~
    firesev          -0.090    0.018   -4.855    0.000
  firesev ~
    age               0.057    0.012    4.568    0.000

Intercepts:
                   Estimate  Std.Err  Z-value  P(>|z|)
    cover             1.095    0.089   12.339    0.000
    firesev           3.061    0.358    8.557    0.000

Variances:
                   Estimate  Std.Err  Z-value  P(>|z|)
    cover             0.080    0.012    6.553    0.000
    firesev           2.167    0.332    6.535    0.000

R-Square:
                   Estimate
    cover             0.212
    firesev           0.194
```

Estimates are shown as usual.

Robust Estimators

USGS

7

Methods have been developed to provide estimates that are robust to deviations from the assumption of normal errors. Here we see what lavaan has to offer in that area.

1. Lavaan permits use of "robust" estimation.

Lavaan has two main options for robust estimation:

MLM – produces chi-squares and standard errors robust to non-normality. AKA the Satorra-Bentler correction.

MLR – similar to MLM, but uses the Yuan-Bentler method so that missing data can be accommodated.

see discussion in:
Yuan & Bentler. 2000. In Sobel & Becker (eds.) Sociological Methodology (pp 165-200)

USGS

8

Robust means the inferences are robust to deviations from normality in the response variables. A useful and brief overview can be found at

http://web.pdx.edu/~newsomj/semclass/ho_estimate.pdf

2. Robust estimation invoked with 'estimator =' command.

```
# create model
mod <- 'y2 ~ y1
        y1 ~ x1'

# estimate model
mod.fit <- sem(mod, data=dat, fixed.x=F,
               estimator="mlm")

# get results
summary(mod.fit)
```

"fixed.x=F" is required when using "mlm" option.

USGS

9

Declaring the estimator when fitting a lavaan model is simple.

3. Results

```
Estimator                              ML       Robust
 Chi-square                         4.213        4.082
 Degrees of freedom                     1            1
 P-value                            0.040        0.043
 Scaling correction factor
 for the Yuan-Bentler correction                 1.032
```

```
Standard Errors                            Robust.mlm

              Estimate  Std.err  Z-value  P(>|z|)
Regressions:
  y2 ~
    y1          -0.154    0.024   -6.386    0.000
  y1 ~
    x1           1.185    0.290    4.091    0.000
```

USGS

10

The results output shows the adjusted values.

Bootstrapping

USGS

11

A commonly used approach to estimating probabilities is resampling and one particularly popular form of resampling is bootstrapping (sampling with replacement).

1. Lavaan has resampling methods for non-normal data.

```
# fit model and request bootstrapped results

mod.fit <- sem(mod, dat=dat,
           test="boot", se="boot", bootstrap=200)
```

USGS

12

Bootstrapping is likewise a simple operation in lavaan.

2. Results

```
Estimator                          ML
 Chi-square                     4.213
 Degrees of freedom                 1
 P-value                        0.040
 P-value (Bollen-Stine)         0.053
```

```
Estimate  Std.err  Z-value  P(>|z|)
Regressions with Robust.mlm standard errors:
  y2.ln ~
    y1.ln      -0.154    0.024   -6.386    0.000
  y1.ln ~
    x1.ln       1.185    0.290    4.091    0.000

Regressions with bootstrapped standard errors:
  y2.ln ~
    y1.ln      -0.154    0.027   -5.750    0.000
  y1.ln ~
    x1.ln       1.185    0.423    2.800    0.005
```

We can expect bootstrapped results to give both different standard errors and p-values because it is not just and adjustment of the standard errors.

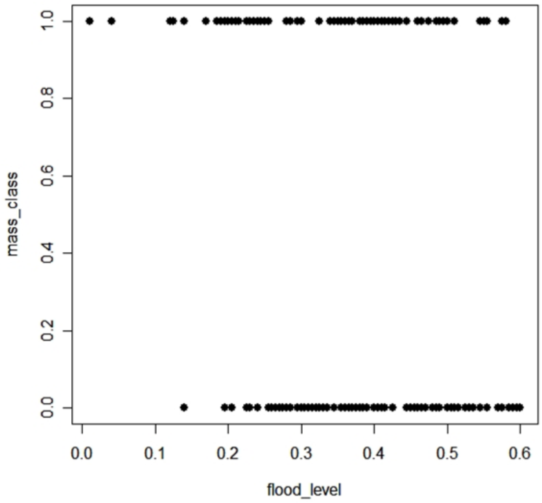
Categorical Responses

USGS

14

Another common issue is when one has response variables that are ordered categorical. In this situation, errors are generally non-normal.

The problem of analyzing categorical responses

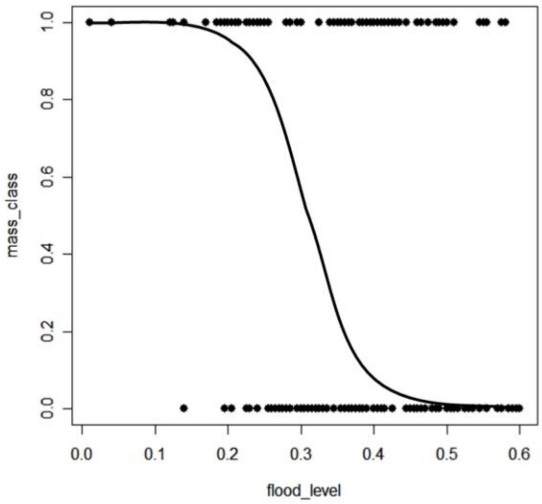


Continuous predictor (flood-level) and binary response (mass class = 0/1)

USGS

Fitting a straight line through such a set of points represents the points quite poorly and leads to illogical extrapolations, like intercepts > 1 or < 0. It also violates assumptions about normality of residuals. What we need is a way to interpret binary outcomes that makes sense. Often this is accomplished by assuming that behind the binary outcomes lies a continuous probability of observing a 1 or 0 response, as shown on the next slide.

A model for binary responses



probit and **logit** models are common response models.

Probit model:
link predictor to responses using cumulative normal probability function.

Logit model:
link predictor to responses using log transformation of ratio of probabilities of outcomes.

USGS

16

Two of the most common ways of representing the probability of observing a 1 or 0 outcome are the probit and logit models.

For the probit model, we link our predictor to our responses using a cumulative normal probability function. With the logit model, we link our predictor to our responses using an inverse log transformation of the ratio of probabilities of outcomes.

Coding lavaan for categorical responses.

```
# load data file
binary.dat <- read.csv("Pearl_BinaryResponse.csv")

# create variables (use "ordered" statement)
masscat <- ordered(binary.dat$massClass)
flood   <- binary.dat$floodLevel

mod.dat <- data.frame(flood, masscat)

# Net effect model
catmod.1 <- 'masscat ~ flood'

catmod.1.fit <- sem(catmod.1, data=mod.dat,
                    ordered="masscat")

summary(catmod.1.fit, rsq=T, standardized=T)
```

USGS

Two requirements

(1) Declare categorical variable to be “ordered” object in R.

(2) Declare variables that are ordered categorical in the “sem” statement.

Results

```
  Number of observations                              190
  Estimator                                 DWLS      Robust
  Minimum Function Test Statistic           0.000      0.000
  Degrees of freedom                        0          0
  P-value (Chi-square)                      0.000      0.000
  Scaling correction factor                            NA

Parameter estimates:
Standard Errors                              Robust.sem
                  Estimate  Std.err  Z-value  P(>|z|) Std.all
Regressions:
  masscat ~
    flood           -3.855    0.839   -4.595    0.000  -0.444


Thresholds:
    masscat|t1      -1.404    0.330   -4.262    0.000

R-square:
    masscat            0.197
```

USGS

Regression weight of -3.885 specifies the effect of one unit change in flood-level on the <u>probability</u> of observing mass_class = 1.

Error variance = 1.0 because it is set to that value to identify the model.

Rosseel gives a little more information about lavaan syntax here

http://lavaan.ugent.be/tutorial/cat.html.

He also gives more technical background at

http://www.personality-project.org/r/tutorials/summerschool.14/rosseel_sem_cat.pdf

Inputing data in the form of a covariance matrix

USGS

19

It is possible to input the covariance matrix as the data for analysis in lavaan.

Syntax for the simulateData function in lavaan can be found at www.inside-r.org/packages/cran/lavaan/docs/simulateData

```
# Create covariance matrix for input
lower <- '
1.0141
0.7177 1.5246
0.3749 0.7686 1.378
'
dat.cov <- getCov(lower, names = c("x1", "y1", "y2"))

# Analysis
mod <- 'y1 ~ x1
        y2 ~ y1
       '
fit <- sem(mod, sample.cov = dat.cov,
           sample.nobs = 10000)
summary(fit)
```

USGS

20

The "getCov" function is used to tell lavaan a covariance matrix is being used as input. Here only the lower matrix values are inputted, which implies the matrix is symmetrical.

You have to tell lavaan the sample size for estimation to take place.

Results:

```
lavaan (0.5-15) converged normally after  13 iterations

  Number of observations                          10000

Minimum Function Test Statistic                 2.556
  Degrees of freedom                                  1
  P-value (Chi-square)                            0.110

Parameter estimates:
                   Estimate  Std.err  Z-value  P(>|z|)
Regressions:
  y1 ~
    x1                0.708    0.010   70.683    0.000
  y2 ~
    y1                0.504    0.008   62.545    0.000
```

USGS Simulation input parameters recovered. 21

A full set of results can be obtained from the covariance matrix alone.

Using the lavaan "simulateData" function

22

In this module I illustrate a particular lavaan option.

An appropriate citation for this material is

Yves Rosseel (2012). lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2), 1-36. URL http://www.jstatsoft.org/v48/i02/

Notes: IP-000000; Support provided by the USGS Climate & Land Use R&D and Ecosystems Programs. I would like to acknowledge formal review of this material by XXXXX and XXXXX. The use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government. Last revised 16.02.15. Questions about this material can be sent to sem@usgs.gov.

Syntax for the simulateData function in lavaan can be found at www.inside-r.org/packages/cran/lavaan/docs/simulateData

```
library(lavaan)
# Here is a simple example to get us started:

sim.mod <- 'y2 ~ 0.5*y1
            y1 ~ 0.7*x1
           '

# generate data
set.seed(1)
sim.dat1 <- simulateData(sim.mod, sample.nobs=10000L)
```

USGS

To simulate data in lavaan, you have to provide the values for the population parameters (in red). You also have to set a seed (if you want results each time to be the same).

A "simulateData" function is evoked, along with a statement about the number of samples requested. All the examples I have found use the capital letter "L" at the end of the number of samples, but I have not found an explanation.

This produces a new data frame "sim.dat1".

Continued:

```
# fit model
test.mod <- 'y2 ~ y1
             y1 ~ x1
            '
fit <- sem(test.mod, data=sim.dat1)
summary(fit)
```

USGS

24

We can test to see what values we recover from our simulated data.

Results:

```
lavaan (0.5-15) converged normally after  11 iterations

Number of observations                          10000

Minimum Function Test Statistic                 2.551
  Degrees of freedom                                1
  P-value (Chi-square)                          0.110

Parameter estimates:
                   Estimate  Std.err  Z-value  P(>|z|)
Regressions:
  y2 ~
    y1                0.504    0.008   62.529    0.000
  y1 ~
    x1                0.708    0.010   70.683    0.000
```

Simulation input parameters recovered. 25

At this large sample size, we recover the estimated parameters.