CS6462

Probabilistic and Explainable AI

Lesson 24

Structural Equation Modeling

by Emil Vassev

March 24, 2025

Introduction to SEM

Definition & specifics:

- computational framework to model the covariance matrix of a set of observable variables
- provides separation of **structural** and **measurement** models
- enables a wide range of multivariate models with both observed and latent variables

Common methods:

- linear regression
- seemingly unrelated regression
- errors-in-variables models
- confirmatory and exploratory factor analysis
- multiple indicators multiple causes models
- instrumental variable models
- random effects models

covariance matrix: matrix that stores pairwise covariance of all jointly modeled random variables

$$cov(X_i, X_j) = \exp\left(-\frac{(X_i - X_j)^2}{2\sigma^2}\right)$$

latent variable: not directly observed but is rather inferred (through a mathematical model) from other variables that are observed

x ≈ Beta(α, β) , ***x*** is a latent variable (e.g., bias of a coin)

Introduction to SEM (cont.)

Key contribution:

- formal languages to specify: 1) models of matrices that hold the parameters θ of a machine learning problem; and 2) methods to compute the parameters’ estimates
- SEM matrix notation leads to a model-implied covariance matrix
- SEM models:
  - combinations of latent variables
  - can be extended to include the mean of observed variables (called model factors)
  - regression paths
  - factor loadings
  - parameter estimates based on data
  - popular name: “LISREL Models” LInear Structural RELations (the most popular SEM framework)*
  - visualized by a *graphical path diagram*

* Jöreskog, K. G., & Sörbom, D. (1993). LISREL 8: Structural equation modeling with the simplis command language. Scientific Software International.

SEM Structure and Steps

SEM Structure:

- *measurement model* – examines relationship between the latent variables and their measures
- *structural model* – provides the relationship between the latent variables and observed variables; used to test hypothetical dependencies based on path analysis

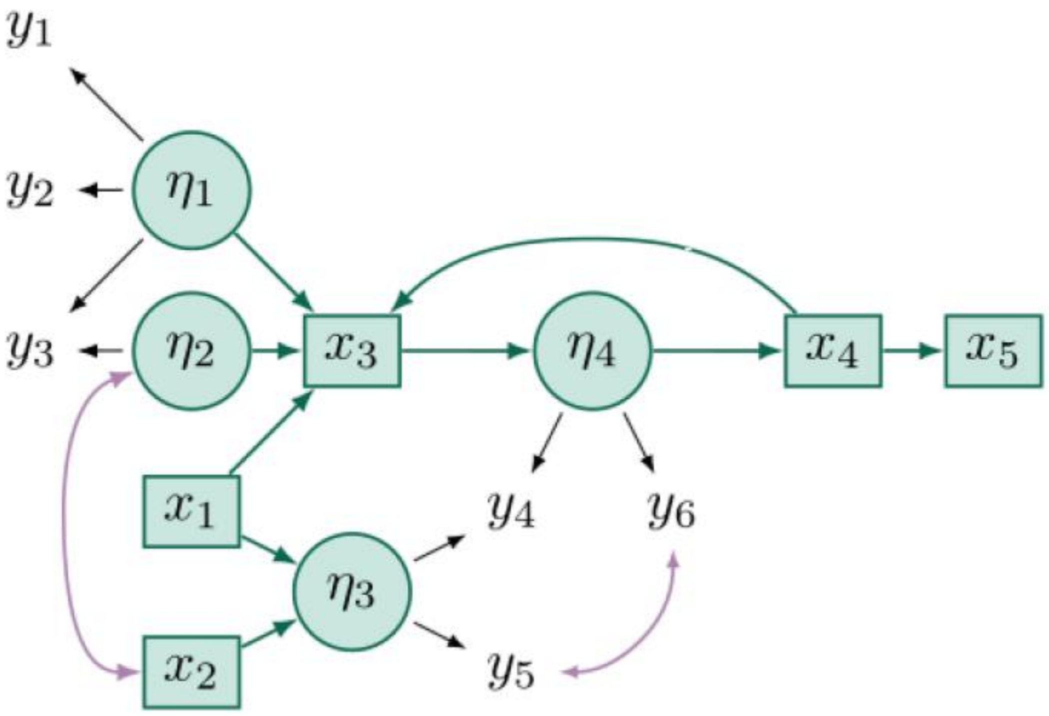
SEM logical steps:

- *model specification* – defines the hypothesized relationships among the variables
- *identification* – checks if the model is over-identified, just-identified, or under-identified
- *parameter estimation* – estimates model coefficients
- *model evaluation* – assesses model performance (e.g., a fit function), with quantitative indices calculated for the overall goodness of fit
- *model modification* – adjusts the model to improve the model fit function

SEM Graphical Path Diagram

Graphical Path Diagram: a means of SEM visualization

- easy access to the equation models (measurement and structural) of the SEM system
- flow-chart that uses arrows to show direct and indirect causal links between variables (both observable and latent):
  - rectangles for observed variables
  - ellipses for latent variables
  - curves with arrow-heads on both sides for correlations
  - straight lines with arrow-heads on one end as paths - link a predicting variable with a predicted variable



semopy.com

SEM Notation

Notation: the most common SEM notation is the LISREL notation:

$y = \Lambda\eta + \varepsilon$ – measurement model

$\eta = B_0\eta + \xi$ – structural model

$\delta = [(vec(\Lambda))^T, (vech(\psi))^T, (vec(B_0))^T, (vech(\Theta))^T]^T$ – parameter vector

y – vector of observable manifest variables

η – vector of latent variables

ε, ξ – error term vectors (ε is uncorrelated with ξ)

Λ – factor loadings (mean of observed (not manifest) variables x)

B_0 – regression parameters of the structural model

ψ – covariance matrix of ξ

Θ – covariance matrix of ε

vec – transforms a matrix into a vector by stacking the columns

$vech$ – vec + eliminates the superadiagonal elements of the matrix

measurement model: specifies linear influences of latent to observed variables
structural model: links latent variables to each other via a system of linear equations

restrictions on δ:

- subset of *free parameters* θ
- δ is identified through θ : $\delta = \delta(\theta)$

Implementing SEM in Python

Abbreviation: SEMOPY - Structural Equation Models Optimization in Python

- Python package designed to employ SEM techniques to handle SEM problems

Model Syntax: SEMOPY uses *lavaan syntax*

- supports three operator symbols characterizing relationships between variables:
  - $\sim$ to specify structural part
  - $=\sim$ to specify measurement part (reads as "measured by")
  - $\sim\sim$ to specify common variance between variables

Example:

$\eta_1 = \beta_1 x_1 + \beta_2 x_2 + \varepsilon_1$

η_1 - latent variable dependent on regressors x_1 and x_2

β_1, β_2 - parameters

ε_1 - error term

SEMOPY: structural part

eta1 ~ x1 + x2

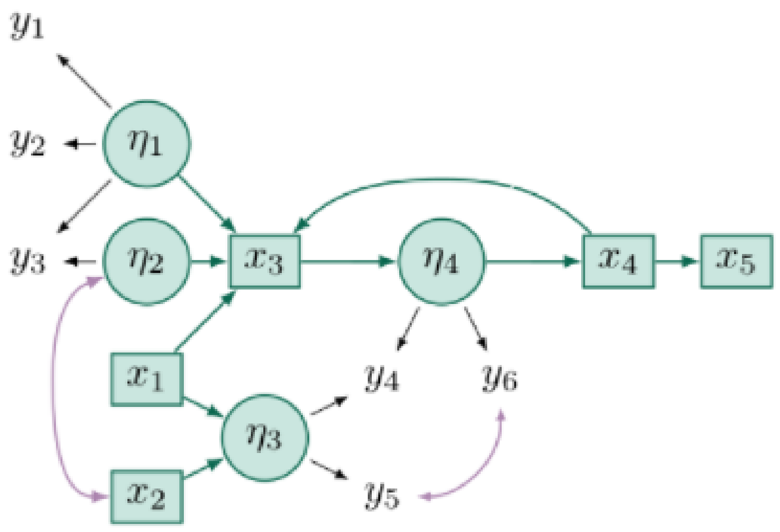
β_1, β_2 - to be estimated by SEMOPY

Implementing SEM in Python (cont.)

Example: model + path diagram

- y_k are indicators (observed) variables
- η_k are latent variables
- x_k are observed variables (but not indicators)
- unidirectional arrows are regressions
- bidirectional arrows are parameterized covariances

```
# structural part
eta3 ~ x1 + x2
eta4 ~ x3
x3 ~ eta1 + eta2 + x1 + x4
x4 ~ eta4
x5 ~ x4
# measurement part
eta1 =~ y1 + y2 + y3
eta2 =~ y3
eta3 =~ y4 + y5
eta4 =~ y4 + y6
# additional covariances
eta2 ~~  x2
y5 ~~  y6
```



semopy.com

Implementing SEM in Python (cont.)

Example: code

The pipeline for working with SEM models in SEMOPY consists of the steps:
1) specify a model
2) load a dataset
3) estimate the parameters of the model

- *Model* class – creates the SEM model:
  - structural model:
    x1 ~ x2 + x3
    x3 ~ x2 + eta1 + eta2
    eta1 ~ x1
  - measurement model:
    eta1 =~ y1 + y2 + y3
    eta2 =~ y1 + y2
- *multivariate_regression* – provides multivariate regression data

```
# SEM Example

import semopy
from semopy import Model
mod = """
x1 ~ x2 + x3
x3 ~ x2 + eta1 + eta2
eta1 ~ x1
eta1 =~ y1 + y2 + y3
eta2 =~ y1 + y2
"""
model = Model(mod)
```

```
from semopy.examples import multivariate_regression
data = multivariate_regression.get_data()
data.head()
```

	y1	y2	y3	x1	x2	x3
0	-1.989468	-0.015637	-0.162064	-0.086448	-0.728435	-0.158917
1	-0.777919	4.919949	1.222909	-0.722815	-0.552797	-2.290903
2	-1.951388	1.024939	2.402742	0.127013	0.796024	-0.040658
3	-2.084306	1.430075	-0.308923	0.317103	-0.015630	0.915825
4	1.729722	-3.632137	-2.926315	0.129123	-1.922594	1.652257

Implementing SEM in Python (cont.)

Example: code (cont,)

- *model.fit(data)* – estimates the parameters and prints the optimizer result
  - maximum likelihood - default objective function for estimating parameters
  - SLSQP (Sequential Least-Squares Quadratic Programming) – default optimization method

```
r = model.fit(data)
print(r)
```

```
Name of objective: MLW
Optimization method: SLSQP
Optimization successful.
Optimization terminated successfully
Objective value: 3.024
Number of iterations: 1
Params: -1.808 -15.623 -81.006 -75.610 -82.180 -22.486 -1.122 -1.259 -0.030
295.774 3.298 146.569 0.129 2.648 48.995 349.256
```

Implementing SEM in Python (cont.)

Example: code (cont,)

- *model.inspect()* – outputs parameter estimates in a pandas DataFrame
    - parameters are provided along with the links connecting the variables

```
print(model.inspect())
```

```
    lval  op  rval   Estimate    Std. Err   z-value   p-value
0     x1   ~    x2  -1.807823    3.573993 -0.505827  0.612978
1     x1   ~    x3 -15.622763   25.564798 -0.611104   0.54113
2     x3   ~    x2 -81.006077  400.904001 -0.202059  0.839871
3     x3   ~  eta1 -75.609574  374.889567 -0.201685  0.840163
4     x3   ~  eta2 -82.180220  407.312628 -0.201762  0.840103
5   eta1   ~    x1 -22.485946   51.028406 -0.440655  0.659462
6     y1   ~  eta1   1.000000           -         -         -
7     y1   ~  eta2   1.000000           -         -         -
8     y2   ~  eta1  -1.122261    0.313129 -3.584018  0.000338
9     y2   ~  eta2  -1.258958    0.262719 -4.792036  0.000002
10    y3   ~  eta1  -0.029972    0.068605 -0.436877    0.6622
11    x1  ~~    x1 295.774437  968.866473  0.305279  0.760154
12  eta1  ~~  eta1 146.568888  670.211267  0.218691  0.826891
13    x3  ~~    x3  48.995028 1579.803905  0.031013  0.975259
14  eta2  ~~  eta2 349.256005 1725.817667  0.202371  0.839626
15    y3  ~~    y3   3.298248     0.47176  6.991368       0.0
16    y1  ~~    y1   0.128691    0.693713   0.18551  0.852829
17    y2  ~~    y2   2.648326    0.636587  4.160198  0.000032
```

Implementing SEM in Python (cont.)

Example: code (cont,)

- *semopy.semplot()* – visualizes the Model instance
  - we can print the structure of the directed graph

```
g = semopy.semplot(model, "images/semopy_multivariate_regression_example.png")
print(g)
```

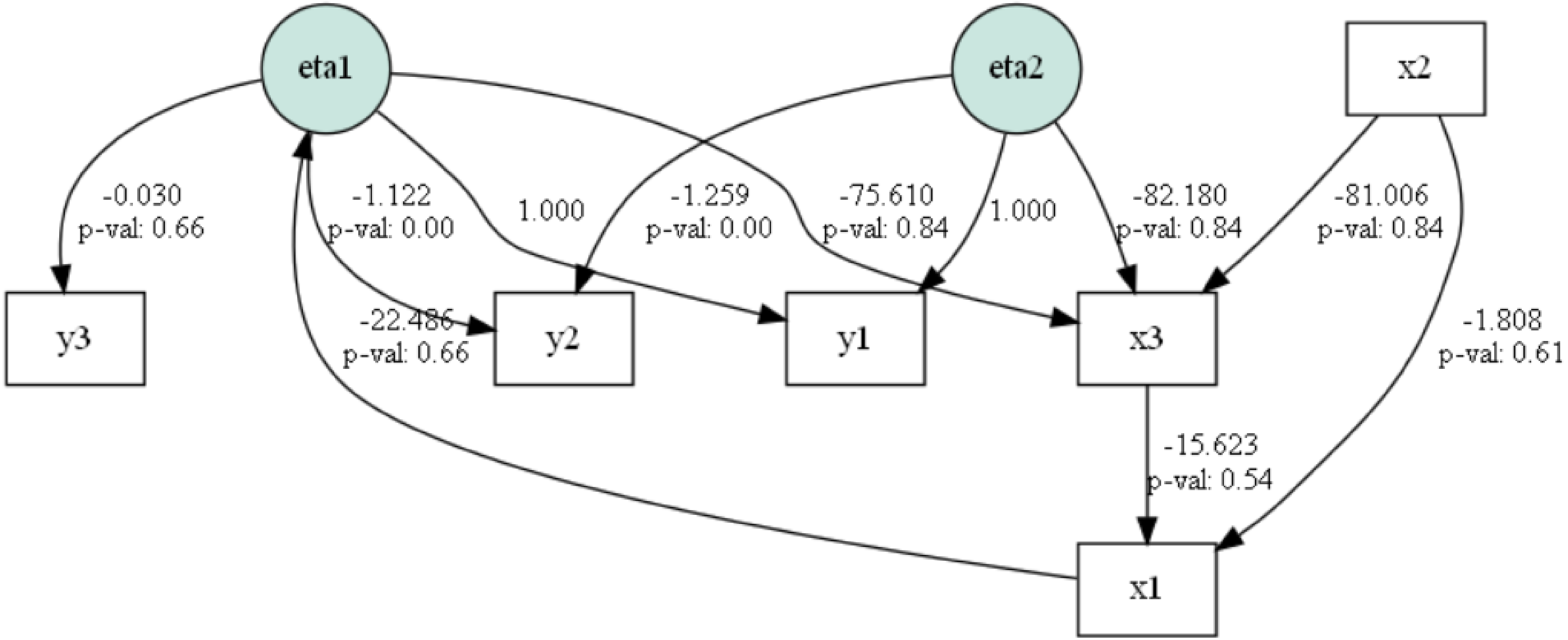
```
digraph G {
	overlap=scale splines=true
	edge [fontsize=12]
	node [fillcolor="#cae6df" shape=circle style=filled]
	eta1 [label=eta1]
	eta2 [label=eta2]
	node [shape=box style=""]
	y1 [label=y1]
	y2 [label=y2]
	y3 [label=y3]
	x1 [label=x1]
	x2 [label=x2]
	x3 [label=x3]
	x2 -> x1 [label="-1.808\np-val: 0.61"]
	x3 -> x1 [label="-15.623\np-val: 0.54"]
	x2 -> x3 [label="-81.006\np-val: 0.84"]
	eta1 -> x3 [label="-75.610\np-val: 0.84"]
	eta2 -> x3 [label="-82.180\np-val: 0.84"]
	x1 -> eta1 [label="-22.486\np-val: 0.66"]
	eta1 -> y1 [label=1.000]
	eta2 -> y1 [label=1.000]
	eta1 -> y2 [label="-1.122\np-val: 0.00"]
	eta2 -> y2 [label="-1.259\np-val: 0.00"]
	eta1 -> y3 [label="-0.030\np-val: 0.66"]
}
```

Implementing SEM in Python (cont.)

Example: code (cont,)

- use *IPython.display.Image* to visualize the Model instance

```
from IPython.display import Image
Image("images/semopy_multivariate_regression_example.png")
```



Summary

Structural Equation Modeling

Introduction to SEM

SEM Structure and Steps

SEM Graphical Path Diagram

SEM Notation

Implementing SEM in Python

- *SEMOPY framework*

Next Lesson:

- Causal Bayesian Networks

Thank You!

Questions?