ADVANCED DATA ANALYSIS FOR PSYCHOLOGICAL SCIENCE

Extra slides: CFA and SEM evaluation

Luca Menghini Ph.D.

luca.menghini@unipd.it

***

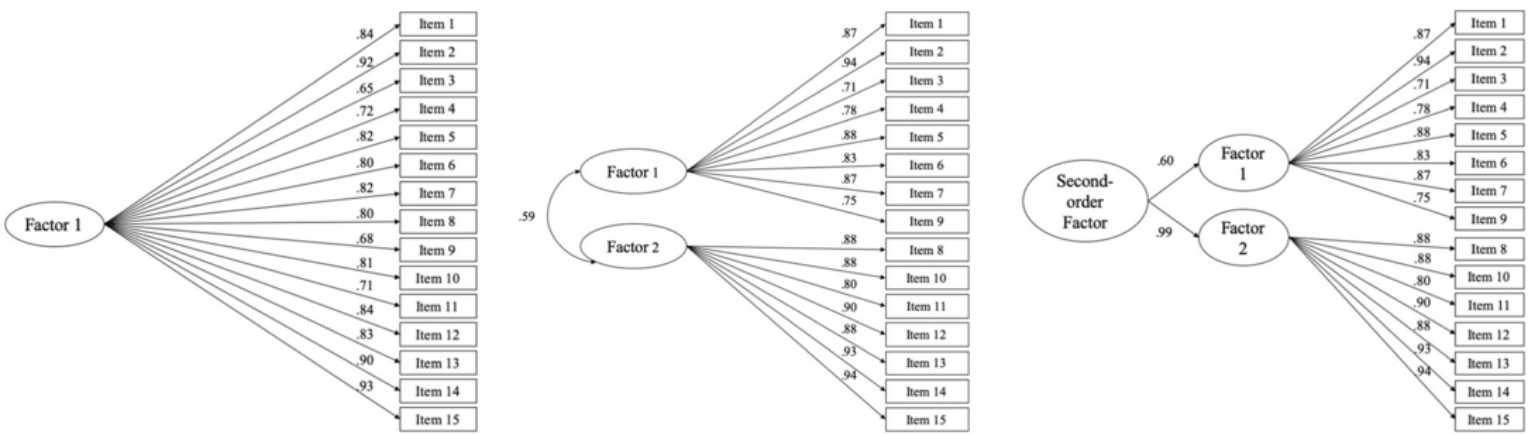
Master degree in Developmental and Educational Psychology

University of Padova

2023-2024

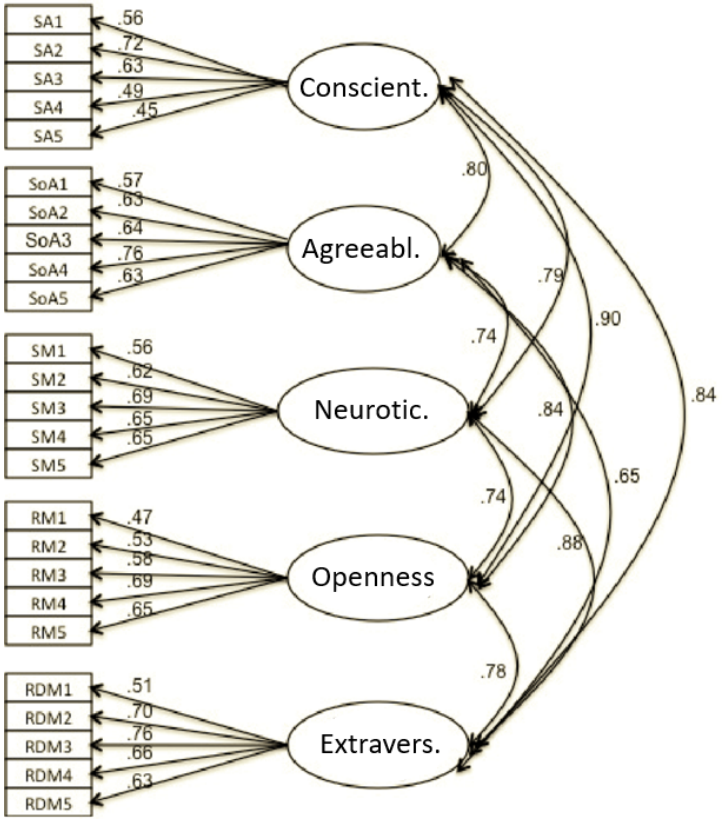
Confirmatory factor analysis (CFA)

- A CFA model is a SEM that includes **both observed & latent variables**
- In 'full' SEM, the measurement model **forms the latent variables** (also called *latent factors* or just *factors*) to be used in the structural model
- In CFA, there is **no structural model** (no directional relationship between latent variables) but **just the measurement model**:
  → CFA focuses on the relationships between latent and observed variables
- Such relationships are called **factor loadings** and are considered as quantitative indicators of the **construct validity** of a set of indicators (e.g., items of a scale)



Source: Moran et al (2021) doi.org/10.1007/s00520-020-05568-w

Factor structure



A **factor structure** is one of the possible structures of relationships between a number of observed variables that are said to measure one or more particular latent factors.

The factor structure of a CFA model defines:

1. the **number** of latent variables (one-factor vs. two-factor, vs. N-factor model)
2. the **relationships** between each particular observed variable and the corresponding latent variable (for models with 2+ latent)

Starting from a covariance matrix of observed variables, CFA tests the **goodness of fit** of an hypothesized factor structure (or set of alternative structures) and provides estimates of the resulting **factor loadings**

Confirmatory vs. Exploratory factor analysis

Confirmatory factor analysis is called "*confirmatory*" due to the assumptions on the underlying factor structure:

Exploratory factor analysis (EFA)

There are **no hypotheses on the factor structure** (unknown number of factors and factor loadings)

Thus, we do not *impose* any predefined structure on the model (data-driven approach), but **the structure is the output**

In psychological testing, EFA is often used in the **initial stages** of measure development

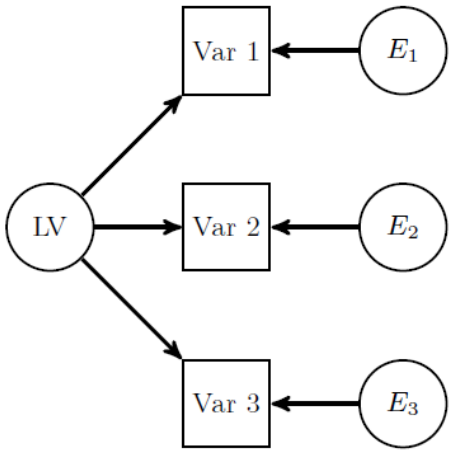
Confirmatory factor analysis (CFA)

There is prior knowledge of the theory, empirical research, or both, to **postulate the factor structure a priori** and test the hypothesis statistically

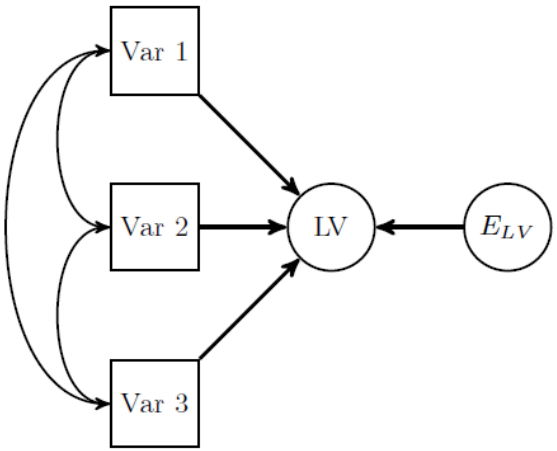
Used to **verify** the hypothesized relationship pattern and **test/quantify** the hypotheses on the factor loadings

Often used in the **later stages** of measure development (including adaptation into other languages)

Reflective vs. Formative models



(a) *LV* is a reflective latent variable.



(b) *LV* is a formative latent variable.

Reflective latent variables are thought to *cause* observed variable variances and covariances

Basic idea: There is a (small number of) latent variable(s) within a given domain (e.g., personality) that influence each of its observed indicators (*parallel forms*) producing the observed covariance matrix

Formative latent variables are thought to be *the result* of observed variable covariation (similar to multiple regression)

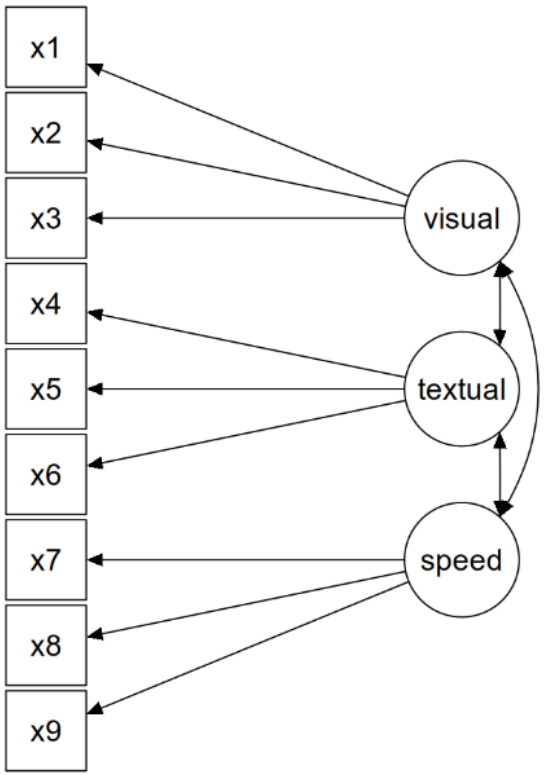
Less common in psychological testing, useful for measures such as **symptom checklists**, whose items are *not* considered as *parallel forms* (i.e., you can have one symptom but not the others)

Case study: Children mental abilities

'Classic' Holzinger and Swineford (1939) dataset used in many SEM papers and books. It is included in the `lavaan` pkg and consists of a subset of **9 mental ability test scores** from 301 7th- and 8th-grade children .

```
data(HolzingerSwineford1939,package="lavaan")
hs39 <- HolzingerSwineford1939 # shortening data name
head(hs39,3) # showing first 3 lines
```

```
  id   x1   x2   x3   x4   x5   x6   x7   x8   x9
1  1 3.33 7.75 0.38 2.33 5.75 1.29 3.39 5.75 6.36
2  2 5.33 5.25 2.12 1.67 3.00 1.29 3.78 6.25 7.92
3  3 4.50 5.25 1.88 1.00 1.75 0.43 3.26 3.90 4.42
```

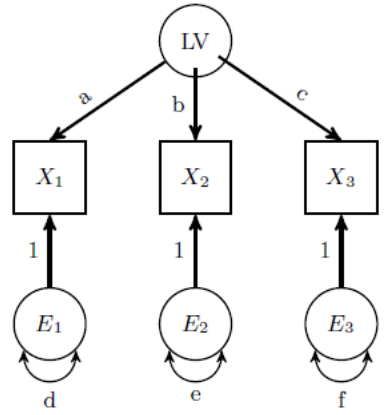


Hypothesized underlying **three-factor structure** with a *visual* (items *x1*, *x2*, *x3*), a *textual* (items *x4*, *x5*, *x6*), and a *speed* factor (items *x7*, *x8*, and *x9*)

Model identification (1/3)

	X_1	X_2	X_3
X_1	σ_1^2		
X_2	σ_{12}	σ_2^2	
X_3	σ_{13}	σ_{23}	σ_3^2

(a) Covariance matrix showing non-redundant elements.



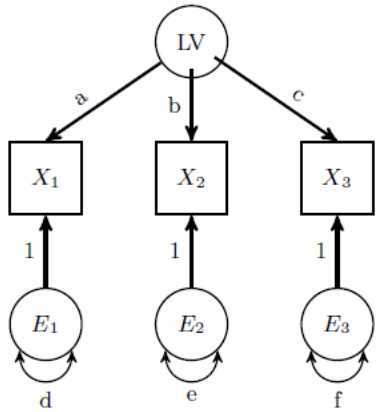
(b) Latent variable model.

- To quantify/form latent variables, we need to use some observed information (i.e., the covariance matrix of observed variables) to estimate something that cannot be observed
- However, we cannot estimate a number of **unknown parameters** that is lower than the number of **non-redundant information in the data** (ident. rule)
- The **number of non-redundant information** in the covariance matrix S of p observed variables can be computed as $p(\,p+1)\,/\,2$
- The **number of parameters** can be determined by... well, you know how ☺

Model identification (2/3)

	X_1	X_2	X_3
X_1	σ_1^2		
X_2	σ_{12}	σ_2^2	
X_3	σ_{13}	σ_{23}	σ_3^2

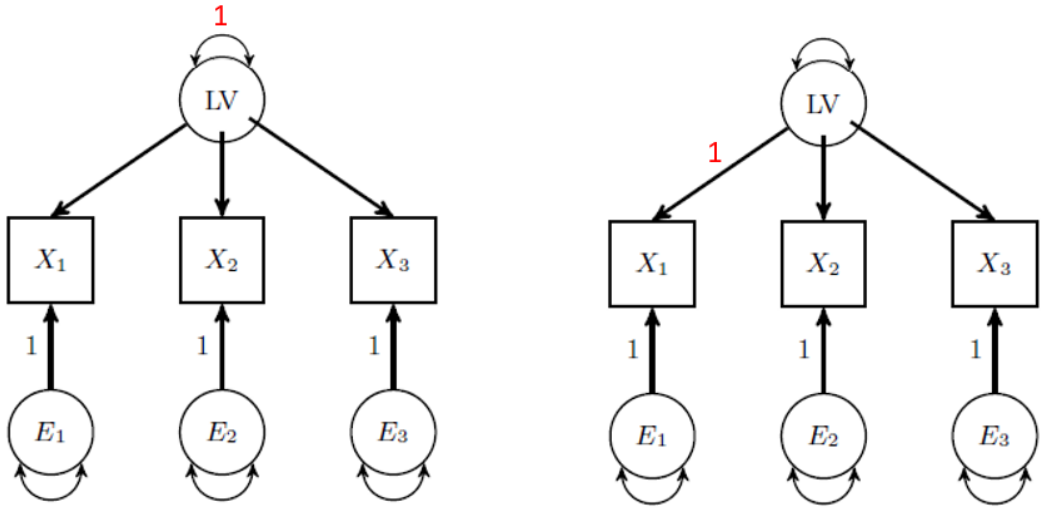
(a) Covariance matrix showing non-redundant elements.



(b) Latent variable model.

- In CFA models, the difference between the number of non-redundant information ($p(\ p+1)\ /\ 2$) and the number of parameters to estimate are called **degrees of freedom** (df) of the model
- When $df < 0$, the model is **underidentified** and *we cannot uniquely estimate* the parameters
- When $df = 0$, the model is **just-identified** and there should be *unique estimates* for each parameter (note: LM models are just-identified models)
- When $df > 0$, the model is **overidentified**, providing both *unique estimates* and *measures of model fit* (see Model evaluation)
- We want our model to be overidentified.

Model identification (3/3)



(a) Standardizing LV

(b) Fixing one loading to 1

- With **4+ indicators per latent variable, *df* would be = 0**
- With **3 indicators per latent variable, *df* would be < 0***

However, we can *constrain* some parameters to **set the latent variable's scale**:

- We can constrain the LV variance to 1 → **standardization of the latent variable** (note: if the observed variables are standardized as well, we're using the *standardized solution*)
- We can constrain a single factor loading for each LV to an arbitrary value (usually, 1) → **marker value** (used to determinate the latent variable variance)

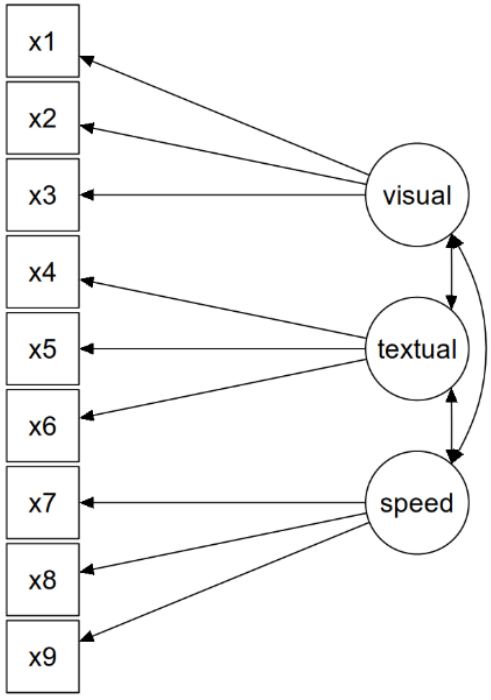
Model identification in our case study

In our example we have 9 observed variables ($p = 9$):

- how many unknown parameters?
  9 loadings + 9 errors + 3 variances + 3 covariances = 24*
- how many non-redundant information in the observed covariance matrix?
  $p(p + 1) / 2 = 9\ (9 + 1) / 2 = 45$

```
cov(hs39[,paste0("x",1:9)]) # observed covar. matrix
```

```
     x1    x2   x3   x4   x5   x6   x7   x8   x9
x1 1.36    NA   NA   NA   NA   NA   NA   NA   NA
x2 0.41  1.39   NA   NA   NA   NA   NA   NA   NA
x3 0.58  0.45 1.28   NA   NA   NA   NA   NA   NA
x4 0.51  0.21 0.21 1.36   NA   NA   NA   NA   NA
x5 0.44  0.21 0.11 1.10 1.67   NA   NA   NA   NA
x6 0.46  0.25 0.24 0.90 1.02 1.20   NA   NA   NA
x7 0.09 -0.10 0.09 0.22 0.14 0.14 1.19   NA   NA
x8 0.26  0.11 0.21 0.13 0.18 0.17 0.54 1.03   NA
x9 0.46  0.24 0.38 0.24 0.30 0.24 0.37 0.46 1.02
```

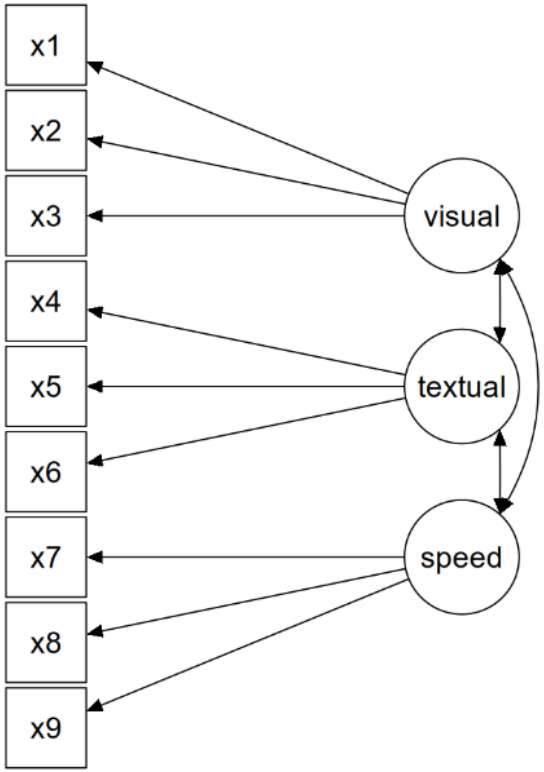


*Note: in CFA, also latent variable variances and covariances count as unknown parameters

Fitting a CFA in R

`lavaan` uses the symbol `=~` to set a latent variable as ***reflective of*** a number of observed variables.

```
library(lavaan)
mymodel <- 'visual =~ x1 + x2 + x3
            textual =~ x4 + x5 + x6
            speed =~ x7 + x8 + x9'
fit <- cfa(mymodel, data=hs39)
```



CFA model output: unstandardized solution

```
parameterestimates(fit)
```

lhs	op	rhs	est	se	z	pvalue
visual	=~	x1	1.00	0.00	NA	NA
visual	=~	x2	0.55	0.10	5.55	0
visual	=~	x3	0.73	0.11	6.68	0
textual	=~	x4	1.00	0.00	NA	NA
textual	=~	x5	1.11	0.07	17.01	0
textual	=~	x6	0.93	0.06	16.70	0
speed	=~	x7	1.00	0.00	NA	NA
speed	=~	x8	1.18	0.16	7.15	0
speed	=~	x9	1.08	0.15	7.15	0
x1	~~	x1	0.55	0.11	4.83	0
x2	~~	x2	1.13	0.10	11.15	0
x3	~~	x3	0.84	0.09	9.32	0
x4	~~	x4	0.37	0.05	7.78	0
x5	~~	x5	0.45	0.06	7.64	0
x6	~~	x6	0.36	0.04	8.28	0
x7	~~	x7	0.80	0.08	9.82	0

CFA model output: standardized solution

```
standardizedsolution(fit)
```

lhs	op	rhs	est.std	se	z	pvalue
visual	=~	x1	0.77	0.05	14.04	0
visual	=~	x2	0.42	0.06	7.11	0
visual	=~	x3	0.58	0.06	10.54	0
textual	=~	x4	0.85	0.02	37.78	0
textual	=~	x5	0.86	0.02	38.27	0
textual	=~	x6	0.84	0.02	35.88	0
speed	=~	x7	0.57	0.05	10.71	0
speed	=~	x8	0.72	0.05	14.31	0
speed	=~	x9	0.67	0.05	13.02	0
x1	~~	x1	0.40	0.08	4.76	0
x2	~~	x2	0.82	0.05	16.25	0
x3	~~	x3	0.66	0.06	10.33	0
x4	~~	x4	0.27	0.04	7.16	0
x5	~~	x5	0.27	0.04	7.04	0
x6	~~	x6	0.30	0.04	7.61	0
x7	~~	x7	0.68	0.06	11.16	0

Parameter matrices

Let's see how parameter matrices change with a CFA:

λ = matrix of **factor loadings**

```
inspect( fit, "estimates")[1]
```

```
$lambda
   visual textul speed
x1  1.000  0.000 0.000
x2  0.554  0.000 0.000
x3  0.729  0.000 0.000
x4  0.000  1.000 0.000
x5  0.000  1.113 0.000
x6  0.000  0.926 0.000
x7  0.000  0.000 1.000
x8  0.000  0.000 1.180
x9  0.000  0.000 1.082
```

ψ = matrix of **latent variable (co)variances**

```
inspect( fit, "estimates")[3]
```

```
$psi
        visual textul speed
visual   0.809
textual  0.408  0.979
speed    0.262  0.173 0.384
```

θ = matrix of **observed variable (co)variances**

```
inspect( fit, "estimates")[2]
```

```
$theta
      x1    x2    x3    x4    x5    x6    x7    x8    x
x1 0.549
x2 0.000 1.134
x3 0.000 0.000 0.844
x4 0.000 0.000 0.000 0.371
x5 0.000 0.000 0.000 0.000 0.446
x6 0.000 0.000 0.000 0.000 0.000 0.356
x7 0.000 0.000 0.000 0.000 0.000 0.000 0.799
x8 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.488
x9 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.5
```

β = matrix of **regression coefficients** (paths)

```
inspect( fit, "estimates")[4]
```

```
$<NA>
NULL
```

Multivariate model evaluation

With 'model evaluation' we refer to two main procedures:

- **Model diagnostics**: Evaluating whether the model fits the data consistently with the underlying ***model assumptions***
- **Fit evaluation**: Evaluating the goodness of fit of the model to the data
- **Model comparison**: Evaluating whether the model fits substantially better or worse than alternative models → ***model selection*** (choosing the best model)

Data analysis pipeline

1. Data exploration & descriptives
2. Model fit
3. Model diagnostics
4. Model comparison
5. Model selection & coefficient interpretation
6. Result visualization

SEM assumptions

Similar to LM(ER), SEM require that some **assumptions about the data** hold true. Otherwise, we cannot trust the estimated parameters or any other result.

Assumptions common to LM:

1. **Linearity**: expected (mean) value of residuals is zero
2. **Normality**: residuals are normally distributed
3. **Homoscedasticity**: residual variance is constant over the levels of fitted values
4. **Independence** between residuals and fitted values
5. **Absence of influential observations** (multivariate outliers)
6. **Absence of multicollinearity**: no linear relationship between different predictors

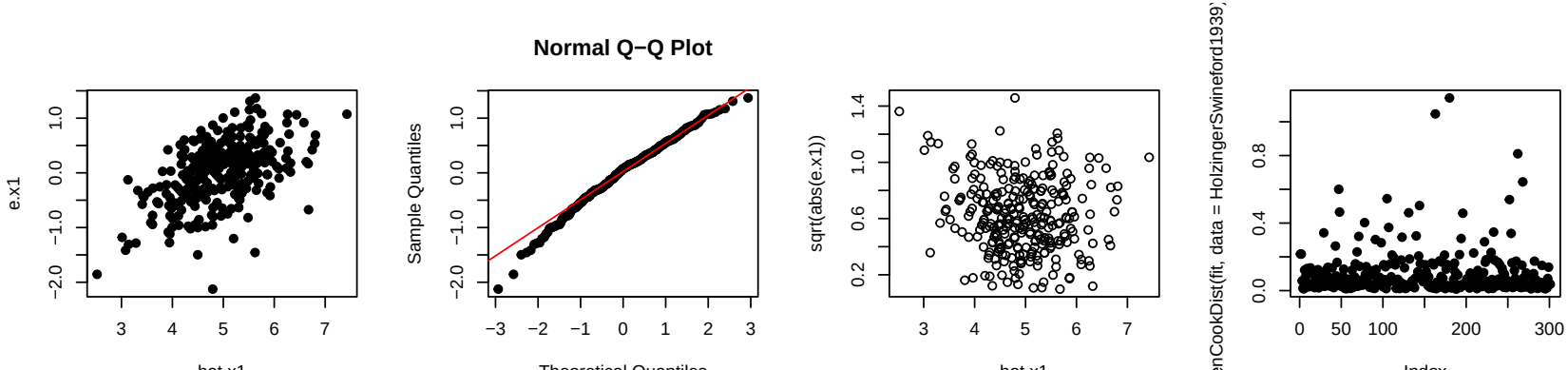
Evaluating SEM assumptions

Here, we see just an example for the endogenous variable **x1** from the previous example.

```
library(influence.SEM)
outres <- sem.fitres( fit ) # computing residuals
head(outres[,c("x1","hat.x1","e.x1")]) # residuals (e) = observed - predicted (hat)
```

```
        x1   hat.x1        e.x1
1 3.333333 4.118095 -0.78476149
2 5.333333 4.985289  0.34804466
3 4.500000 4.174373  0.32562700
4 5.333333 5.355112 -0.02177832
5 4.833333 4.519865  0.31346875
6 5.333333 4.959025  0.37430783
```

```
plot( e.x1 ~ hat.x1, data = outres, pch = 19) # violation of independence
qqnorm( outres$e.x1, pch = 19 ); qqline( outres$e.x1, col = "red") # normality quite
plot( sqrt( abs( e.x1 ) ) ~ hat.x1, data = outres ) # omoscedasticity ok
plot( genCookDist( fit, data = HolzingerSwineford1939 ), pch = 19 ) # some influentio
```



Evaluating SEM goodness of fit

The general principle behind the evaluation of the goodness of fit of a SEM is based on the **comparison** between the **observed covariance matrix** S and the **predicted covariance matrix** $\Sigma(\hat{\theta})$ that is implied by the models based on parameter estimates θ

The smaller the distance between S and $\Sigma(\hat{\theta})$, the better goodness of fit

When the model is not saturated ($df > 0$), there are several **fit indices** to evaluate such distance either based on residuals or on the difference between the target and a baseline model, such as:

- Root-Mean-Square Error of Approximation (RMSEA) $\rightarrow$ should be < 0.06
- Standardized root mean square residual (SRMR) $\rightarrow$ should be < 0.08
- Comparative fit index (CFI) $\rightarrow$ should be > 0.95

```
inspect(fit,"fit")[c("rmsea","srmr","cfi")]
```

```
     rmsea       srmr        cfi
0.09212148 0.06520506 0.93055965
```

Model comparison based on fit indices

The same **fit indices** can be used to **compare multiple models** and find the one that shows the better fit.

```
# model 1: 3 factors
mymodel1 <- 'visuoTextual =~ x1 + x2 + x3
             textual =~ x4 + x5 + x6
             speed =~ x7 + x8 + x9'
fit1 <- cfa(mymodel1, data=hs39)

# model 2: 2 factors
mymodel2 <- 'visuoTextual =~ x1 + x2 + x3 + x4 + x5 + x6
             speed =~ x7 + x8 + x9'
fit2 <- cfa(mymodel2, data=hs39)

# model comparison (model 1 is better)
rbind(inspect(fit1,"fit")[c("rmsea","srmr","cfi")],
      inspect(fit2,"fit")[c("rmsea","srmr","cfi")])
```

```
          rmsea       srmr       cfi
[1,] 0.09212148 0.06520506 0.9305597
[2,] 0.14088568 0.11255352 0.8240514
```

Model comparison based on information criteria

SEM can also be compared based on the same information criteria that we saw in Part 1: the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC).

```
AIC(fit1,fit2) # AIC: the lower the better -> m1 is better
```

```
     df      AIC
fit1 21 7517.490
fit2 19 7609.521
```

```
MuMIn::Weights(AIC(fit1,fit2)) # AIC weight: the higher the better
```

```
 model weights
[1] 1 0
```

```
BIC(fit1,fit2) # BIC: the lower the better -> m1 is better
```

```
     df      BIC
fit1 21 7595.339
fit2 19 7679.956
```

```
MuMIn::Weights(BIC(fit1,fit2)) # BIC weight: the higher the better
```

```
 model weights
[1] 1 0
```

Model comparison based on likelihood ratio test

Even the likelihood ratio test (see Part 1) can be used to compare two nested SEM:

```
anova(fit1,fit2) # in this case it is significant, but...
```

```
Chi-Squared Difference Test

     Df    AIC    BIC   Chisq Chisq diff   RMSEA Df diff Pr(>Chisq)
fit1 24 7517.5 7595.3  85.305
fit2 26 7609.5 7680.0 181.337     96.031 0.39522       2  < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Coefficient of determination in SEM

SEM allow to compute a **coefficient of determination** R^2 for each endogenous variable. The interpretation is identical to that shown in Part 1.

```
inspect( fit1 , 'rsquare' )
```

```
   x1    x2    x3    x4    x5    x6    x7    x8    x9
0.596 0.179 0.338 0.725 0.731 0.702 0.324 0.523 0.442
```

Interpretation: the model explains from 18 to 73% of the variance in the obs variables

That's all for now (and forever)! :)

Credits

The present slides are partially based on:

- Beaujean, A. A. (2014) Latent Variable Modeling Using R. A Step-by-Step Guide. New York: Routledge
- Pastore, M. (2015). Analisi dei dati in psicologia (e applicazioni in R). Il Mulino.
- Pastore, M. (2021). Analisi dei dati in ambito di comunità

Achronyms & Greek letters

- CFA: confirmatory factor analysis
- LM: linear models/modeling
- LV: latent variable
- OV: observed variable
- SEM: structural equation models/modeling
- SS: sum of squares

- β = *beta*, indexing path coefficients (or regression coefficients)
- ϵ = *epsilon*, indexing the error of an observed variable
- σ = *sigma*, indexing the variance σ^2 of the errors ϵ
- η = *eta*, indexing latent variables
- θ = *theta*, indexing overall model parameters

Achronyms & Greek letters

- CFA: confirmatory factor analysis
- LM: linear models/modeling
- LV: latent variable
- OV: observed variable
- SEM: structural equation models/modeling
- SS: sum of squares

- β = *beta*, indexing path coefficients (or regression coefficients)
- ϵ = *epsilon*, indexing the error of an observed variable
- σ = *sigma*, indexing the variance σ^2 of the errors ϵ
- η = *eta*, indexing latent variables
- θ = *theta*, indexing overall model parameters
- ciao