Multivariate Statistics and Methodology with R

Practical Issues in Structural Equation Modeling

Aja Murray; Aja.Murray@ed.ac.uk

This week

- Techniques
  - *Full structural equation modeling with:*
    - Non-normal data
    - Ordered-categorical items
    - Missing data
- Functions
  - *sem( ) and cfa( ) from lavaan*
- Reading
  - *lavaan tutorial: http://lavaan.ugent.be/tutorial/tutorial.pdf (sections 10 and 12)*

Learning outcomes

- Know how to deal with non-normal data in SEM
- Know how to fit structural equation models with ordered-categorical items
- Understand the distinction between different missing data mechanisms and how to deal with missingness in SEM
  - *Full information maximum likelihood estimation*
  - *Multiple imputation*

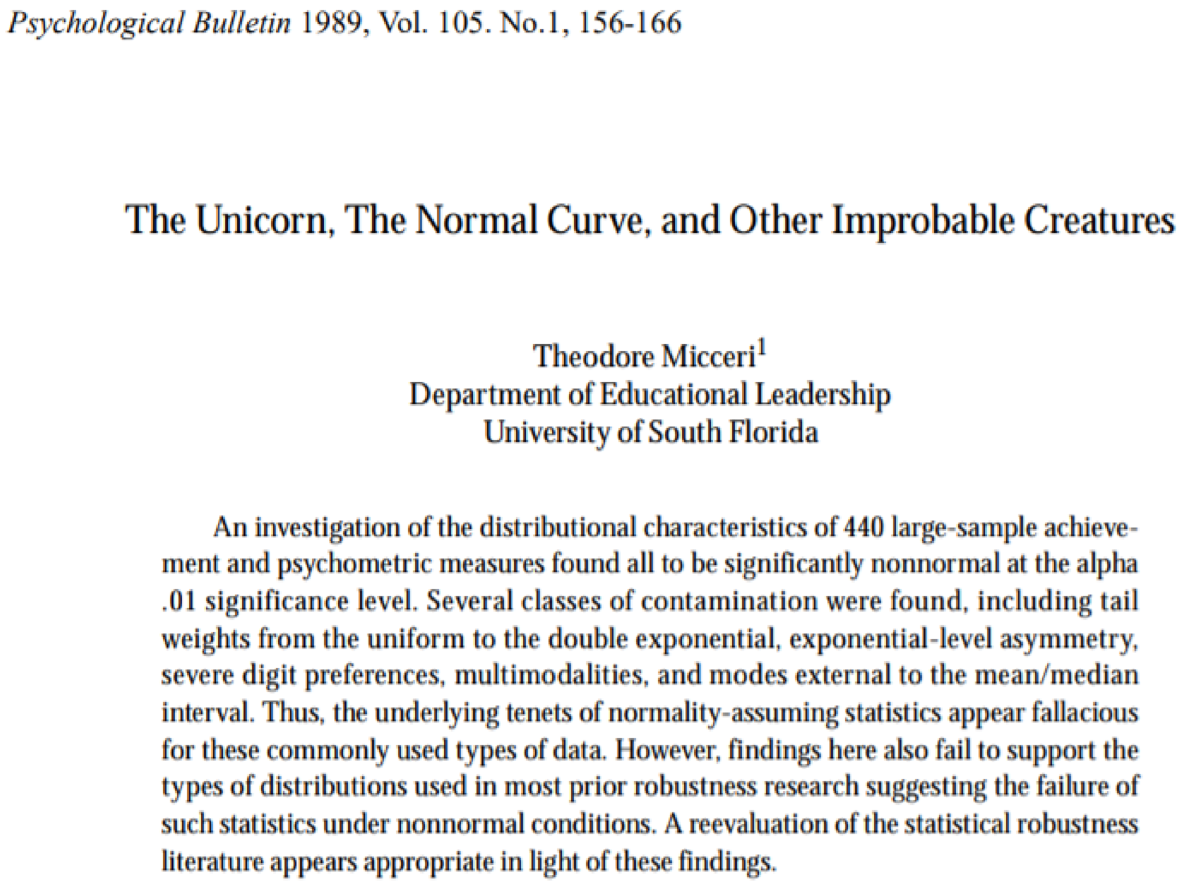
SEM with continuous normally distributed items

- Thus far we have been assuming that our items approximate continuous distributions
- Justifiable if:
  - *We have a truly continuous measure*
  - *We have at least 5 response options*
  - *All 5 response options are used by participants*
  - *The distribution of responses is close to a normal distribution*
- If these conditions are met, we can happily use maximum likelihood estimation to fit our models

Why item distributions matter

- In SEM, non-normality matters because SEM uses the sample covariance matrix
- The sample covariance matrix completely summarises the variation in observed variables only if there is multivariate normality
- If not, information from higher-order moments (skewness, kurtosis) needs to be taken into account

SEM with continuous non-normally distributed items



Psychological Bulletin 1989, Vol. 105. No.1, 156-166

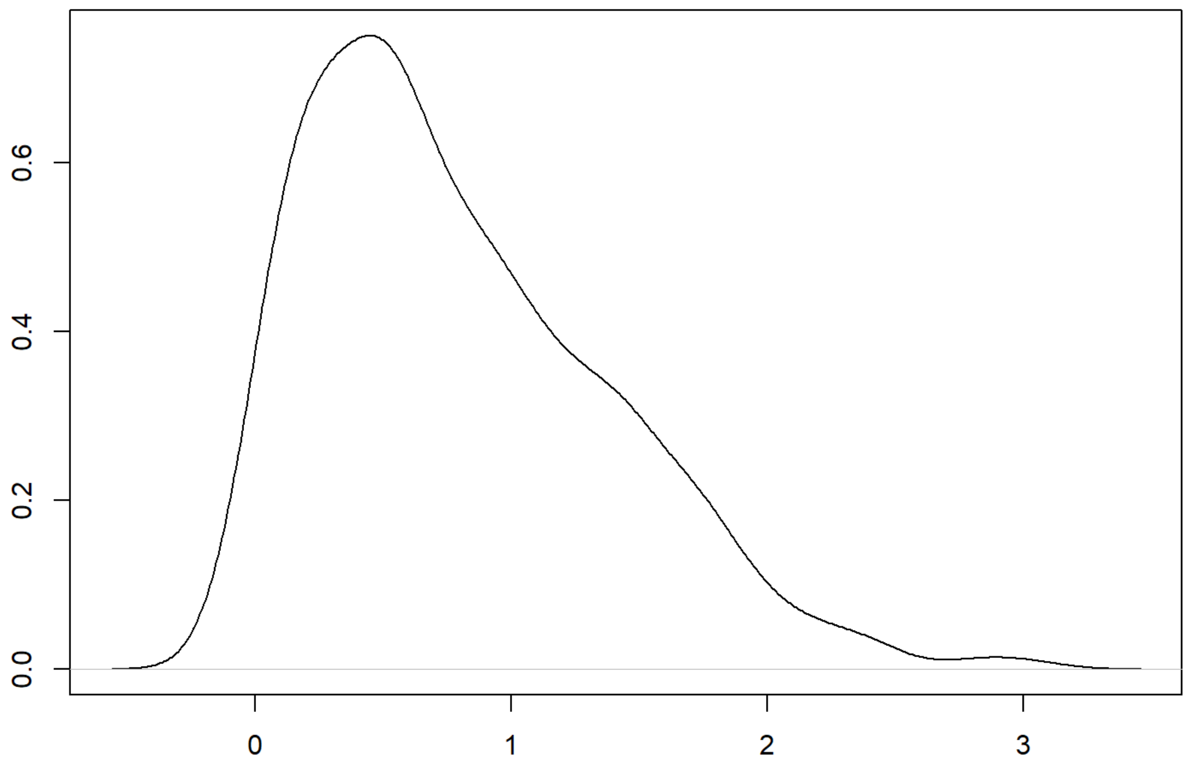
The Unicorn, The Normal Curve, and Other Improbable Creatures

Theodore Micceri[1]
Department of Educational Leadership
University of South Florida

An investigation of the distributional characteristics of 440 large-sample achievement and psychometric measures found all to be significantly nonnormal at the alpha .01 significance level. Several classes of contamination were found, including tail weights from the uniform to the double exponential, exponential-level asymmetry, severe digit preferences, multimodalities, and modes external to the mean/median interval. Thus, the underlying tenets of normality-assuming statistics appear fallacious for these commonly used types of data. However, findings here also fail to support the types of distributions used in most prior robustness research suggesting the failure of such statistics under nonnormal conditions. A reevaluation of the statistical robustness literature appears appropriate in light of these findings.

- Continuous but non-normal data is extremely common in psychology

SEM with continuous non-normally distributed items



- Consequences of non-normality in SEM are:
  - *Model fit is underestimated*
  - *Standard errors of parameter estimates are under-estimated*
  - *Statistical significance is overestimated*

SEM with continuous non-normally distributed items

- Possible solutions are:
  - *Transform the variables to have normal distributions*
  - *Use bootstrapping*
  - *Use a **robust estimator***

Transforming the variables to have normal distributions

- In linear regression it can sometimes be helpful to transform our outcome variable to normality
  - ln *or* log_{10} *transformations*
  - *Box-cox transformations*
- However, this can make interpretation more difficult because the scale of the variable changes
- In SEM you are dealing with a much bigger set of variables, making transformation more difficult/cumbersome
- Not the best option

Using bootstrapping for non-normal distributions

- We discussed bootstrapping in the context of the indirect effects in mediation
- Can be used to assess significance when a statistic would not be expected to follow its theoretical sampling distribution
  - *Can, therefore, be used in the presence of non-normal variables to assess parameter significance*
  - *Can also be used for the model test statistic*

Using a robust estimator for non-normally distributed items

- Robust maximum likelihood estimation:
  - *Gives the same parameter estimates as ML*
  - *Corrects the fit statistics using a* ***correction factor***
  - *The correction factor depends on the degree of non-normality*
  - *More non-normality = a bigger correction*
  - *Corrects the standard errors using a conceptually similar mechanism*

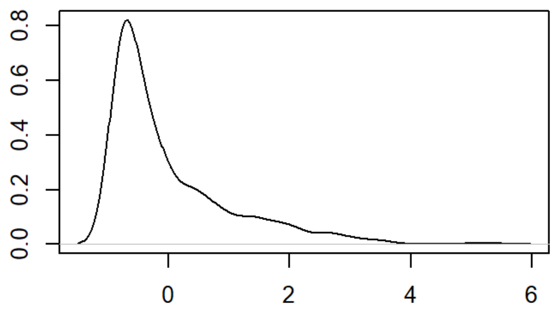
Example in lavaan

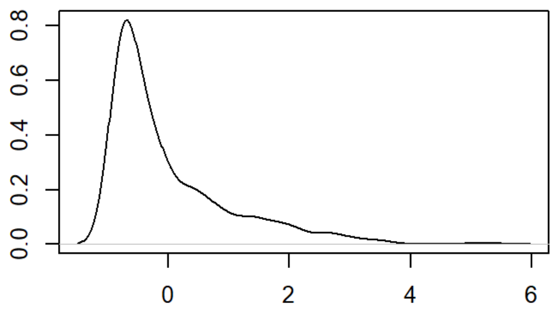
- A researcher wants to test whether a one-factor CFA fits for her 4-item aggression scale
- She collects n=500 datapoints from a set of emerging adults
- The data looks like…

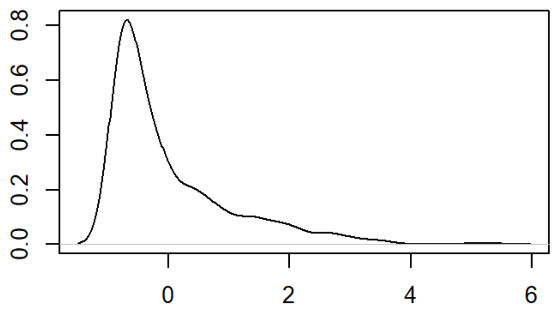
```
library(psych)
describe(Agg_data)
```

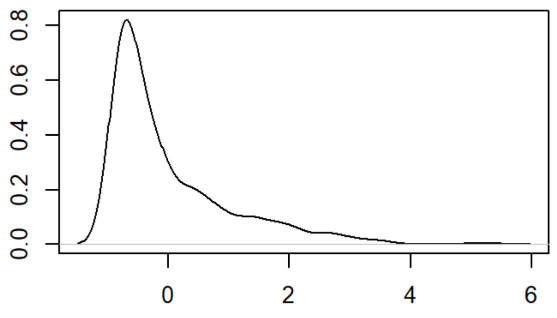
```
##       vars   n mean   sd median trimmed  mad   min  max range skew kurtosis
## item1    1 500 0.01 1.02  -0.40   -0.18 0.59 -0.83 5.32  6.15 1.75     3.30
## item2    2 500 0.05 1.31  -0.42   -0.21 0.83 -1.04 6.34  7.39 1.76     3.13
## item3    3 500 0.17 1.28  -0.24   -0.06 0.96 -0.98 5.25  6.23 1.64     2.63
## item4    4 500 0.11 1.16  -0.37   -0.12 0.64 -0.83 7.26  8.09 2.05     5.57
##         se
## item1 0.05
## item2 0.06
## item3 0.06
## item4 0.05
```

Observed variable distributions









Fitting the model ignoring non-normality

- First, let’s fit the model ignoring non-normality
- No obvious signs that anything is amiss…
- Important to use descriptive and graphical checks prior to model-fitting

```
model1<-'Agg=~item1+item2+item3+item4'
model1.est<-cfa(model1, data=Agg_data)
summary(model1.est, fit.measures=T, standardized=T)
```

```
## lavaan 0.6-5 ended normally after 31 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of free parameters                          8
##
##   Number of observations                           500
##
## Model Test User Model:
##
##   Test statistic                                12.998
##   Degrees of freedom                                 2
##   P-value (Chi-square)                           0.002
##
## Model Test Baseline Model:
##
##   Test statistic                               187.030
##   Degrees of freedom                                 6
##   P-value                                        0.000
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.939
##   Tucker-Lewis Index (TLI)                       0.818
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)              -3097.350
##   Loglikelihood unrestricted model (H1)      -3090.851
##
##   Akaike (AIC)                                6210.700
##   Bayesian (BIC)                              6244.417
##   Sample-size adjusted Bayesian (BIC)         6219.024
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.105
##   90 Percent confidence interval - lower         0.056
##   90 Percent confidence interval - upper         0.162
##   P-value RMSEA <= 0.05                          0.035
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.038
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##   Standard errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   Agg =~
##     item1             1.000                               0.583    0.570
##     item2             1.179    0.183    6.439    0.000    0.688    0.524
##     item3             0.942    0.160    5.907    0.000    0.550    0.429
##     item4             1.080    0.167    6.481    0.000    0.630    0.541
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .item1             0.705    0.068   10.424    0.000    0.705    0.675
##    .item2             1.252    0.108   11.610    0.000    1.252    0.726
##    .item3             1.337    0.100   13.352    0.000    1.337    0.816
##    .item4             0.957    0.086   11.182    0.000    0.957    0.707
##     Agg               0.340    0.070    4.826    0.000    1.000    1.000
```

Using bootstrapping for non-normal data

- Now let's try bootstrapping
- We can set se='bootstrap' to obtain bootstrapped standard errors for the parameter estimates

```
model1<-'Agg=~item1+item2+item3+item4'
model1.est<-cfa(model1, data=Agg_data, se='bootstrap')
summary(model1.est, fit.measures=T, standardized=T, ci=T)
```

```
## lavaan 0.6-5 ended normally after 31 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of free parameters                          8
##
##   Number of observations                           500
##
## Model Test User Model:
##
##   Test statistic                                12.998
##   Degrees of freedom                                 2
##   P-value (Chi-square)                           0.002
##
## Model Test Baseline Model:
##
##   Test statistic                               187.030
##   Degrees of freedom                                 6
##   P-value                                        0.000
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.939
##   Tucker-Lewis Index (TLI)                       0.818
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)              -3097.350
##   Loglikelihood unrestricted model (H1)      -3090.851
##
##   Akaike (AIC)                                6210.700
##   Bayesian (BIC)                              6244.417
##   Sample-size adjusted Bayesian (BIC)         6219.024
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.105
##   90 Percent confidence interval - lower         0.056
##   90 Percent confidence interval - upper         0.162
##   P-value RMSEA <= 0.05                          0.035
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.038
##
## Parameter Estimates:
##
##   Standard errors                            Bootstrap
##   Number of requested bootstrap draws             1000
##   Number of successful bootstrap draws            1000
##
```

```
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|) ci.lower ci.upper
##   Agg =~
##     item1             1.000                               1.000    1.000
##     item2             1.179    0.199    5.912    0.000    0.832    1.626
##     item3             0.942    0.278    3.393    0.001    0.526    1.607
##     item4             1.080    0.277    3.900    0.000    0.702    1.764
##    Std.lv  Std.all
##
##     0.583    0.570
##     0.688    0.524
##     0.550    0.429
##     0.630    0.541
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|) ci.lower ci.upper
##    .item1             0.705    0.101    6.953    0.000    0.493    0.892
##    .item2             1.252    0.147    8.511    0.000    0.961    1.550
##    .item3             1.337    0.141    9.465    0.000    1.055    1.609
##    .item4             0.957    0.134    7.152    0.000    0.693    1.213
##     Agg               0.340    0.100    3.401    0.001    0.176    0.578
##    Std.lv  Std.all
##     0.705    0.675
##     1.252    0.726
##     1.337    0.816
##     0.957    0.707
##     1.000    1.000
```

Using bootstrapping for non-normal data

- We can set test='bootstrap' to obtain a bootstrapped p-value for our test statistic (χ^2)

```
model1<-'Agg=~item1+item2+item3+item4'
model1.est<-cfa(model1, data=Agg_data, test='bootstrap')
```

```
## Warning in bootstrap.internal(object = NULL, lavmodel. = lavmodel,
## lavsamplestats. = lavsamplestats, : lavaan WARNING: only 999 bootstrap draws
## were successful
```

```
summary(model1.est, standardized=T, fit.measures=T)
```

```
## lavaan 0.6-5 ended normally after 31 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of free parameters                          8
##
##   Number of observations                           500
##
## Model Test User Model:
##
##   Test statistic                                12.998
##   Degrees of freedom                                 2
##   P-value (Chi-square)                           0.002
##
##   Test statistic                                12.998
##   Degrees of freedom                                 2
##   P-value (Bollen-Stine bootstrap)               0.002
##
## Model Test Baseline Model:
##
##   Test statistic                               187.030
##   Degrees of freedom                                 6
##   P-value                                        0.000
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.939
##   Tucker-Lewis Index (TLI)                       0.818
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)              -3097.350
##   Loglikelihood unrestricted model (H1)      -3090.851
##
##   Akaike (AIC)                                6210.700
##   Bayesian (BIC)                              6244.417
##   Sample-size adjusted Bayesian (BIC)         6219.024
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.105
##   90 Percent confidence interval - lower         0.056
##   90 Percent confidence interval - upper         0.162
##   P-value RMSEA <= 0.05                          0.035
##
## Standardized Root Mean Square Residual:
##
```

```
##   SRMR                                          0.038
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##   Standard errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   Agg =~
##     item1             1.000                               0.583    0.570
##     item2             1.179    0.183    6.439    0.000    0.688    0.524
##     item3             0.942    0.160    5.907    0.000    0.550    0.429
##     item4             1.080    0.167    6.481    0.000    0.630    0.541
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .item1             0.705    0.068   10.424    0.000    0.705    0.675
##    .item2             1.252    0.108   11.610    0.000    1.252    0.726
##    .item3             1.337    0.100   13.352    0.000    1.337    0.816
##    .item4             0.957    0.086   11.182    0.000    0.957    0.707
##     Agg               0.340    0.070    4.826    0.000    1.000    1.000
```

Issues with bootstrapping

- Assumes that resampling provides a good approximation to the sampling distribution of our statistic
- Whilst it can correct the χ^2 statistic p-value, this statistic may be of little interest
    - *Tendency to reject models that are trivially mis-specified*

Using a robust estimator for non-normal data

- We can use a robust estimator by setting:
  - *estimator='MLR' or estimator='MLM'*
  - *each provides a slightly different correction method*
  - *only MLR is available when there is no missing data*

Using MLM for non-normal data

```r
model1<-'Agg=~item1+item2+item3+item4'
model1.est<-cfa(model1, data=Agg_data, estimator='MLM')
summary(model1.est, fit.measures=T, standardized=T, ci=T)
```

```
## lavaan 0.6-5 ended normally after 31 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of free parameters                          8
##
##   Number of observations                           500
##
## Model Test User Model:
##                                               Standard      Robust
##   Test Statistic                                12.998       9.941
##   Degrees of freedom                                 2           2
##   P-value (Chi-square)                           0.002       0.007
##   Scaling correction factor                                  1.307
##     for the Satorra-Bentler correction
##
## Model Test Baseline Model:
##
##   Test statistic                               187.030     114.973
##   Degrees of freedom                                 6           6
##   P-value                                        0.000       0.000
##   Scaling correction factor                                  1.627
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.939       0.927
##   Tucker-Lewis Index (TLI)                       0.818       0.781
##
##   Robust Comparative Fit Index (CFI)                         0.941
##   Robust Tucker-Lewis Index (TLI)                            0.824
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)              -3097.350   -3097.350
##   Loglikelihood unrestricted model (H1)      -3090.851   -3090.851
##
##   Akaike (AIC)                                6210.700    6210.700
##   Bayesian (BIC)                              6244.417    6244.417
##   Sample-size adjusted Bayesian (BIC)         6219.024    6219.024
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.105       0.089
##   90 Percent confidence interval - lower         0.056       0.045
##   90 Percent confidence interval - upper         0.162       0.140
##   P-value RMSEA <= 0.05                          0.035       0.069
##
##   Robust RMSEA                                               0.102
##   90 Percent confidence interval - lower                     0.045
##   90 Percent confidence interval - upper                     0.169
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.038       0.038
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##   Standard errors                           Robust.sem
##
```

```
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|) ci.lower ci.upper
##   Agg =~
##     item1             1.000                               1.000    1.000
##     item2             1.179    0.216    5.465    0.000    0.756    1.602
##     item3             0.942    0.209    4.510    0.000    0.533    1.352
##     item4             1.080    0.203    5.321    0.000    0.682    1.478
##    Std.lv  Std.all
##
##     0.583    0.570
##     0.688    0.524
##     0.550    0.429
##     0.630    0.541
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|) ci.lower ci.upper
##    .item1             0.705    0.091    7.705    0.000    0.526    0.884
##    .item2             1.252    0.142    8.799    0.000    0.973    1.531
##    .item3             1.337    0.138    9.690    0.000    1.066    1.607
##    .item4             0.957    0.128    7.489    0.000    0.707    1.208
##     Agg               0.340    0.091    3.738    0.000    0.162    0.518
##    Std.lv  Std.all
##     0.705    0.675
##     1.252    0.726
##     1.337    0.816
##     0.957    0.707
##     1.000    1.000
```

Using MLR for non-normal data

```
model1<-'Agg=~item1+item2+item3+item4'
model1.est<-cfa(model1, data=Agg_data, estimator='MLR')
summary(model1.est, fit.measures=T, standardized=T, ci=T)
```

```
## lavaan 0.6-5 ended normally after 31 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of free parameters                          8
##
##   Number of observations                           500
##
## Model Test User Model:
##                                               Standard      Robust
##   Test Statistic                                12.998      13.837
##   Degrees of freedom                                 2           2
##   P-value (Chi-square)                           0.002       0.001
##   Scaling correction factor                                  0.939
##     for the Yuan-Bentler correction (Mplus variant)
##
## Model Test Baseline Model:
##
##   Test statistic                               187.030     148.841
##   Degrees of freedom                                 6           6
##   P-value                                        0.000       0.000
##   Scaling correction factor                                  1.257
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.939       0.917
##   Tucker-Lewis Index (TLI)                       0.818       0.751
##
##   Robust Comparative Fit Index (CFI)                         0.938
##   Robust Tucker-Lewis Index (TLI)                            0.814
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)              -3097.350   -3097.350
##   Scaling correction factor                                  2.129
##       for the MLR correction
##   Loglikelihood unrestricted model (H1)      -3090.851   -3090.851
##   Scaling correction factor                                  1.891
##       for the MLR correction
##
##   Akaike (AIC)                                6210.700    6210.700
##   Bayesian (BIC)                              6244.417    6244.417
##   Sample-size adjusted Bayesian (BIC)         6219.024    6219.024
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.105       0.109
##   90 Percent confidence interval - lower         0.056       0.058
##   90 Percent confidence interval - upper         0.162       0.168
##   P-value RMSEA <= 0.05                          0.035       0.030
##
##   Robust RMSEA                                               0.105
##   90 Percent confidence interval - lower                     0.058
##   90 Percent confidence interval - upper                     0.161
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.038       0.038
##
## Parameter Estimates:
##
```

```
##   Information                                      Observed
##   Observed information based on                     Hessian
##   Standard errors                        Robust.huber.white
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|) ci.lower ci.upper
##   Agg =~
##     item1             1.000                               1.000    1.000
##     item2             1.179    0.194    6.090    0.000    0.800    1.559
##     item3             0.942    0.260    3.619    0.000    0.432    1.453
##     item4             1.080    0.249    4.339    0.000    0.592    1.568
##    Std.lv  Std.all
##
##     0.583    0.570
##     0.688    0.524
##     0.550    0.429
##     0.630    0.541
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|) ci.lower ci.upper
##    .item1             0.705    0.098    7.201    0.000    0.513    0.897
##    .item2             1.252    0.148    8.434    0.000    0.961    1.543
##    .item3             1.337    0.143    9.341    0.000    1.056    1.617
##    .item4             0.957    0.134    7.148    0.000    0.695    1.220
##     Agg               0.340    0.099    3.433    0.001    0.146    0.534
##    Std.lv  Std.all
##     0.705    0.675
##     1.252    0.726
##     1.337    0.816
##     0.957    0.707
##     1.000    1.000
```

Summary of what to do when you have non-normal continuous data

- Use MLR to:
  - *Obtain corrected model fit statistics*
  - *Parameter significance tests that are based on corrected standard errors*
  - *The parameter estimates are the same as from normal ML estimation*
- We discussed an example using the cfa() function but this generalises to the sem() function as well

Ordinal-categorical data

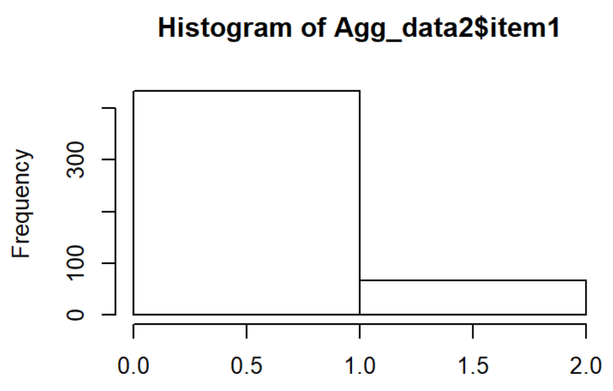
- Ordinal-categorical data is also very common in SEM
  - *Items usually have a Likert-type scale*
  - *When there are less than 5 response options that can adversely affect SEM models*
- Consequences include:
  - *Underestimated correlations between items and associated parameter bias*
    - Lower loadings, factor correlations etc.
  - *Underestimated model fit*

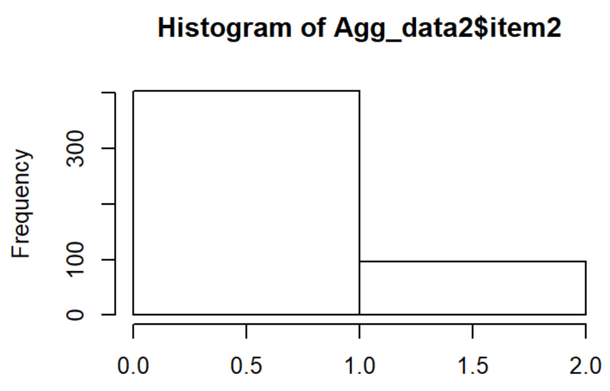
Solutions for ordinal-categorical data

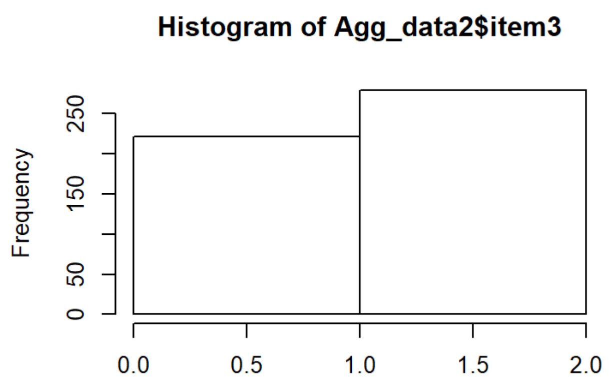
- Use a **categorical estimator** instead of maximum likelihood estimation
- It is possible to treat ordinal-categorical items ‘as if’ there is a latent continuous variable underlying them
- A series of k-1 thresholds on this continuous distribution is imagined
  - *k= number of categories*
    - E.g., for a binary item there is one threshold
    - Those below the threshold get a score of 0, those above get a score of 1 for the item
- It is then possible to fit the model based on the assumed associations between the underlying latent continuous variable

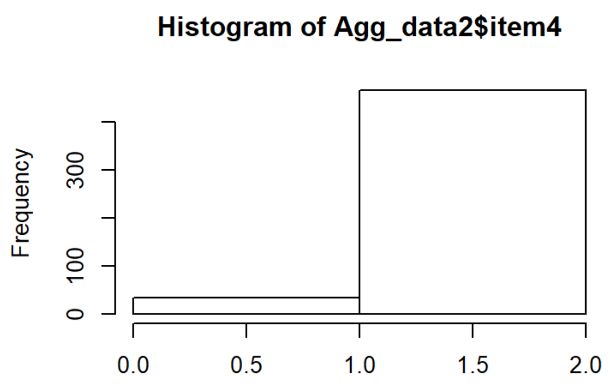
Example: Ordinal data

- Imagine (the more realistic scenario), where our four aggression items are on a binary response scale









Fitting a model with ordinal-categorical data

- Let’s first fit the model ignoring the ordinal-categorical nature of the items

```r
model1<-'Agg=~item1+item2+item3+item4'
model1.est<-cfa(model1, data=Agg_data2)
summary(model1.est, fit.measures=T, standardized=T)
```

```
## lavaan 0.6-5 ended normally after 36 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of free parameters                          8
##
##   Number of observations                           500
##
## Model Test User Model:
##
##   Test statistic                                12.856
##   Degrees of freedom                                 2
##   P-value (Chi-square)                           0.002
##
## Model Test Baseline Model:
##
##   Test statistic                                64.410
##   Degrees of freedom                                 6
##   P-value                                        0.000
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.814
##   Tucker-Lewis Index (TLI)                       0.442
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)               -768.162
##   Loglikelihood unrestricted model (H1)       -761.734
##
##   Akaike (AIC)                                1552.325
##   Bayesian (BIC)                              1586.042
##   Sample-size adjusted Bayesian (BIC)         1560.649
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.104
##   90 Percent confidence interval - lower         0.055
##   90 Percent confidence interval - upper         0.162
##   P-value RMSEA <= 0.05                          0.036
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.045
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##   Standard errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   Agg =~
##     item1              1.000                               0.128    0.375
##     item2              1.413    0.440    3.214    0.001    0.180    0.458
##     item3              1.504    0.457    3.287    0.001    0.192    0.387
##     item4              0.527    0.184    2.868    0.004    0.067    0.267
##
## Variances:
##                     Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .item1              0.100    0.008   11.743    0.000    0.100    0.859
##    .item2              0.123    0.013    9.232    0.000    0.123    0.790
##    .item3              0.210    0.018   11.413    0.000    0.210    0.851
##    .item4              0.059    0.004   13.973    0.000    0.059    0.929
##     Agg                0.016    0.007    2.393    0.017    1.000    1.000
```

Fitting a model with a categorical estimator

- We can request categorical estimation by setting estimator by letting lavaan know which items are categorical
- We do this using the 'ordered' argument in the cfa() function

```
model1<-'Agg=~item1+item2+item3+item4'
model1.est<-cfa(model1, data=Agg_data2, ordered=c('item1','item2','item3','item4'))
summary(model1.est, fit.measures=T, standardized=T)
```

```
## lavaan 0.6-5 ended normally after 24 iterations
##
##   Estimator                                       DWLS
##   Optimization method                           NLMINB
##   Number of free parameters                          8
##
##   Number of observations                           500
##
## Model Test User Model:
##                                               Standard      Robust
##   Test Statistic                                 7.257       7.807
##   Degrees of freedom                                 2           2
##   P-value (Chi-square)                           0.027       0.020
##   Scaling correction factor                                  0.943
##   Shift parameter                                            0.115
##     for the simple second-order correction
##
## Model Test Baseline Model:
##
##   Test statistic                                80.264      75.274
##   Degrees of freedom                                 6           6
##   P-value                                        0.000       0.000
##   Scaling correction factor                                  1.072
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.929       0.916
##   Tucker-Lewis Index (TLI)                       0.788       0.749
##
##   Robust Comparative Fit Index (CFI)                            NA
##   Robust Tucker-Lewis Index (TLI)                               NA
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.073       0.076
##   90 Percent confidence interval - lower         0.021       0.026
##   90 Percent confidence interval - upper         0.133       0.136
##   P-value RMSEA <= 0.05                          0.194       0.166
##
##   Robust RMSEA                                                  NA
##   90 Percent confidence interval - lower                        NA
##   90 Percent confidence interval - upper                        NA
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.089       0.089
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model          Unstructured
##   Standard errors                           Robust.sem
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   Agg =~
##     item1             1.000                               0.516    0.516
##     item2             1.115    0.296    3.767    0.000    0.576    0.576
##     item3             1.102    0.314    3.504    0.000    0.569    0.569
##     item4             1.300    0.393    3.308    0.001    0.671    0.671
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .item1             0.000                               0.000    0.000
##    .item2             0.000                               0.000    0.000
##    .item3             0.000                               0.000    0.000
##    .item4             0.000                               0.000    0.000
##     Agg               0.000                               0.000    0.000
##
## Thresholds:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     item1|t1          1.108    0.071   15.690    0.000    1.108    1.108
##     item2|t1          0.871    0.065   13.484    0.000    0.871    0.871
##     item3|t1         -0.146    0.056   -2.590    0.010   -0.146   -0.146
##     item4|t1         -1.491    0.086  -17.370    0.000   -1.491   -1.491
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .item1             0.733                               0.733    0.733
##    .item2             0.669                               0.669    0.669
##    .item3             0.676                               0.676    0.676
##    .item4             0.550                               0.550    0.550
##     Agg               0.267    0.106    2.526    0.012    1.000    1.000
##
## Scales y*:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     item1             1.000                               1.000    1.000
##     item2             1.000                               1.000    1.000
##     item3             1.000                               1.000    1.000
##     item4             1.000                               1.000    1.000
```

Summary of what to do with ordered-categorical items

- Declare your items as ‘ordered’ and use a categorical estimator
- This gives you parameter estimates/fit statistics ‘as if’ analysing the associations between underlying continous variables

Missing data

- Missing data is a near-universal feature of real data
- It reduces our sample size
  - *Less precision and statistical power*
- If missingness is ‘non-random’ it can also bias our parameter estimates

Missing data mechanisms

- Rubin defined three different possible missing data mechanisms:
    - ***Missing completely at random (MCAR)***
    - ***Missing at random (MAR)***
    - ***Missing not at random (MNAR)***
- The effects of missingness depend on a combination of the missing data mechanism and the method we use to deal with it

MAR

- Missing at random (MAR) means:
  - *When the probability of missing data on a variable Y is related to other variables in the model but not to the values of Y itself*
- Example:
  - *X = self-control and Y= aggression.*
  - *People with lower self-control are more likely to have missing data on aggression*
  - *After taking into account self-control, people who are high in aggression are no more likely to have missing data on aggression*
  - *Challenge is that there is no way to confirm that there is no relation between aggression scores and missing data on aggression because that would knowledge of the missing scores*
- Common misconception is that Little’s test can be used to confirm MAR

MCAR

- Genuinely random missingness
- No relation between Y or any other variable in the model and missingness on Y
- The data you have are a simple random sample of the complete data
- The ideal missing data scenario!
- Example:
  - *X = self-control and Y= aggression*
  - *People of all levels of self-control and aggression are equally likely to have missing data on aggression*

MNAR

- Missing not random (MNAR) means:
- When the probability of missingness on Y is related to the values of Y itself
- Example:
  - *X= self-control, Y= aggression*
  - *Those high in aggression are more likely to have missing data on the aggression variable, even after taking into account self-control*
- As with MAR, there is no way to verify that data are MNAR without knowledge of the missing values

Methods of dealing with missingness

- Deletion methods:
  - *Listwise deletion*
  - *Pairwise deletion*
- Imputation methods:
  - *Mean imputation*
  - *Regression imputation*
  - *Multiple imputation*
- Maximum likelihood estimation
- Methods for MNAR data:
  - *Pattern mixture models*
  - *Random coefficient models*

Listwise deletion

- ‘complete case analysis’
- Example:
  - *Delete everyone from the analysis who has missing data on either self-control or aggression*
- Will give biased results unless data are MCAR
- Even if data are MCAR, power will be reduced by reducing the sample size
- Bottom line: **not recommended**

Pairwise deletion

- ‘available-case analysis’
- Uses available data for each analysis
- Different cases contribute to different correlations in a correlation matrix
- Example:
    - *Cases 2,3,7,18, 56, 100 not used in the self-control- aggression correlation*
    - *Cases 2,7,18,77, 103 not used in the aggression-substance use correlation*
- Doesn’t reduce power as much as listwise deletion
- But still gives biased results whenever data are not MCAR
- Bottom line: **not recommended**

Mean imputation

- Replace missing values with the mean on that variable
- Example:
  - *Replace missing aggression values with the mean of the aggression variable*
- Two major issues:
  - *artificially reduces the variability of the data*
  - *can give very biased estimates even when data are MCAR*
- Bottom line: **not recommended**
- 'possibly the worst missing data handling method available… you should absolutely avoid this approach' Enders (2011)

Regression imputation

- Replaces the missing values with values predicted from a regression
- Estimate a set of regression equations where the incomplete variables are predicted from the complete variables
- Use the regression equations to calculate the predicted values on the incomplete variables
- Based on the principle of using information from the complete data to estimate the missing data
- Two forms:
  - *‘normal’ regression imputation*
  - *Stochastic regression imputation (adds a residual term to overcome loss of variance)*
- Stochastic regression is preferred and gives unbiased results if data are MAR

Multiple imputation

- Procedure:
  - *Imputes missing data several times to create multiple complete datasets*
  - *Can include many more predictor variables in imputation model than analysis model*
  - *Analyses are conducted for each dataset*
  - *Analysis results are pooled across datasets to get parameter estimates and standard errors*
  - *3-5 datasets are often enough but more is better and 20+ is ideal*
- Unlike in most single imputation approaches, the standard errors take account of the additional uncertainty due to missingness
- Gives unbiased parameter estimates under MAR
- Can be cumbersome if pooling has to be done by hand but usually pooling can be automated
- Bottom line: **recommended** method if data are likely to be MAR

Full Information Maximum Likelihood (FIML) Estimation method

- Makes use of all the information in the model to arrive at the parameter estimates ‘as if’ the data were complete
- Does not ‘impute’ individual values
- Gives unbiased estimates under MAR
- Even under MCAR it is superior to listwise and pairwise deletion because it uses more information from the observed data
- Practical advantage= easy to implement, usually much more so than multiple imputation
- Bottom line: **recommended** method if data are likely to be MAR

Missingness in lavaan

- The default method of dealing with missingness in lavaan is listwise deletion
- If data are continuous we can easily switch to one of the recommended methods: FIML

```
model1<-'Agg=~item1+item2+item3+item4'
model1.est<-cfa(model1, data=Agg_data, missing='ML')
summary(model1.est, fit.measures=T, standardized=T)
```

```
## lavaan 0.6-5 ended normally after 31 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of free parameters                         12
##
##   Number of observations                           500
##   Number of missing patterns                         1
##
## Model Test User Model:
##
##   Test statistic                                12.998
##   Degrees of freedom                                 2
##   P-value (Chi-square)                           0.002
##
## Model Test Baseline Model:
##
##   Test statistic                               187.030
##   Degrees of freedom                                 6
##   P-value                                        0.000
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.939
##   Tucker-Lewis Index (TLI)                       0.818
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)              -3097.350
##   Loglikelihood unrestricted model (H1)      -3090.851
##
##   Akaike (AIC)                                6218.700
##   Bayesian (BIC)                              6269.275
##   Sample-size adjusted Bayesian (BIC)         6231.186
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.105
##   90 Percent confidence interval - lower         0.056
##   90 Percent confidence interval - upper         0.162
##   P-value RMSEA <= 0.05                          0.035
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.032
##
## Parameter Estimates:
##
##   Information                                 Observed
##   Observed information based on                Hessian
##   Standard errors                             Standard
##
```

```
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   Agg =~
##     item1             1.000                               0.583    0.570
##     item2             1.179    0.171    6.911    0.000    0.688    0.524
##     item3             0.942    0.179    5.276    0.000    0.550    0.429
##     item4             1.080    0.180    6.010    0.000    0.630    0.541
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .item1             0.011    0.046    0.245    0.807    0.011    0.011
##    .item2             0.050    0.059    0.851    0.395    0.050    0.038
##    .item3             0.172    0.057    3.004    0.003    0.172    0.134
##    .item4             0.105    0.052    2.024    0.043    0.105    0.091
##     Agg               0.000                               0.000    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .item1             0.705    0.070   10.127    0.000    0.705    0.675
##    .item2             1.252    0.110   11.346    0.000    1.252    0.726
##    .item3             1.337    0.103   13.001    0.000    1.337    0.816
##    .item4             0.957    0.090   10.656    0.000    0.957    0.707
##     Agg               0.340    0.072    4.699    0.000    1.000    1.000
```

Advanced topics in missingness

- Missingness with ordered-categorical data:
  - *Unfortunately the FIML option is not available with ordered-categorical data*
  - *Instead we must use multiple imputation*
    - mice package + survey.lavaan package
    - mice can create the imputed datasets for us
    - survey.lavaan can be used to pool them
- MNAR data:
- If we suspect that the data are not MAR or MCAR there are other options available
  - *Random coefficient models*
  - *Selection models*
- However, these methods are based on strong, untestable assumptions
- If these assumptions are false, can result in even greater bias
- Many, therefore, argue that multiple imputation or MAR are best even if MNAR is suspected

Summary of missingess

- The lavaan default is listwise deletion but this is a suboptimal method
  - *Biased parameter estimates unless data are MCAR*
  - *Inefficient even if data are MCAR*
- FIML can be easily used with continuous data
- Multiple imputation (not covered in this course) is needed for categorical data