Missing data and inclusion of (co)variances of exogeneous variables

Dealing with missing data

We will analyse the Holzinger Swineford data included in the **lavaan** package.

```
library("lavaan")
summary(HolzingerSwineford1939)
```

```
##        id              sex            ageyr        agemo
##  Min.   :  1.0   Min.   :1.000   Min.   :11   Min.   : 0.000
##  1st Qu.: 82.0   1st Qu.:1.000   1st Qu.:12   1st Qu.: 2.000
##  Median :163.0   Median :2.000   Median :13   Median : 5.000
##  Mean   :176.6   Mean   :1.515   Mean   :13   Mean   : 5.375
##  3rd Qu.:272.0   3rd Qu.:2.000   3rd Qu.:14   3rd Qu.: 8.000
##  Max.   :351.0   Max.   :2.000   Max.   :16   Max.   :11.000
##
##          school        grade             x1               x2
##  Grant-White:145   Min.   :7.000   Min.   :0.6667   Min.   :2.250
##  Pasteur    :156   1st Qu.:7.000   1st Qu.:4.1667   1st Qu.:5.250
##                    Median :7.000   Median :5.0000   Median :6.000
##                    Mean   :7.477   Mean   :4.9358   Mean   :6.088
##                    3rd Qu.:8.000   3rd Qu.:5.6667   3rd Qu.:6.750
##                    Max.   :8.000   Max.   :8.5000   Max.   :9.250
##                    NA's   :1
##        x3              x4              x5              x6
##  Min.   :0.250   Min.   :0.000   Min.   :1.000   Min.   :0.1429
##  1st Qu.:1.375   1st Qu.:2.333   1st Qu.:3.500   1st Qu.:1.4286
##  Median :2.125   Median :3.000   Median :4.500   Median :2.0000
##  Mean   :2.250   Mean   :3.061   Mean   :4.341   Mean   :2.1856
##  3rd Qu.:3.125   3rd Qu.:3.667   3rd Qu.:5.250   3rd Qu.:2.7143
##  Max.   :4.500   Max.   :6.333   Max.   :7.000   Max.   :6.1429
##
##        x7              x8              x9
##  Min.   :1.304   Min.   : 3.050   Min.   :2.778
##  1st Qu.:3.478   1st Qu.: 4.850   1st Qu.:4.750
##  Median :4.087   Median : 5.500   Median :5.417
##  Mean   :4.186   Mean   : 5.527   Mean   :5.374
##  3rd Qu.:4.913   3rd Qu.: 6.100   3rd Qu.:6.083
##  Max.   :7.435   Max.   :10.000   Max.   :9.250
##
```

```
nrow(HolzingerSwineford1939)
```

```
## [1] 301
```

See `?HolzingerSwineford` for more info.

We will fit a three-factor CFA model to the x variables in the dataset:

```
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'
```

Benchmark: Complete data

```
CD_fit <- cfa(HS.model, data = HolzingerSwineford1939, meanstructure = TRUE)
(CD_summ <- summary(CD_fit, standardized = TRUE))
```

```
## lavaan 0.6-18 ended normally after 35 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        30
##
##   Number of observations                           301
##
## Model Test User Model:
##
##   Test statistic                                85.306
##   Degrees of freedom                                24
##   P-value (Chi-square)                           0.000
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   visual =~
##     x1                1.000                               0.900    0.772
##     x2                0.554    0.100    5.554    0.000    0.498    0.424
##     x3                0.729    0.109    6.685    0.000    0.656    0.581
##   textual =~
##     x4                1.000                               0.990    0.852
##     x5                1.113    0.065   17.014    0.000    1.102    0.855
##     x6                0.926    0.055   16.703    0.000    0.917    0.838
##   speed =~
##     x7                1.000                               0.619    0.570
##     x8                1.180    0.165    7.152    0.000    0.731    0.723
##     x9                1.082    0.151    7.155    0.000    0.670    0.665
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
```

```
##   visual ~~
##     textual           0.408    0.074    5.552    0.000    0.459    0.459
##     speed             0.262    0.056    4.660    0.000    0.471    0.471
##   textual ~~
##     speed             0.173    0.049    3.518    0.000    0.283    0.283
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .x1                4.936    0.067   73.473    0.000    4.936    4.235
##    .x2                6.088    0.068   89.855    0.000    6.088    5.179
##    .x3                2.250    0.065   34.579    0.000    2.250    1.993
##    .x4                3.061    0.067   45.694    0.000    3.061    2.634
##    .x5                4.341    0.074   58.452    0.000    4.341    3.369
##    .x6                2.186    0.063   34.667    0.000    2.186    1.998
##    .x7                4.186    0.063   66.766    0.000    4.186    3.848
##    .x8                5.527    0.058   94.854    0.000    5.527    5.467
##    .x9                5.374    0.058   92.546    0.000    5.374    5.334
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .x1                0.549    0.114    4.833    0.000    0.549    0.404
##    .x2                1.134    0.102   11.146    0.000    1.134    0.821
##    .x3                0.844    0.091    9.317    0.000    0.844    0.662
##    .x4                0.371    0.048    7.779    0.000    0.371    0.275
##    .x5                0.446    0.058    7.642    0.000    0.446    0.269
##    .x6                0.356    0.043    8.277    0.000    0.356    0.298
##    .x7                0.799    0.081    9.823    0.000    0.799    0.676
##    .x8                0.488    0.074    6.573    0.000    0.488    0.477
##    .x9                0.566    0.071    8.003    0.000    0.566    0.558
##     visual            0.809    0.145    5.564    0.000    1.000    1.000
##     textual           0.979    0.112    8.737    0.000    1.000    1.000
##     speed             0.384    0.086    4.451    0.000    1.000    1.000
```

```
fit.inds <- c("chisq", "df", "pvalue", "cfi", "rmsea", "srmr", "aic", "bic")
(CD_fitm <- fitmeasures(CD_fit, fit.inds))
```

```
##    chisq       df   pvalue      cfi    rmsea     srmr      aic      bic
##   85.306   24.000    0.000    0.931    0.092    0.060 7535.490 7646.703
```

Create missing values

We introduce some missing values. The values will be missing completely at random, with a probability of .2 for any value being missing:

```
HSMiss <- HolzingerSwineford1939[,paste("x", 1:9, sep="")]
set.seed(42)
randomMiss <- rbinom(prod(dim(HSMiss)), 1, 0.20)
randomMiss <- matrix(as.logical(randomMiss), nrow=nrow(HSMiss))
HSMiss[randomMiss] <- NA
head(HSMiss)
```

```
##         x1   x2    x3       x4   x5        x6       x7   x8       x9
```

```
## 1       NA 7.75 0.375 2.333333   NA 1.2857143 3.391304   NA       NA
## 2       NA 5.25 2.125 1.666667   NA 1.2857143 3.782609 6.25 7.916667
## 3 4.500000 5.25 1.875       NA 1.75 0.4285714       NA 3.90       NA
## 4       NA   NA 3.000 2.666667 4.50 2.4285714 3.000000 5.30 4.861111
## 5 4.833333   NA 0.875 2.666667 4.00 2.5714286 3.695652   NA 5.916667
## 6 5.333333 5.00 2.250 1.000000 3.00 0.8571429 4.347826 6.65 7.500000
```

Some values are now missing.

Listwise deletion approach

By default, functions `cfa()`, `growth()`, `lavaan()` and `sem()` will remove every observation with missing values, which drastically decreases sample size:

```
LD_fit <- cfa(HS.model, data = HSMiss, meanstructure = TRUE)
```

```
## Warning: lavaan->lav_object_post_check():
##    covariance matrix of latent variables is not positive definite ; use
##    lavInspect(fit, "cov.lv") to investigate.
```

```
## lavInspect(LD_fit, "cov.lv") ## suppressed, because same output given below
(LD_summ <- summary(LD_fit, standardized = TRUE))
```

```
## lavaan 0.6-18 ended normally after 40 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        30
##
##                                                   Used       Total
##   Number of observations                            50         301
##
## Model Test User Model:
##
##   Test statistic                                22.454
##   Degrees of freedom                                24
##   P-value (Chi-square)                           0.552
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   visual =~
##     x1                1.000                               0.709    0.526
##     x2                0.660    0.226    2.916    0.004    0.468    0.470
##     x3                0.558    0.236    2.358    0.018    0.395    0.359
##   textual =~
##     x4                1.000                               1.168    0.951
```

```
##     x5                0.965    0.131    7.352    0.000    1.127    0.804
##     x6                0.764    0.111    6.905    0.000    0.892    0.773
##   speed =~
##     x7                1.000                               0.344    0.384
##     x8                1.289    0.603    2.140    0.032    0.443    0.425
##     x9                2.796    1.465    1.908    0.056    0.961    0.971
##
## Covariances:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   visual ~~
##     textual           0.774    0.248    3.123    0.002    0.935    0.935
##     speed             0.152    0.100    1.519    0.129    0.624    0.624
##   textual ~~
##     speed             0.061    0.069    0.885    0.376    0.152    0.152
##
## Intercepts:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .x1                4.877    0.191   25.562    0.000    4.877    3.615
##    .x2                5.895    0.141   41.889    0.000    5.895    5.924
##    .x3                2.038    0.156   13.070    0.000    2.038    1.848
##    .x4                2.747    0.174   15.810    0.000    2.747    2.236
##    .x5                4.165    0.198   21.001    0.000    4.165    2.970
##    .x6                2.186    0.163   13.389    0.000    2.186    1.894
##    .x7                4.383    0.127   34.621    0.000    4.383    4.896
##    .x8                5.707    0.147   38.699    0.000    5.707    5.473
##    .x9                5.424    0.140   38.757    0.000    5.424    5.481
##
## Variances:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .x1                1.317    0.304    4.334    0.000    1.317    0.724
##    .x2                0.771    0.168    4.603    0.000    0.771    0.779
##    .x3                1.059    0.216    4.894    0.000    1.059    0.871
##    .x4                0.144    0.111    1.297    0.195    0.144    0.096
##    .x5                0.696    0.175    3.987    0.000    0.696    0.354
##    .x6                0.537    0.127    4.227    0.000    0.537    0.403
##    .x7                0.683    0.145    4.709    0.000    0.683    0.853
##    .x8                0.891    0.195    4.561    0.000    0.891    0.819
##    .x9                0.055    0.369    0.150    0.881    0.055    0.056
##     visual            0.503    0.293    1.715    0.086    1.000    1.000
##     textual           1.365    0.319    4.278    0.000    1.000    1.000
##     speed             0.118    0.097    1.219    0.223    1.000    1.000
```

```
(LD_fitm <- fitmeasures(LD_fit, fit.inds))
```

```
##    chisq       df   pvalue      cfi    rmsea     srmr      aic      bic
##   22.454   24.000    0.552    1.000    0.000    0.076 1298.143 1355.503
```

Using listwise deletion, we have only 50 observations left. We also get warnings about the (co)variance matrix of the latent variables being problematic. Furthermore, the estimated loadings, intercepts and variances seem more variable than before.

Multiple imputation approach

We now impute the data using package **mice**. We use generate five imputed datasets and use the predictive mean matching method, which is the current state of the art in missing data imputation (although many would suggest using a higher value for m; increasing m will never hurt quality of results, but it will make computations longer, so for the current simple example I opt for computation speed):

```
library("mice")
m <- 5
set.seed(42)
imp_data <- mice(HSMiss, m = m, method = "pmm")
```

```
##
##  iter imp variable
##   1   1  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   1   2  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   1   3  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   1   4  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   1   5  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   2   1  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   2   2  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   2   3  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   2   4  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   2   5  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   3   1  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   3   2  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   3   3  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   3   4  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   3   5  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   4   1  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   4   2  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   4   3  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   4   4  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   4   5  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   5   1  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   5   2  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   5   3  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   5   4  x1  x2  x3  x4  x5  x6  x7  x8  x9
##   5   5  x1  x2  x3  x4  x5  x6  x7  x8  x9
```

We extract the imputed datasets using function `complete()` and save them in a list:

```
data_list <- list()
for (i in 1:m) data_list[[i]] <- complete(imp_data, action = i)
lapply(data_list, head) ## show first few rows of every imputed dataset
```

```
## [[1]]
##         x1   x2    x3       x4   x5        x6       x7   x8       x9
## 1 3.666667 7.75 0.375 2.333333 3.00 1.2857143 3.391304 4.70 3.777778
## 2 5.666667 5.25 2.125 1.666667 3.00 1.2857143 3.782609 6.25 7.916667
## 3 4.500000 5.25 1.875 1.000000 1.75 0.4285714 3.130435 3.90 3.361111
## 4 5.833333 7.25 3.000 2.666667 4.50 2.4285714 3.000000 5.30 4.861111
## 5 4.833333 5.00 0.875 2.666667 4.00 2.5714286 3.695652 6.30 5.916667
```

```
## 6 5.333333 5.00 2.250 1.000000 3.00 0.8571429 4.347826 6.65 7.500000
##
## [[2]]
##         x1   x2    x3       x4   x5        x6       x7   x8       x9
## 1 4.166667 7.75 0.375 2.333333 5.25 1.2857143 3.391304 6.40 7.916667
## 2 5.333333 5.25 2.125 1.666667 4.25 1.2857143 3.782609 6.25 7.916667
## 3 4.500000 5.25 1.875 1.666667 1.75 0.4285714 3.695652 3.90 5.500000
## 4 4.500000 5.75 3.000 2.666667 4.50 2.4285714 3.000000 5.30 4.861111
## 5 4.833333 8.00 0.875 2.666667 4.00 2.5714286 3.695652 4.30 5.916667
## 6 5.333333 5.00 2.250 1.000000 3.00 0.8571429 4.347826 6.65 7.500000
##
## [[3]]
##         x1   x2    x3       x4   x5        x6       x7   x8       x9
## 1 3.833333 7.75 0.375 2.333333 3.00 1.2857143 3.391304 4.50 3.333333
## 2 6.000000 5.25 2.125 1.666667 2.75 1.2857143 3.782609 6.25 7.916667
## 3 4.500000 5.25 1.875 2.666667 1.75 0.4285714 3.956522 3.90 4.833333
## 4 4.500000 5.75 3.000 2.666667 4.50 2.4285714 3.000000 5.30 4.861111
## 5 4.833333 5.25 0.875 2.666667 4.00 2.5714286 3.695652 9.10 5.916667
## 6 5.333333 5.00 2.250 1.000000 3.00 0.8571429 4.347826 6.65 7.500000
##
## [[4]]
##         x1   x2    x3       x4   x5        x6       x7   x8       x9
## 1 2.666667 7.75 0.375 2.333333 4.25 1.2857143 3.391304 5.10 3.361111
## 2 4.166667 5.25 2.125 1.666667 4.50 1.2857143 3.782609 6.25 7.916667
## 3 4.500000 5.25 1.875 2.000000 1.75 0.4285714 2.043478 3.90 5.083333
## 4 3.166667 8.00 3.000 2.666667 4.50 2.4285714 3.000000 5.30 4.861111
## 5 4.833333 5.25 0.875 2.666667 4.00 2.5714286 3.695652 6.95 5.916667
## 6 5.333333 5.00 2.250 1.000000 3.00 0.8571429 4.347826 6.65 7.500000
##
## [[5]]
##         x1   x2    x3       x4   x5        x6       x7   x8       x9
## 1 4.833333 7.75 0.375 2.333333 3.00 1.2857143 3.391304 4.00 4.416667
## 2 3.166667 5.25 2.125 1.666667 3.25 1.2857143 3.782609 6.25 7.916667
## 3 4.500000 5.25 1.875 1.666667 1.75 0.4285714 2.000000 3.90 5.083333
## 4 4.833333 8.75 3.000 2.666667 4.50 2.4285714 3.000000 5.30 4.861111
## 5 4.833333 6.50 0.875 2.666667 4.00 2.5714286 3.695652 5.35 5.916667
## 6 5.333333 5.00 2.250 1.000000 3.00 0.8571429 4.347826 6.65 7.500000
```

We see that the missing values have been imputed with different values in every dataset.

Now we use the `cfa.mi()` function to fit a CFA model on the imputed data:

```
library("lavaan.mi")
```

```
MI_fit <- cfa.mi(HS.model, data_list, meanstructure = TRUE)
(MI_summ <- summary(MI_fit, standardized = TRUE))
```

```
## lavaan.mi object fit to 5 imputed data sets using:
##  - lavaan    (0.6-18)
##  - lavaan.mi (0.1-0)
## See class?lavaan.mi help page for available methods.
##
## Convergence information:
```

```
## The model converged on 5 imputed data sets.
## Standard errors were available for all imputations.
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        30
##
##   Number of observations                           301
##
## Model Test User Model:
##
##   Test statistic                                43.546
##   Degrees of freedom                                24
##   P-value                                        0.009
##   Pooling method                                    D4
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
##   Pooled across imputations              Rubin's (1987) rules
##   Augment within-imputation variance     Scale by average RIV
##   Wald test for pooled parameters          t(df) distribution
##
##   Pooled t statistics with df >= 1000 are displayed with
##   df = Inf(inity) to save space. Although the t distribution
##   with large df closely approximates a standard normal
##   distribution, exact df for reporting these t tests can be
##   obtained from parameterEstimates.mi()
##
##
## Latent Variables:
##                    Estimate  Std.Err  t-value       df  P(>|t|)   Std.lv
##   visual =~
##     x1                1.000                                        0.959
##     x2                0.429    0.104    4.110   20.098    0.001    0.412
##     x3                0.591    0.107    5.532   20.155    0.000    0.567
##   textual =~
##     x4                1.000                                        0.961
##     x5                1.132    0.081   13.900   75.696    0.000    1.088
##     x6                0.936    0.068   13.703   34.335    0.000    0.900
##   speed =~
##     x7                1.000                                        0.541
##     x8                1.126    0.214    5.250  647.659    0.000    0.609
##     x9                1.444    0.274    5.270   22.627    0.000    0.782
##   Std.all
##
##     0.821
##     0.352
##     0.524
##
##     0.849
```

```
##     0.846
##     0.834
##
##     0.504
##     0.615
##     0.735
##
## Covariances:
##                   Estimate  Std.Err  t-value       df  P(>|t|)   Std.lv
##   visual ~~
##     textual          0.495    0.090    5.474      Inf    0.000    0.537
##     speed            0.292    0.070    4.182  163.670    0.000    0.563
##   textual ~~
##     speed            0.169    0.054    3.122  297.158    0.002    0.324
##   Std.all
##
##     0.537
##     0.563
##
##     0.324
##
## Intercepts:
##                   Estimate  Std.Err  t-value       df  P(>|t|)   Std.lv
##    .x1               4.952    0.080   61.554      Inf    0.000    4.952
##    .x2               6.123    0.081   76.022      Inf    0.000    6.123
##    .x3               2.200    0.075   29.527   54.113    0.000    2.200
##    .x4               3.009    0.078   38.614  117.055    0.000    3.009
##    .x5               4.324    0.088   48.866  317.076    0.000    4.324
##    .x6               2.162    0.074   29.104   92.839    0.000    2.162
##    .x7               4.184    0.074   56.562   80.495    0.000    4.184
##    .x8               5.512    0.068   80.810  157.431    0.000    5.512
##    .x9               5.357    0.073   73.302  110.788    0.000    5.357
##   Std.all
##     4.237
##     5.237
##     2.032
##     2.658
##     3.363
##     2.003
##     3.893
##     5.566
##     5.037
##
## Variances:
##                   Estimate  Std.Err  t-value       df  P(>|t|)   Std.lv
##    .x1               0.446    0.144    3.106   43.694    0.003    0.446
##    .x2               1.197    0.123    9.754   41.235    0.000    1.197
##    .x3               0.850    0.097    8.730   52.643    0.000    0.850
##    .x4               0.358    0.055    6.486   87.401    0.000    0.358
##    .x5               0.469    0.071    6.591  161.355    0.000    0.469
##    .x6               0.355    0.051    6.936   22.414    0.000    0.355
##    .x7               0.861    0.099    8.738   78.258    0.000    0.861
##    .x8               0.609    0.084    7.268   13.055    0.000    0.609
##    .x9               0.520    0.105    4.973   10.066    0.001    0.520
```

```
##     visual            0.920    0.186    4.953   81.836    0.000    1.000
##     textual           0.924    0.127    7.261  201.440    0.000    1.000
##     speed             0.293    0.091    3.220  164.978    0.002    1.000
##   Std.all
##     0.327
##     0.876
##     0.726
##     0.279
##     0.284
##     0.304
##     0.746
##     0.621
##     0.460
##     1.000
##     1.000
##     1.000
```

We see that fitting a SEM model on imputed data is quite straightforward: we use function `cfa.mi()` instead of `cfa()`. Using function `summary()`, we obtain the pooled result. The output is very similar to what were used to with a single dataset. One of the differences is that with imputed data, we get t instead of z statistics for every parameter estimate.

```
(MI_fitm <- fitmeasures(MI_fit))
```

```
##
## Test statistic(s) pooled using the D4 pooling method.
##   Pooled statistic: "standard"
##
##                  npar                  fmin                 chisq
##                30.000                 0.117                43.546
##                    df                pvalue        baseline.chisq
##                24.000                 0.009               516.343
##           baseline.df       baseline.pvalue                   cfi
##                36.000                 0.000                 0.959
##                   tli                  nnfi                   rfi
##                 0.939                 0.939                 0.873
##                   nfi                  pnfi                   ifi
##                 0.916                 0.610                 0.960
##                   rni                  logl     unrestricted.logl
##                 0.959             -3733.405             -3694.909
##                   aic                   bic                ntotal
##              7526.811              7638.024              301.000
##                  bic2                 rmsea        rmsea.ci.lower
##              7542.881                 0.052                 0.026
##        rmsea.ci.upper        rmsea.ci.level          rmsea.pvalue
##                 0.076                 0.900                 0.416
##        rmsea.close.h0 rmsea.notclose.pvalue     rmsea.notclose.h0
##                 0.050                 0.027                 0.080
##                   rmr            rmr_nomean                  srmr
##                 0.067                 0.073                 0.055
##          srmr_bentler   srmr_bentler_nomean                  crmr
##                 0.055                 0.059                 0.057
##           crmr_nomean            srmr_mplus     srmr_mplus_nomean
```

```
##                 0.063                 0.059                 0.059
##                 cn_05                 cn_01                   gfi
##               252.708               298.085                 0.997
##                  agfi                  pgfi                   mfi
##                 0.992                 0.443                 0.968
##                  ecvi
##                 0.344
```

```
FIML_fit <- cfa(HS.model, data = HSMiss, missing = "ml")
(FIML_summ <- summary(FIML_fit, standardized = TRUE))
```

Full information Maximum Likelihood (FIML)

```
## lavaan 0.6-18 ended normally after 55 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        30
##
##   Number of observations                           301
##   Number of missing patterns                       104
##
## Model Test User Model:
##
##   Test statistic                                46.119
##   Degrees of freedom                                24
##   P-value (Chi-square)                           0.004
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Observed
##   Observed information based on                Hessian
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   visual =~
##     x1                1.000                               0.948    0.808
##     x2                0.482    0.113    4.270    0.000    0.457    0.388
##     x3                0.633    0.115    5.503    0.000    0.599    0.545
##   textual =~
##     x4                1.000                               0.950    0.850
##     x5                1.154    0.083   13.872    0.000    1.096    0.854
##     x6                0.951    0.069   13.808    0.000    0.904    0.840
##   speed =~
##     x7                1.000                               0.512    0.484
##     x8                1.122    0.209    5.355    0.000    0.574    0.578
##     x9                1.586    0.410    3.869    0.000    0.812    0.778
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
```

```
##   visual ~~
##     textual           0.467    0.084    5.543    0.000    0.518    0.518
##     speed             0.274    0.063    4.370    0.000    0.564    0.564
##   textual ~~
##     speed             0.157    0.049    3.242    0.001    0.324    0.324
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .x1                4.949    0.073   67.953    0.000    4.949    4.221
##    .x2                6.136    0.075   81.358    0.000    6.136    5.207
##    .x3                2.212    0.069   31.891    0.000    2.212    2.010
##    .x4                3.014    0.068   44.535    0.000    3.014    2.698
##    .x5                4.319    0.077   55.835    0.000    4.319    3.365
##    .x6                2.167    0.065   33.320    0.000    2.167    2.014
##    .x7                4.176    0.068   61.263    0.000    4.176    3.951
##    .x8                5.501    0.064   86.193    0.000    5.501    5.536
##    .x9                5.361    0.066   81.613    0.000    5.361    5.140
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .x1                0.477    0.140    3.402    0.001    0.477    0.347
##    .x2                1.179    0.119    9.873    0.000    1.179    0.849
##    .x3                0.852    0.100    8.530    0.000    0.852    0.703
##    .x4                0.346    0.056    6.158    0.000    0.346    0.277
##    .x5                0.447    0.071    6.257    0.000    0.447    0.271
##    .x6                0.341    0.051    6.686    0.000    0.341    0.295
##    .x7                0.855    0.107    8.022    0.000    0.855    0.766
##    .x8                0.658    0.098    6.729    0.000    0.658    0.666
##    .x9                0.429    0.141    3.044    0.002    0.429    0.394
##     visual            0.898    0.178    5.053    0.000    1.000    1.000
##     textual           0.902    0.114    7.898    0.000    1.000    1.000
##     speed             0.262    0.097    2.710    0.007    1.000    1.000
```

```
(FIML_fitm <- fitmeasures(LD_fit, fit.inds))
```

```
##    chisq       df   pvalue      cfi    rmsea     srmr      aic      bic
##   22.454   24.000    0.552    1.000    0.000    0.076 1298.143 1355.503
```

Comparison of methods

We compare parameter estimates and standard errors between the complete dataset, listwise deletion, multiple imputation and FIML:

```
comp_data <- data.frame(
  method = rep(c("CD", "LD", "MI", "FIML"), each = nrow(LD_summ$pe)),
  rbind(
    CD_summ$pe[c(1:3, 5:6)], ## parameter estimates from listw. deletion model
    LD_summ$pe[c(1:3, 5:6)],
    MI_summ$pe[c(1:3, 5:6)],
    FIML_summ$pe[c(1:3, 5:6)]
  )
)
```

```
comp_data <- comp_data[comp_data$se > 0, ] ## omit fixed parameters
head(comp_data)
```

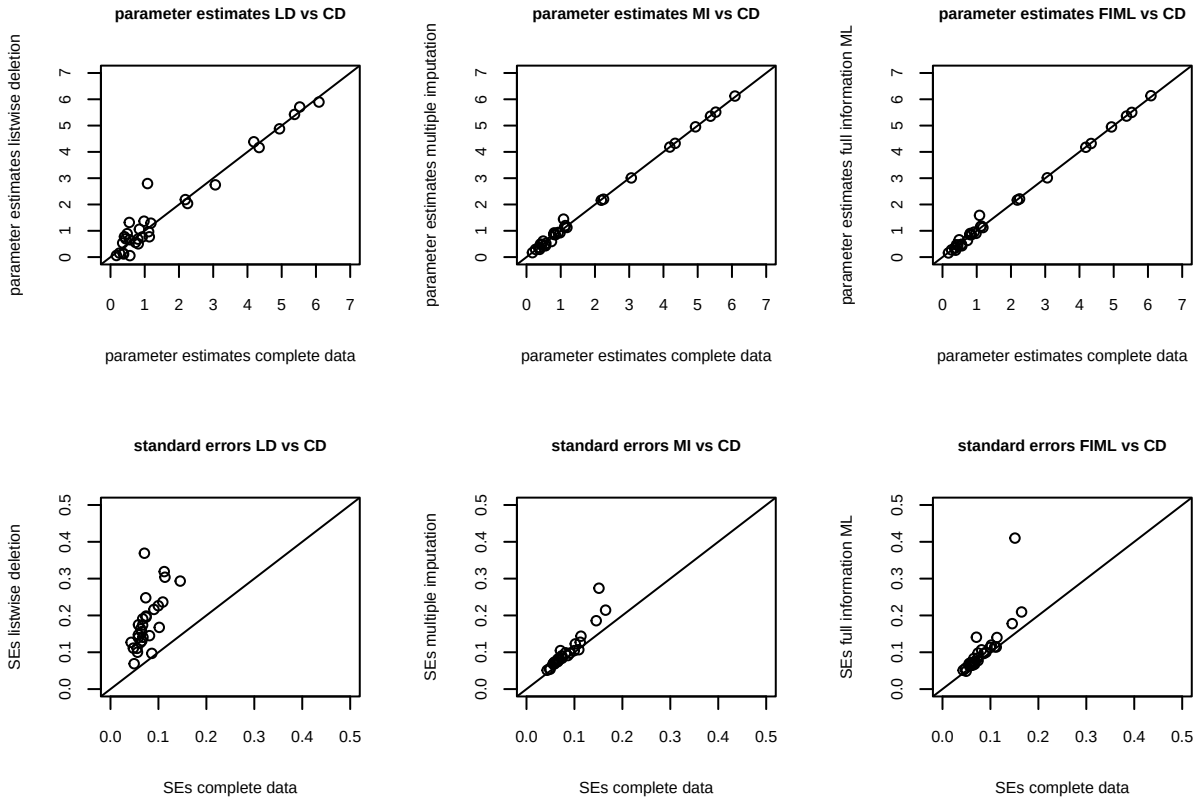
```
##   method     lhs op rhs       est         se
## 2     CD  visual =~  x2 0.5535003 0.09966512
## 3     CD  visual =~  x3 0.7293702 0.10910970
## 5     CD textual =~  x5 1.1130766 0.06542011
## 6     CD textual =~  x6 0.9261462 0.05544886
## 8     CD   speed =~  x8 1.1799508 0.16498657
## 9     CD   speed =~  x9 1.0815302 0.15116744
```

Those are a lot of numbers to compare, let's plot some comparisons:

```
par(mfrow = c(2, 3))
## Plot the parameter estimates
plot(comp_data$est[comp_data$method == "CD"],
     comp_data$est[comp_data$method == "LD"],
     xlim = c(0, 7), ylim = c(0, 7),
     cex.axis = .7, cex.lab= .7, cex.main = .7,
     main = "parameter estimates LD vs CD",
     xlab = "parameter estimates complete data",
     ylab = "parameter estimates listwise deletion")
abline(0, 1)
plot(comp_data$est[comp_data$method == "CD"],
     comp_data$est[comp_data$method == "MI"],
     xlim = c(0, 7), ylim = c(0, 7),
     cex.axis = .7, cex.lab= .7, cex.main = .7,
     main = "parameter estimates MI vs CD",
     ylab = "parameter estimates multiple imputation",
     xlab = "parameter estimates complete data")
abline(0, 1)
plot(comp_data$est[comp_data$method == "CD"],
     comp_data$est[comp_data$method == "FIML"],
     xlim = c(0, 7), ylim = c(0, 7),
     cex.axis = .7, cex.lab= .7, cex.main = .7,
     main = "parameter estimates FIML vs CD",
     ylab = "parameter estimates full information ML",
     xlab = "parameter estimates complete data")
abline(0, 1)

## plot the standard errors
plot(comp_data$se[comp_data$method == "CD"],
     comp_data$se[comp_data$method == "LD"],
     xlim = c(0, 0.5), ylim = c(0, 0.5),
     cex.axis = .7, cex.lab= .7, cex.main = .7,
     main = "standard errors LD vs CD",
     xlab = "SEs complete data",
     ylab = "SEs listwise deletion")
abline(0, 1)
plot(comp_data$se[comp_data$method == "CD"],
     comp_data$se[comp_data$method == "MI"],
     xlim = c(0, 0.5), ylim = c(0, 0.5),
     cex.axis = .7, cex.lab= .7, cex.main = .7,
```

```
     main = "standard errors MI vs CD",
     ylab = "SEs multiple imputation",
     xlab = "SEs complete data")
abline(0, 1)
plot(comp_data$se[comp_data$method == "CD"],
     comp_data$se[comp_data$method == "FIML"],
     xlim = c(0, 0.5), ylim = c(0, 0.5),
     cex.axis = .7, cex.lab= .7, cex.main = .7,
     main = "standard errors FIML vs CD",
     ylab = "SEs full information ML",
     xlab = "SEs complete data")
abline(0, 1)
```



(Note, two very large outlying standard errors for listwise deletion have been omitted from the plots, so we can focus on the patterns shown by the less extreme cases, so the standard errors for listwise deletion are even worse than what we see here.)

Top panels: Parameter estimates for listwise deletion deviate considerably from those obtained with the complete data. This is largely corrected by using either multiple imputation or full information ML.

Bottom panels: Listwise deletion yields much larger standard errors than we would obtain with the complete data. The standard errors obtained with multiple imputation and full information ML are (much) closer to those obtained with the complete data. They are still a bit larger, which they should be, because there is of course missing data, which should result in lower prediction of the estimates.

Parameters relating to exogenous variables

In many SEM analyses, parameters relating to exogenous variables will often not be provided. Often, exogenous variables will be considered fixed. As a result, their (co)variances are fixed to their sample (co)variances, instead of being estimated as parameters in the model. For the model fit (χ^2 and df), this does not make a difference. But sometimes you may want to inspect the variation or associations between the exogonous variables.

```
HS_data <- HolzingerSwineford1939
HS_data$age <- with(HS_data, ageyr + agemo/12)
HS_data$sex <- HS_data$sex - 1 # to make it 0-1 coded
HS.model2 <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  visual + textual ~ sex + age
'
HS_mod1 <- cfa(HS.model2, data = HS_data, estimator = "MLR")
summary(HS_mod1, standardized = TRUE)
```

```
## lavaan 0.6-18 ended normally after 30 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        17
##
##   Number of observations                           301
##
## Model Test User Model:
##                                               Standard      Scaled
##   Test Statistic                                35.619      35.485
##   Degrees of freedom                                16          16
##   P-value (Chi-square)                           0.003       0.003
##   Scaling correction factor                                  1.004
##     Yuan-Bentler correction (Mplus variant)
##
## Parameter Estimates:
##
##   Standard errors                             Sandwich
##   Information bread                           Observed
##   Observed information based on                Hessian
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   visual =~
##     x1                1.000                               0.850    0.729
##     x2                0.635    0.163    3.890    0.000    0.540    0.459
##     x3                0.804    0.174    4.610    0.000    0.683    0.605
##   textual =~
##     x4                1.000                               0.993    0.855
##     x5                1.110    0.067   16.632    0.000    1.102    0.856
##     x6                0.919    0.061   14.952    0.000    0.912    0.834
##
## Regressions:
```

```
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   visual ~
##     sex              -0.329    0.123   -2.676    0.007   -0.387   -0.194
##     age              -0.038    0.064   -0.593    0.553   -0.045   -0.045
##   textual ~
##     sex               0.076    0.122    0.624    0.533    0.077    0.038
##     age              -0.236    0.057   -4.129    0.000   -0.237   -0.241
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##  .visual ~~
##    .textual           0.384    0.105    3.652    0.000    0.479    0.479
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .x1                0.636    0.171    3.714    0.000    0.636    0.468
##    .x2                1.091    0.110    9.957    0.000    1.091    0.789
##    .x3                0.808    0.111    7.294    0.000    0.808    0.634
##    .x4                0.364    0.050    7.257    0.000    0.364    0.270
##    .x5                0.445    0.058    7.606    0.000    0.445    0.268
##    .x6                0.364    0.048    7.559    0.000    0.364    0.304
##    .visual            0.695    0.192    3.613    0.000    0.963    0.963
##    .textual           0.925    0.112    8.235    0.000    0.937    0.937
```

We see that the (co)variances of the exogenous variables (`sex` and `age`) are not estimated in the model. As a results, we cannot inspect their association. To include them in the model as model parameters, we have to additionally specify `fixed.x = FALSE` in the call to `cfa()`:

```
HS_mod2 <- cfa(HS.model2, data = HS_data, estimator = "MLR", fixed.x = FALSE)
summary(HS_mod2, standardized = TRUE)
```

```
## lavaan 0.6-18 ended normally after 32 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        20
##
##   Number of observations                           301
##
## Model Test User Model:
##                                              Standard      Scaled
##   Test Statistic                               35.619      35.485
##   Degrees of freedom                               16          16
##   P-value (Chi-square)                          0.003       0.003
##   Scaling correction factor                                 1.004
##     Yuan-Bentler correction (Mplus variant)
##
## Parameter Estimates:
##
##   Standard errors                             Sandwich
##   Information bread                           Observed
##   Observed information based on                Hessian
##
```

```
## Latent Variables:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   visual =~
##     x1               1.000                               0.850    0.729
##     x2               0.635    0.163    3.890    0.000    0.540    0.459
##     x3               0.804    0.174    4.610    0.000    0.683    0.605
##   textual =~
##     x4               1.000                               0.993    0.855
##     x5               1.110    0.067   16.632    0.000    1.102    0.856
##     x6               0.919    0.061   14.952    0.000    0.912    0.834
##
## Regressions:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   visual ~
##     sex             -0.329    0.123   -2.676    0.007   -0.387   -0.194
##     age             -0.038    0.064   -0.593    0.553   -0.045   -0.045
##   textual ~
##     sex              0.076    0.122    0.624    0.533    0.077    0.038
##     age             -0.236    0.057   -4.129    0.000   -0.237   -0.241
##
## Covariances:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##  .visual ~~
##    .textual          0.384    0.105    3.652    0.000    0.479    0.479
##   sex ~~
##     age             -0.081    0.029   -2.791    0.005   -0.081   -0.160
##
## Variances:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .x1               0.636    0.171    3.714    0.000    0.636    0.468
##    .x2               1.091    0.110    9.957    0.000    1.091    0.789
##    .x3               0.808    0.111    7.294    0.000    0.808    0.634
##    .x4               0.364    0.050    7.257    0.000    0.364    0.270
##    .x5               0.445    0.058    7.606    0.000    0.445    0.268
##    .x6               0.364    0.048    7.559    0.000    0.364    0.304
##    .visual           0.695    0.192    3.613    0.000    0.963    0.963
##    .textual          0.925    0.112    8.235    0.000    0.937    0.937
##     sex              0.250    0.001  289.990    0.000    0.250    1.000
##     age              1.035    0.087   11.907    0.000    1.035    1.000
```