Computer lab:
Experimenting with lavaan

1 CFA example: Holzinger & Swineford

1.1 Dataset and model

In this section, we use the built-in dataset called `HolzingerSwineford1939`. See the help page for this dataset by typing

```
> library(lavaan)
> ?HolzingerSwineford1939
```

at the `R` prompt. This is a 'classic' dataset that is used in many papers and books on Structural Equation Modeling (SEM), including some manuals of commercial SEM software packages. The data consists of mental ability test scores of seventh- and eighth-grade children from two different schools (Pasteur and Grant-White). In our version of the dataset, only 9 out of the original 26 tests are included. A CFA model that is often proposed for these 9 variables consists of three latent variables (or factors), each with three indicators:

- a *visual* factor measured by 3 variables: `x1`, `x2` and `x3`
- a *textual* factor measured by 3 variables: `x4`, `x5` and `x6`
- a *speed* factor measured by 3 variables: `x7`, `x8` and `x9`

1.2 Exercises

1. enter the model syntax and call it 'HS.model'; fit the model:

```
> HS.model <- ' visual  =~ x1 + x2 + x3
+               textual =~ x4 + x5 + x6
+               speed   =~ x7 + x8 + x9
+             '
> fit <- cfa(HS.model, data=HolzingerSwineford1939)
```

2. inspect the summary output; do you understand the meaning of every number printed in the output?

```
> summary(fit, fit.measures = TRUE)
```

3. how many free parameters do we have in this model? How many sample statistics? Can you explain why we have 24 degrees of freedom?

4. the 'baseline model' corresponds (in this context) to an independence model: how does this model look like? How many free parameters? Can you explain why we have 36 degrees of freedom?

5. use the 'cov()' function in R to compute the observed covariance matrix of the data (but only for the variables included in the model)

```
> Data9 <- HolzingerSwineford1939[,c("x1","x2","x3","x4","x5","x6","x7","x8","x9")]
> S <- cov(Data9)
> round(S, 3)
```

6. below, we extract the observed covariance matrix as used by lavaan; can you spot the difference (and do you have an explanation?)

```
> inspect(fit, "sampstat") # alias for lavInspect()
```

7. to get the model-implied covariance matrix, you can use:

```
> fitted(fit)
```

8. (optional) try to manually compute this implied-covariance matrix using the matrix formula for the CFA model; first we need to provide the matrices, so you can work with them; you can refer to them by their names (lambda, theta, psi); to multiply two matrices A and B, you can use `A %*% B`; to add two matrices, you can use `A + B`; the transpose of the matrix A is given by `t(A)`

```
> attach(inspect(fit, "est"))
> # Sigma.hat <- ??
```

9. fit a more restricted model with orthogonal factors, and compare the results; which type of test is used here?

```
> fit.ortho <- cfa(HS.model, data=HolzingerSwineford1939, orthogonal=TRUE)
> anova(fit, fit.ortho)
```

10. try the following commands:

```
> parameterEstimates(fit)
> coef(fit)
> resid(fit)
> resid(fit, type="standardized")
```

11. how is the model represented?

```
> inspect(fit)              # free paraeters
> inspect(fit, "start")     # starting values
> parTable(fit)             # or parameterTable(fit)
```

12. (optional) use the lavaan() function (instead of the cfa() function) to fit the model; adjust the model syntax to include the missing pieces, or use the auto.* arguments of the lavaan() function

2 SEM example: political democracy

2.1 Dataset and model

The `PoliticalDemocracy` data is a dataset that has been used by Bollen in his 1989 book on structural equation modeling (and elsewhere). To learn more about the dataset, see the help page and the references therein. See the slides for a path diagram of the model.

2.2 Excercises

1. fit the model and explore the results

```
> model <- '
+ # latent variable definitions
+   ind60 =~ x1 + x2 + x3
+   dem60 =~ y1 + y2 + y3 + y4
+   dem65 =~ y5 + y6 + y7 + y8
+
+ # regressions
+   dem60 ~ ind60
+   dem65 ~ ind60 + dem60
+
+ # residual covariances
+   y1 ~~ y5
+   y2 ~~ y4 + y6
+   y3 ~~ y7
+   y4 ~~ y8
+   y6 ~~ y8
+ '
> fit <- sem(model, data=PoliticalDemocracy)
> summary(fit)
> summary(fit, standardized=TRUE, rsquare=TRUE, fit.measures=TRUE)
```

2. inspect the parameter table

```
> parameterTable(fit)
> # alias: parTable(fit)
```

3. run the `lavaanify()` function directly to construct a parameter table; which parameters are missing? (hint: use the `auto.*` arguments to add them)

```
> lavaanify(model)
```

4. fit a similar model where the (free) factor loadings of the `dem60` factor are constrained to be equal to the factor loadings of the `dem65` factor; why is this a reasonable idea? compare with the unconstrained model using the `anova()` function

```
> # first specify the model in a model object `model.equal'
> # next, fit the model and call the fitted model `fit.equal'
> # lastly, compare the two models as follows
> anova(fit.equal, fit)
```

5. remove some of the residual covariances, fit the revised model, and inspect the modification indices

```
> # after the modified model has been specified and fitted:
> modindices(fit, sort = TRUE, max = 5)
```

6. look at the manual page of the `inspect` function, and try out the different options

```
> ?inspect
```

3 Meanstructures and multiple groups

3.1 Dataset and model

We use again the built-in dataset `HolzingerSwineford1939`, and the default HS.model:

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

3.2 Exercises

1. enter the model syntax and call it 'HS.model'; fit the model:

```
> library(lavaan)
> HS.model <- ' visual  =~ x1 + x2 + x3
+               textual =~ x4 + x5 + x6
+               speed   =~ x7 + x8 + x9
+             '
> fit <- cfa(HS.model, data=HolzingerSwineford1939)
```

2. use the 'meanstructure=TRUE' option and study the output; what has changed (and what is still the same)?

```
> fit <- cfa(HS.model, data=HolzingerSwineford1939, meanstructure=TRUE)
```

3. fit a multiple group CFA, using `school` as the grouping factor; inspect the results

```
> fit <- cfa(HS.model, data=HolzingerSwineford1939, group="school")
> summary(fit)
```

4. (optional) create a dataset for each school; fit the model for each school separately; compare the results and the chi-square test statistics with the ones obtained from the multiple group analysis

```
> Group1 <- subset(HolzingerSwineford1939, school=="Pasteur")
> Group2 <- subset(HolzingerSwineford1939, school=="Grant-White")
> fit1 <- cfa(HS.model, data=Group1)
> fit2 <- cfa(HS.model, data=Group2)
```

5. fit the multiple group model again, but constrain the intercepts and the factor loadings to be equal across groups: what happens to the means of the latent variables in the two groups?

```
> fit <- cfa(HS.model, data=HolzingerSwineford1939, group="school",
+            group.equal=c("loadings", "intercepts"))
> summary(fit)
```

4 Missing data

4.1 Dataset and model

We use again the built-in dataset `HolzingerSwineford1939`, and the default HS.model:

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

However, we will remove (randomly) some values, to create some missing data

```
> set.seed(1234)
> HS.missing <- as.data.frame(lapply(HolzingerSwineford1939, function(x) {
+                     idx <- sample(1:length(x), 5); x[idx] <- NA; x}))
```

4.2 Exercises

1. fit the model as usual; what happened to the observations with missing values?

   ```
   > fit <- cfa(HS.model, data=HS.missing)
   ```

2. fit the model with full-information maximum likelihood; if you switch on the verbose=TRUE option, you can see how the EM algorithm is used to estimate the (unrestricted) mean vector and covariance matrix of the data, in the presence of missing data

   ```
   > fit <- cfa(HS.model, data=HS.missing, missing="ml", verbose=TRUE)
   > summary(fit)
   ```

3. look at the missing data patterns; how many patterns do we have?

   ```
   > inspect(fit, "patterns")
   ```

4. look at the pairwise coverage:

   ```
   > inspect(fit, "coverage")
   ```

5 Nonnormal data and alternative estimators

5.1 Dataset and model

In this section, we use the built-in dataset `PoliticalDemocracy` and the default model (without equality constraints):

```
> model <- '
+ # latent variable definitions
+   ind60 =~ x1 + x2 + x3
+   dem60 =~ y1 + y2 + y3 + y4
+   dem65 =~ y5 + y6 + y7 + y8
+
+ # regressions
+   dem60 ~ ind60
+   dem65 ~ ind60 + dem60
+
+ # residual covariances
+   y1 ~~ y5
+   y2 ~~ y4 + y6
+   y3 ~~ y7
+   y4 ~~ y8
+   y6 ~~ y8
+ '
```

5.2 Exercises

1. fit the model using robust standard errors; on average, are they larger or smaller?

   ```
   > fit <- sem(model, data=PoliticalDemocracy, se="robust")
   > summary(fit)
   ```

2. fit the model using bootstrap standard errors; how do they compare to the 'robust' standard errors?

   ```
   > fit <- sem(model, data=PoliticalDemocracy, se="bootstrap", bootstrap=200,
   +              verbose=TRUE)
   > summary(fit)
   ```

3. fit the model using a scaled 'Satorra-Bentler' test statistic; request the additional fit measures in the summary output; which fit measures are affected by the scaling of the test statistic?

   ```
   > fit <- sem(model, data=PoliticalDemocracy, test="Satorra-Bentler")
   > summary(fit, fit.measures=TRUE)
   ```

4. fit the model using a bootstrap test statistic

   ```
   > fit <- sem(model, data=PoliticalDemocracy, test="bootstrap", bootstrap=200,
   +              verbose=TRUE)
   > fit
   ```

5. use the 'Yuan-Bentler' scale test statistic, together with robust standard errors

   ```
   > fit <- sem(model, data=PoliticalDemocracy, estimator="MLR")
   > summary(fit)
   ```

6. (optional) try out other estimators: GLS, WLS and MLF

6 IRT using lavaan (optional)

6.1 Dataset and model

We use again the built-in dataset `HolzingerSwineford1939`. However, we will transform the 9 continuous variables into binary variables, and fit a single-factor model. To create the binary variables, we proceed as follows:

```
> HS9 <- HolzingerSwineford1939[,c("x1","x2","x3","x4","x5","x6","x7","x8","x9")]
> HSbinary <- as.data.frame( lapply(HS9, cut, 2, labels=FALSE) )
```

All analyses in this section will use the newly created data.frame 'HSbinary'.

6.2 Exercises

1. fit a single-factor model, treating all variables as binary; fix the variance of the factor to unity. Explore the results.

   ```
   > model <- ' trait =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 '
   > fit <- cfa(model, data=HSbinary, ordered=names(HSbinary), std.lv=TRUE)
   > summary(fit)
   ```

2. inspect the model matrices; which matrices are specific for the categorical variables? Request the model-implied statistics.

   ```
   > inspect(fit)
   > fitted(fit)
   ```

3. (optional) manually compute the sample-based threshold for the first item (x1)

   ```
   > x <- HSbinary$x1
   > N <- length(x)
   > prop <- table(x)/N
   > ncat <- length(prop)
   > prop.cumulative <- cumsum(prop)
   > thresholds <- qnorm(prop.cumulative)[-ncat]
   ```

4. (optional) can you manually compute the model-implied correlations?

5. (optional) how did we get values for the (residual) variances of the items in the summary(output); do you know how to compute these values?

6. (optional) can you compute item discrimination and item difficulty values (using the probit metric) based on these parameters?

   ```
   > pe <- parameterEstimates(fit)
   > lambda <- pe$est[ pe$op == "=~" ]
   > tau <- pe$est[ pe$op == "|" ]
   > rvar <- pe$est[ pe$op == "~~" & pe$lhs != "trait"]
   > item.discrimination <- lambda/sqrt(rvar)
   > item.difficulty <- tau/lambda
   ```

7. (optional) refit the model, but constrain all the factor loadings to be equal; compute again the item discrimination and item difficulty values

7 Binary CFA example

In this section, we (again) use the binary version of the Holzinger & Swineford dataset

```
> HS9 <- HolzingerSwineford1939[,c("x1","x2","x3","x4","x5",
+                                  "x6","x7","x8","x9")]
> HSbinary <- as.data.frame( lapply(HS9, cut, 2, labels=FALSE) )
> HSbinary$school <- HolzingerSwineford1939$school
> head(HSbinary)
```

1. check which variables in the data frame 'HSbinary' are declared as numeric, factor, ordered

   ```
   > varTable(HSbinary)
   ```

2. fit the usual 3-factor model, using the default estimator (WLSMV):

   ```
   > model <- ' visual  =~ x1 + x2 + x3
   +            textual =~ x4 + x5 + x6
   +            speed   =~ x7 + x8 + x9 '
   > # binary CFA
   > fit <- cfa(model, data=HSbinary,
   +            ordered=c("x1","x2","x3","x4","x5","x6","x7", "x8","x9"))
   > summary(fit, fit.measures=TRUE)
   ```

3. look at the sample statistics, the parameter table and the parameter matrices; do you understand what you see?

   ```
   > fitted(fit)
   > parTable(fit)
   > inspect(fit)
   ```

8 Repeated measures analysis using lavaan

In this section, we use the 'positive affect' data, from Todd Little's book (Longitudinal SEM, 2013): table 3.8

```
> MEAN <- c(3.06893, 2.92590, 3.11013, 3.02577, 2.85656, 3.09346)
> SDS  <- c(0.84194, 0.88934, 0.83470, 0.84081, 0.90864, 0.83984)
> lower <- '
+     1.00000
+     0.55226    1.00000
+     0.56256    0.60307    1.00000
+     0.31889    0.35898    0.27757    1.00000
+     0.24363    0.35798    0.31889    0.56014    1.00000
+     0.32217    0.36385    0.32072    0.56164    0.59738    1.00000 '
> COV <- getCov(lower, sds=SDS, names = c("Glad1", "Cheer1", "Happy1",
+                                         "Glad2", "Cheer2", "Happy2"))
```

In the model, we assume a single latent variable ('positive affect'), measured two times in time. The latent variable is measured by three indicators ('Glad', 'Cheer' and 'Happy'), for which we have measures for each time point.

1. fit the 'configural' longitudinal CFA model

   ```
   > model1 <- '
   +     posAffect1 =~ 1*Glad1 + Cheer1 + Happy1
   +     posAffect2 =~ 1*Glad2 + Cheer2 + Happy2
   +     posAffect1 ~~ posAffect2
   +
   +     # intercepts
   +     Glad1  ~ 1
   +     Glad2  ~ 1
   +     Cheer1 ~ 1
   +     Cheer2 ~ 1
   +     Happy1 ~ 1
   +     Happy2 ~ 1
   +
   +     # residual covariances
   +     Glad1  ~~ Glad2
   +     Cheer1 ~~ Cheer2
   ```

```
+      Happy1 ~~ Happy2
+
+      # latent means: fixed to zero
+      posAffect1 ~ 0
+      posAffect2 ~ 0
+ '
> fit1 <- lavaan(model1, sample.cov = COV, sample.mean = MEAN,
+                  sample.nobs = 823, auto.var = TRUE)
> summary(fit1, standardized = TRUE)
```

2. can we establish measurement invariance across time points for this model?
3. if we have measurement invariance: is there a difference between the latent means of 'positive afffect' at time point 1 and time point 2?

9 Panel models + Growth curves

Here we use data from the Curran & Bollen (2001) book chapter 'The Best of Two Worlds' about the developmental relation between antisocial behavior and depressive symptomatology. The original data come from the National Longitudinal Survey of Youth (NLSY); the original 1979 panel included a total of 12,686 respondents. In 1986, the children of the original NLSY female respondents were also included in the survey. The sample used for this example only includes data of children that were 8 years old at the first wave of measurement, have no missings on all four waves, only biological children, resulting in a final sample of $N = 180$. We only include the sumscores of two constructs (antisocial behavior and depressive symptomatology), measured at 4 time points:

```
> lower <- '
+  2.926
+  1.390 4.257
+  1.698 2.781 4.536
+  1.628 2.437 2.979 5.605
+  1.240 0.789 0.903 1.278 3.208
+  0.592 1.890 1.419 1.004 1.706 3.994
+  0.929 1.278 1.900 1.000 1.567 1.654 3.583
+  0.659 0.949 1.731 2.420 0.988 1.170 1.146 3.649 '
> COV <- getCov(lower, names=c("anti1", "anti2", "anti3", "anti4",
+                               "dep1", "dep2", "dep3", "dep4"))
> MEANS <- c(1.750, 1.928, 1.978, 2.322, 2.178, 2.489, 2.294, 2.222)
```

1. what happens over time with antisocial behaviour? fit a panel model and a growth curve model
2. what happens over time with depressive symptomatology? fit a panel model and a growth curve model
3. how do the two variables relate to each other (over time)? try to answer this with either a panel model or a growth curve model

10 Use your own data

If you have your own data,

- sketch a path diagram of the model you wish to fit (start with a simple version)
- write the lavaan model syntax for this path diagram, and fit the model
- try to interpret the results, and show them to me