Solutions to Lab 7

Multivariate Statistics with R

This week, we will familiarise ourselves further with `OpenMX` and its helper package `umx` and run and compare some models. Onwards!

Task 1: Install the `umx` package.

Task 2: Open help on `umx`: Have a quick scan of the functions, look at an example or two.

Task 3: Run the function `umx_set_optimizer()`.

Question 3.1: What is an optimiser?

```
### ANSWER ###
```

An optimiser is and algorithm used to efficiently find a minimum or maximum of a mathematical function. Here, the optimiser is used to derive the log-likelihood of models.

> **Note** This is a *helper* function: all functions beginning `umx_` are helpers.

Question 3.2: How do you see the code inside a function?

```
### ANSWER ###
```

Simply run the command without the `()`.

Question 3.3: Look at the code inside `umx_set_optimizer`.

```
umx_set_optimizer
```

```
function (opt = NA, model = NULL, silent = FALSE)
{
  if (is.na(opt)) {
    if (is.null(model)) {
      o = mxOption(NULL, "Default optimizer")
    }
    else {
      o = mxOption(model, "Default optimizer")
    }
    if (!silent) {
      quoteOptions = omxQuotes(mxAvailableOptimizers())
      message("Current Optimizer is: ", omxQuotes(o),
        ". Options are: ", quoteOptions)
    }
    invisible(o)
  }
  else {
    if (!opt %in% mxAvailableOptimizers()) {
      stop("The Optimizer ", omxQuotes(opt), " is not legal. Legal values (from mxAvailableOptimizers()
        omxQuotes(mxAvailableOptimizers()))
    }
```

```
    if (is.null(model)) {
      mxOption(NULL, "Default optimizer", opt)
    }
    else {
      stop(paste0("'Default optimizer' is a global option and cannot be set on models. just say:\n",
        "umx_set_optimizer(", omxQuotes(opt), ")"))
    }
  }
}
```

Task 4: Get help on `umxRAM()`

Task 5: Look the first simple example in `umxRAM()` help.

Question 5.1: What do `wt`, `disp`, and `mpg` stand for in `mtcars`?

```
### ANSWER ###
```

From `?mtcars`:

`[, 1] mpg`	Miles/(US) gallon
...	
`[, 3] disp`	Displacement (cu.in.)
...	
`[, 6] wt`	Weight (1000 lbs)

Question 5.2: Create a `mxData` object from the relevant `mtcars` variables (`umxRAM()` can actually cope with dataframes as input but do this anyway).

```
### ANSWER ###

myData <- mxData(cov(mtcars[ , c(1, 3, 6)]), "cov", numObs = nrow(mtcars))
```

Question 5.3: Are we using the raw data or a covariance matrix?

```
### ANSWER ###
```

Covariance matrix.

Question 5.4: Build model `m1`.

```
### ANSWER ###

m1 <- umxRAM("tim", data = myData,
             umxPath(c("wt", "disp"), to = "mpg"),
             umxPath("wt", with = "disp"),
             umxPath(var = c("wt", "disp", "mpg")))
```

```
## [1] "<U+03C7>²(0) = -0.05, p = 1.000; CFI = 1.001; TLI = 1; RMSEA = 0"
```

Task 6: By default, once you are done, you see a compact fit summary.

Question 6.1: Is this good fit?

```
### ANSWER ###
```

Yes! In fact, it is a perfect fit ($\chi^2(0) \approx 0$) as the model is *saturated*, AKA "just identified". Such a model has zero degrees of freedom because the **number of estimated parameters**, in this case six, **is equal to the number known values** (three variances and three covariances).

Task 7: Get a `summary()` of the model.

```
### ANSWER ###

summary(m1)
```

```
## Summary of tim
##
## free parameters:
##             name matrix  row  col      Estimate    Std.Error A lbound
## 1    disp_to_mpg      A  mpg disp -1.772516e-02 8.748828e-03
## 2      wt_to_mpg      A  mpg   wt -3.350778e+00 1.108187e+00
## 3   mpg_with_mpg      S  mpg  mpg  7.708730e+00 1.927117e+00 !      0
## 4 disp_with_disp      S disp disp  1.488075e+04 3.720157e+03        0
## 5   disp_with_wt      S disp   wt  1.043185e+02 2.777299e+01
## 6     wt_with_wt      S   wt   wt  9.274526e-01 2.318590e-01        0
##   ubound
## 1
## 2
## 3
## 4
## 5
## 6
##
## Model Statistics:
##                |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
##        Model:              6                     0              416.6821
##    Saturated:              6                     0              416.7300
## Independence:              3                     3              515.0320
## Number of observations/statistics: 32/6
##
## chi-square:  <U+03C7>² ( df=0 ) = -0.04787503,  p = 1
## Information Criteria:
##       |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
## AIC:    -0.04787503               11.95212                       NA
## BIC:    -0.04787503               20.74654                 2.041965
## CFI: 1.000502
## TLI: 1   (also known as NNFI)
## RMSEA:  0  [95% CI (NA, NA)]
## Prob(RMSEA <= 0.05): NA
## timestamp: 2017-11-01 13:38:28
## Wall clock time: 0.423198 secs
## optimizer:  CSOLNP
## OpenMx version number: 2.7.18
## Need help?  See help(mxSummary)
```

Question 7.1: What do the components mean?

```
### ANSWER ###
```

Most notably:

`free parameters` - paths we are estimating

- names of the parameters
- the estimated values
- the *SE*s of the estimates

`Model Statistics` - summary of model fit

- the rows are the model that are compared in order to produce comparative model fit indices:
  - `Model` is the model we specified using `umxRAM()`
  - `Saturated` is the, well, saturated model (see above). As you can see, in our case `Model` and `Saturated` have the same values becuase, as discussed, the model we fit is saturated.
  - `Independence` is the **worst possible** model, *i.e.,* one where we assume nothing is related to anything. In this case, it is a model, where we only estimate the variance of each variable, but no correlations or regression coefficients (hence 3 parameters).
- the columns are:
  - the number of parameters in each of the 3 models
  - Model degrees of freedom (see above)
  - the model fit in terms of $-2LL$ (log-likelihood)

`Information Criteria` - Better measures of fit

- `AIC` is the 'Akaike Information Criterion' which penalises for degrees of freedom. That means that a simpler model (with relatively more *df*s) will have a smaller value of AIC than en equally well-fitting more complex model.
- `CFI`, `TLI`, `RMSEA` are some other fit indices; we will go over these next week.

Task 8: Compare with `umxSummary()`.

Question 8.1: Get `umxSummary()` to show the path estimates.

```
### ANSWER ###

umxSummary(m1, showEstimates = "raw")
```

```
##
##
## |name           | Estimate|      SE|
## |:--------------|--------:|-------:|
## |disp_to_mpg    |    -0.02|    0.01|
## |wt_to_mpg      |    -3.35|    1.11|
## |mpg_with_mpg   |     7.71|    1.93|
## |disp_with_disp | 14880.75| 3720.16|
## |disp_with_wt   |   104.32|   27.77|
## |wt_with_wt     |     0.93|    0.23|
## [1] "<U+03C7>²(0) = -0.05, p = 1.000; CFI = 1.001; TLI = 1; RMSEA = 0"
```

Task 9: Get the AIC from the model.

```
### ANSWER ###

AIC(m1)
```

```
## [1] 428.6821
```

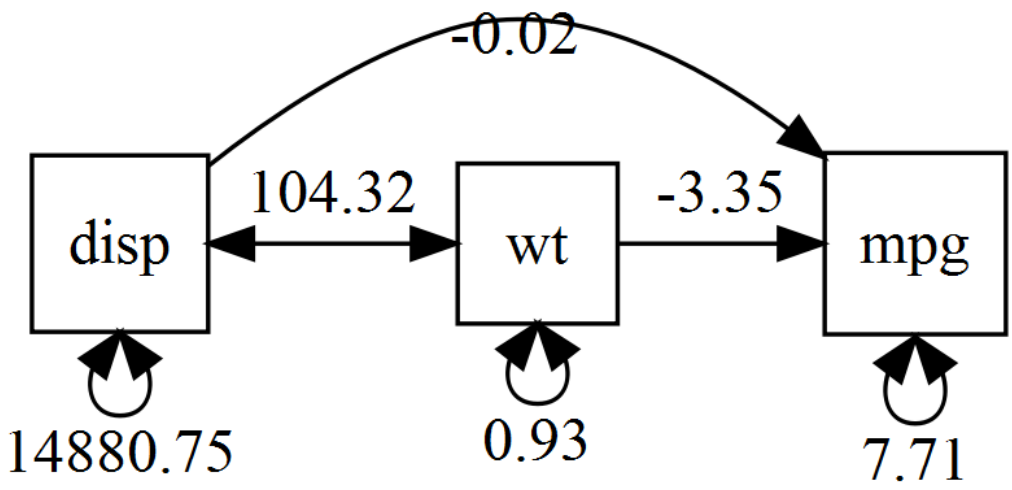
Question 9.1: What does AIC stand for?

```
### ANSWER ###
```

"Akaike Information Criterion". For more info see David Kenny's excellent (well, content-wise) page on fit indices.

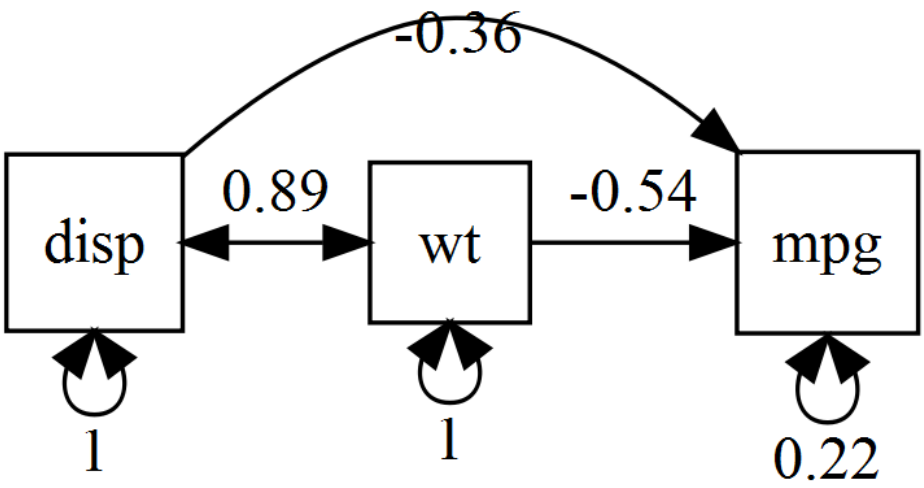
Task 10: Plot the output.

```
### ANSWER ###

plot(m1)
```



Task 11: Get `plot()` to draw a standardised model.

```
### ANSWER ###

plot(m1, std = T)
```



Task 12: Why aren't there any means in the plot?

```
### ANSWER ###
```

Because we only provided a variance-covariance matrix to `umxRAM()` without specifying a vector of means. Since var-cov matrix reduces the information in the data, there is no way to reconstruct the means of the variables from the matrix.

Task 13: Draw a model of the hypothesis in your thesis.

Question 13.1: Do all the paths in your drawing have arrow heads on at least one end?

Question 13.2: Why must we always draw the arrow heads on our paths (not just leave them blank)?

```
### ANSWER ###
```

Because we need to tell `R` the direction of the relationship between the variables (*e.g.,* A predicts B, B predicts A, A and B are correlated, A loads on B).

Question 13.3: Are all the circles on your model connected to squares?

```
### ANSWER ###
```

Please say yes!

Question 13.4: Is everything you are interested in measured multiple ways, so it can be a latent variable?

```
### ANSWER ###
```

Remember that for a measurement model to be identified, you need at least two observed variables per latent variable (assuming several things discussed in a later lecture).

Question 13.5: Did you draw the expected residuals and covariance?

```
### ANSWER ###
```

That is very nice!

Task 14: Build a new model `m2`.

Question 14.1: Make it like the one in Question 4.4, but leave out the path from `wt` to `mpg`.

```
### ANSWER ###

m2 = umxRAM("tim2", data = myData,
            umxPath("disp", to = "mpg"),
            umxPath("wt", with = "disp"),
            umxPath(var = c("wt", "disp", "mpg")))
```

```
## [1] "<U+03C7>²(1) = 7.99, p = 0.005; CFI = 0.927; TLI = 0.78; RMSEA = 0.467"
```

Question 14.2: Did it run?

```
### ANSWER ###
```

It should have!

Task 15: umxCompare `m1` and `m2`

```
### ANSWER ###

umxCompare(m1, m2)
```

```
##   Model EP delta -2LL delta df     p         AIC Compare with Model
## 1   tim  6         NA       NA  <NA> -0.04787503               <NA>
## 2  tim2  5   8.041605        1 0.005  5.99372987                tim
```

Question 15.1: How did `m2` fit the data compared to `m1`? How do you know?

```
### ANSWER ###
```

The fit was worse, since the p-value of the likelihood ratio test is significant and there is a decrease in likelihood in `m2` compared to `m1`. (or an *increase* in $-2LL$ of 8.04). Also, given that `m1` is a saturated model with a perfect fit, any non-saturated model will fit worse than `m1` (though not necessarily statisctically significantly worse!).

Question 15.2: What happened to AIC?

```
### ANSWER ###
```

It has increased by 6.04, reflecting the worse fit. (Remember, lower AIC is better!)

Task 16: Visit the `OpenMx` home page.

Task 17: Build and run the 1-factor CFA model on the home page.

```
### ANSWER ###

data(demoOneFactor)
manifests <- names(demoOneFactor)
latents <- c("G")
factorModel <- mxModel("One Factor",
                       type = "RAM",
                       manifestVars = manifests,
                       latentVars = latents,
                       mxPath(from = latents, to = manifests, values = 0.8),
                       mxPath(from = manifests, arrows = 2, values = 1),
                       mxPath(from = latents, arrows = 2,
                              free = FALSE, values = 1.0),
                       mxData(cov(demoOneFactor), type = "cov", numObs = 500))

summary(factorModelFit <- mxRun(factorModel))
```

```
## Summary of One Factor
##
## free parameters:
##                 name matrix row col   Estimate   Std.Error A
## 1  One Factor.A[1,6]      A  x1   G 0.39675442 0.015518623
## 2  One Factor.A[2,6]      A  x2   G 0.50315690 0.018196001
## 3  One Factor.A[3,6]      A  x3   G 0.57666351 0.020407447
## 4  One Factor.A[4,6]      A  x4   G 0.70207009 0.023963327
## 5  One Factor.A[5,6]      A  x5   G 0.79545281 0.026616042
## 6  One Factor.S[1,1]      S  x1  x1 0.04073255 0.002804281
## 7  One Factor.S[2,2]      S  x2  x2 0.03794394 0.002797378
## 8  One Factor.S[3,3]      S  x3  x3 0.04074551 0.003142855
## 9  One Factor.S[4,4]      S  x4  x4 0.03930827 0.003398652
## 10 One Factor.S[5,5]      S  x5  x5 0.03621453 0.003667530
##
## Model Statistics:
##                |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
##        Model:             10                     5            -3660.5967
##    Saturated:             15                     0            -3667.9905
## Independence:              5                    10               64.5342
## Number of observations/statistics: 500/15
##
## chi-square:  <U+03C7>² ( df=5 ) = 7.393793,  p = 0.1929616
## Information Criteria:
##       |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
## AIC:      -2.606207              27.39379                     NA
## BIC:     -23.679247              69.53987                37.79926
## CFI: 0.9993569
## TLI: 0.9987139   (also known as NNFI)
## RMSEA:  0.03094378  [95% CI (0, 0.08143354)]
## Prob(RMSEA <= 0.05): 0.7135768
## timestamp: 2017-11-01 13:38:36
## Wall clock time: 0.01559997 secs
## optimizer:  CSOLNP
## OpenMx version number: 2.7.18
## Need help?  See help(mxSummary)
```

As mentioned above, every unsaturated model will have a worse fit to the data than the saturated model, however the difference need not be significant. That is exactly the case here: the model's $-2LL$ was higher by 7.39 but this change had an associated p-value of .193.

Question 17.1: Try leaving out a path from g to one of the items.

```
### ANSWER ###

m3 <- mxModel("One Factor",
              type = "RAM",
              manifestVars = manifests,
              latentVars = latents,
              mxPath(from = latents, to = manifests[-1], values = 0.8),
              mxPath(from = manifests, arrows = 2, values = 1),
              mxPath(from = latents, arrows = 2,
                     free = FALSE, values = 1.0),
              mxData(cov(demoOneFactor), type = "cov", numObs = 500))

summary(m3Fit <- mxRun(m3))
```

```
## Summary of One Factor
##
## free parameters:
##                name matrix row col   Estimate   Std.Error A
## 1 One Factor.A[2,6]      A  x2   G 0.50309374 0.018204525
## 2 One Factor.A[3,6]      A  x3   G 0.57591381 0.020437260
## 3 One Factor.A[4,6]      A  x4   G 0.70232747 0.023965927
## 4 One Factor.A[5,6]      A  x5   G 0.79590692 0.026615648
## 5 One Factor.S[1,1]      S  x1  x1 0.19814720 0.012531932
## 6 One Factor.S[2,2]      S  x2  x2 0.03800766 0.002845366
## 7 One Factor.S[3,3]      S  x3  x3 0.04160982 0.003259771
## 8 One Factor.S[4,4]      S  x4  x4 0.03894719 0.003522483
## 9 One Factor.S[5,5]      S  x5  x5 0.03549237 0.003878474
##
## Model Statistics:
##                |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
##        Model:              9                      6            -2910.1801
##    Saturated:             15                      0            -3667.9905
## Independence:              5                     10               64.5342
## Number of observations/statistics: 500/15
##
## chi-square:  <U+03C7>² ( df=6 ) = 757.8104,  p = 2.003975e-160
## Information Criteria:
##       |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
## AIC:       745.8104               775.8104                      NA
## BIC:       720.5227               813.7418                785.1753
## CFI: 0.7980375
## TLI: 0.6633958   (also known as NNFI)
## RMSEA:  0.5006031  [95% CI (0.4650893, 0.5367762)]
## Prob(RMSEA <= 0.05): 0
## timestamp: 2017-11-01 13:38:36
## Wall clock time: 0.01559997 secs
## optimizer:  CSOLNP
## OpenMx version number: 2.7.18
```

```
## Need help?  See help(mxSummary)
```

Question 17.2: Does the model even run?

```
### ANSWER ###
```

Again, it should...

Question 17.3: Does it fit worse or better? How would we know?

```
### ANSWER ###

# you can use either
mxCompare(factorModelFit, m3Fit)
```

```
##         base comparison ep  minus2LL df        AIC   diffLL diffdf
## 1 One Factor       <NA> 10 -3660.597  5  -2.606207       NA     NA
## 2 One Factor One Factor  9 -2910.180  6 745.810370 750.4166      1
##                p
## 1             NA
## 2 3.257038e-165
```

```
# or
umxCompare(factorModelFit, m3Fit)
```

```
##        Model EP delta -2LL delta df       p        AIC Compare with Model
## 1 One Factor 10          NA       NA    <NA>  -2.606207               <NA>
## 2 One Factor  9    750.4166        1 < 0.001 745.810370         One Factor
```

The fit of this new model is singificantly worse that the original one, $\Delta\chi^2(1) = 750.42,\ p < .001$.

Task 18: Visit the `umx` home page

Question 18.1: For homework, try some other example models.

Next week, we will discuss model fit and model comparison in more detail.

See you!