Moderation

There aren't many options for latent variable interaction in R - at least not readily available. The two most common ones, LMS and product indicator approach, can be done in R with existing packages. Besides those, the authors of the book chapter have created multiple functions for other forms of latent variable interactions.

This script covers LMS and product indicator approaches.

LMS approach

https://cran.r-project.org/web/packages/nlsem/nlsem.pdf (https://cran.r-project.org/web/packages/nlsem/nlsem.pdf)

NLSEM can be used to perform latent variable interactions in models with **latent variables and latent interactions only**. The model cannot have observed variables as outcomes, latent variables with fewer than 3 indicators, and entering constraints is less straightforward than in *Mplus* or *lavaan*.

For this reason, we're using a different dataset to show this package.

```
library(nlsem)
```

```
## Warning: package 'nlsem' was built under R version 4.1.3
```

```
## Loading required package: orthopolynom
```

```
## Warning: package 'orthopolynom' was built under R version 4.1.3
```

```
## Loading required package: polynom
```

```
dataset <- read.csv("sample_timms2015.csv", header = TRUE)
```

The package cannot read variable names. You have to have your variables in order from x to y variables and with names x1, x2, x3, … y1, y2, …, and so on. So we're changing the variable names.

```
colnames(dataset) <- c(
                       "x1",
                       "x2",
                       "x3",
                       "x4",
                       "x5",
                       "x6",
                       "y1",
                       "y2",
                       "y3"
                       )
```

There are a couple of ways of specifying models with NLSEM, one of them is to specify a lavaan type of model, and then use the `` `lav2nlsem()` `` function turn it into an object that can be read by the NLSEM package.

You also need to name the latent variables xi1, xi2, …, for the exogenous latent variables (as in LISREL notation), and eta1, eta2, …, for the endogenous latent variables.

```
lav_model <- 'xi1 =~ x1 + x2 + x3
              xi2 =~ x4 + x5 + x6
              eta1 =~ y1 + y2 + y3
              eta1 ~ xi1 + xi2 + xi1:xi2'

model <- lav2nlsem(lav_model)
model
```

```
## Model of class singleClass
##
## Number of latent endogenous variables: 1 (with 3 indicators)
## Number of latent exogenous variables: 2 (with 6 indicators)
##
## Structural model:
## ------------------------------------------------------------
## eta1 = xi1 + xi2
## eta1 = xi1:xi2
## ------------------------------------------------------------
##
## Measurement model:
## ------------------------------------------------------------
## xi1 = x1 + x2 + x3
## xi2 = x4 + x5 + x6
## eta1 = y1 + y2 + y3
## ------------------------------------------------------------
```

According to the NLSEM documentation, we then fit the model along with starting values, which we compute using the `runif()` function. Then we fit the model using the `em()` function.

```
# fit model
set.seed(123)
start <- runif(count_free_parameters(model))
start
```

```
##  [1] 0.28757752 0.78830514 0.40897692 0.88301740 0.94046728 0.04555650
##  [7] 0.52810549 0.89241904 0.55143501 0.45661474 0.95683335 0.45333416
## [13] 0.67757064 0.57263340 0.10292468 0.89982497 0.24608773 0.04205953
## [19] 0.32792072 0.95450365 0.88953932 0.69280341 0.64050681 0.99426978
## [25] 0.65570580 0.70853047 0.54406602 0.59414202 0.28915974 0.14711365
## [31] 0.96302423
```

If you run this code, you will notice that there are lots of errors during model fitting and that the process takes a **very** long time. This model can be fitted properly in Mplus, so here's a plug to go to Mplus if you can.

```
model_fit <- em(model = model,
                data = dataset,
                start = start,
                m = 16,
                max.iter = 500
                )
```

```
## Warning in value[[3L]](cond): Starting parameters for Phi are not positive
## definite. Identity matrix was used instead.
```

```
## Warning in em(model = model, data = dataset, start = start, m = 16, max.iter =
## 500): Loglikelihood should be increasing.
```

```
summary(model_fit)
```

```
## Warning in calc_standard_error(object$neg.hessian): Standard errors for some
## coefficients could not be computed.
```

```
##
## Summary for model of class singleClass
##
## Estimates:
##              Estimate Std. Error z value Pr(>|z|)
## Lambda.x2   1.2408417   0.052727  23.533  < 2e-16 ***
## Lambda.x3   1.3268918   0.051046  25.994  < 2e-16 ***
## Lambda.x11  0.8768280   0.022218  39.464  < 2e-16 ***
## Lambda.x12  0.8786082   0.022584  38.904  < 2e-16 ***
## Lambda.y2   1.0307247   0.024733  41.674  < 2e-16 ***
## Lambda.y3   1.0197537   0.024809  41.103  < 2e-16 ***
## Gamma1      0.5625107   0.113827   4.942 7.74e-07 ***
## Gamma2      0.0315354   0.074615   0.423  0.67256
## Theta.d1    0.1667143   0.011792  14.138  < 2e-16 ***
## Theta.d8    0.1644617   0.011630  14.141  < 2e-16 ***
## Theta.d15   0.1016765   0.007191  14.139  < 2e-16 ***
## Theta.d22   0.0827390   0.010568   7.830 4.90e-15 ***
## Theta.d29   0.0941960   0.009541   9.873  < 2e-16 ***
## Theta.d36   0.0987931   0.009892   9.988  < 2e-16 ***
## Theta.e1    0.1361468   0.011821  11.517  < 2e-16 ***
## Theta.e5    0.0580989   0.008392   6.923 4.42e-12 ***
## Theta.e9    0.0635175   0.008457   7.510 5.90e-14 ***
## Psi         0.7589051   0.066262  11.453  < 2e-16 ***
## Phi1        0.3344837   0.019148  17.468  < 2e-16 ***
## Phi2        0.3836239   0.032333  11.865  < 2e-16 ***
## Phi4        0.8423361   0.025777  32.678  < 2e-16 ***
## nu.x2       0.0017431        NaN     NaN      NaN
## nu.x3       0.0016258   0.004880   0.333  0.73901
## nu.x5      -0.0001687        NaN     NaN      NaN
## nu.x6      -0.0001684        NaN     NaN      NaN
## nu.y2      -0.0206417   0.020872  -0.989  0.32268
## nu.y3      -0.0089141   0.009646  -0.924  0.35541
## alpha      -0.1176024   0.055974  -2.101  0.03564 *
## tau1        0.0181249   0.006627   2.735  0.00624 **
## tau2        0.0194061   0.032983   0.588  0.55629
## Omega3      0.2536833   0.080090   3.167  0.00154 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations: 101
## Final loglikelihood: -2168.723
```

Product indicator approach

As mentioned in lecture, if LMS is not possible, the product indicator approach is the next best option. the *semTools* package can do this in combination with lavaan.

semTools: https://cran.r-project.org/web/packages/semTools/semTools.pdf (https://cran.r-project.org/web/packages/semTools/semTools.pdf)

Load packages

```
library(semTools)
```

```
## Loading required package: lavaan
```

```
## This is lavaan 0.6-9
## lavaan is FREE software! Please report any bugs.
```

```
##
```

```
## ###############################################################################
```

```
## This is semTools 0.5-5
```

```
## All users of R (or SEM) are invited to submit functions or ideas for functions.
```

```
## ###############################################################################
```

```
library(lavaan)
```

Load data and give variable names

```
narr_data <- read.table("PCL_MR.dat")
```

```
colnames(narr_data) <- c(
                      "PANASNEG",
                      "neurot",
                      "AIMN",
                      "ces",
                      "ptgi",
                      "closure",
                      "pcl"
                      )
```

Create a new dataframe with new variables for the product of the indicators of the X and Z variables. To do this, we use the `indProd()` function of *semTools*.

```
prod_data <- indProd(
                  data = narr_data, #create a new data.frame with the interaction ind
icators
                  var1 = 1:3, #interaction indicators from the first latent
                  var2 = 4:6, #interaction indicators from the second latent
                  match = T, #use match-paired approach
                  meanC = T, # mean centering the main effect indicator before making
the products
                  #residualC = T, #residual centering the products by the main effect
indicators
                  #doubleMC = T #centering the resulting products
                  )

head(prod_data[,(ncol(prod_data)-2):ncol(prod_data)]) #check the last three columns o
f the new data.frame
```

```
##   PANASNEG.ces neurot.ptgi AIMN.closure
## 1   -0.6368835   -1.438418    3.1226613
## 2    4.3838250   12.211928    5.9789113
## 3   -4.8706055   -8.966489   -5.7041317
## 4  -10.0010635  -11.455919   -0.2276461
## 5   -4.7256009    2.390108   -2.0786194
## 6    3.3995299   -5.275317   15.0419748
```

Specify a lavaan model

We'll now specify a lavaan model by creating a third latent variable for the interaction between X and Z. The indicators are the product terms created in the previous step.

```
product_model <- '
          NA_var =~ PANASNEG + neurot + AIMN
          Narr =~ ces + ptgi + closure
          NAxNarr =~ PANASNEG.ces + neurot.ptgi + AIMN.closure
          NAxNarr ~~ 0*NA_var + 0*Narr

          # Lines below necessary for model identification
          pcl ~ NA_var + Narr + NAxNarr # No correlation between int and xi1 xi2
          NA_var + Narr + NAxNarr ~ 0*1 # Intercept of latent variables fixed to 0
          '
```

Fit the model

```
product_fit <- sem(product_model,
                   data = prod_data,
                   estimator = "MLR",
                   meanstructure = TRUE)
```

Request the output

Note that, although the parameter estimates are not identical to Mplus's LMS approach, they have the same pattern and direction.

```
summary(product_fit,
        standardized = TRUE,
        rsquare = TRUE
        )
```

```
## lavaan 0.6-9 ended normally after 126 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        33
##
##   Number of observations                           488
##
## Model Test User Model:
##                                                Standard      Robust
##   Test Statistic                                153.435     110.239
##   Degrees of freedom                                 32          32
##   P-value (Chi-square)                            0.000       0.000
##   Scaling correction factor                                   1.392
##        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
##   NA_var =~
##     PANASNEG          1.000                               4.661    0.606
##     neurot            1.152    0.094   12.263    0.000    5.370    0.861
##     AIMN              1.245    0.099   12.534    0.000    5.805    0.805
##   Narr =~
##     ces               1.000                               0.989    0.909
##     ptgi              0.915    0.064   14.361    0.000    0.905    0.634
##     closure           1.353    0.150    9.015    0.000    1.338    0.505
##   NAxNarr =~
##     PANASNEG.ces      1.000                               6.132    0.710
##     neurot.ptgi       0.396    0.224    1.764    0.078    2.425    0.270
##     AIMN.closure      1.246    0.834    1.494    0.135    7.643    0.375
##
## Regressions:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   pcl ~
##     NA_var            0.736    0.136    5.401    0.000    3.431    0.221
##     Narr              9.089    0.883   10.291    0.000    8.992    0.579
##     NAxNarr           0.431    0.239    1.806    0.071    2.646    0.170
##
## Covariances:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   NA_var ~~
##     NAxNarr           0.000                               0.000    0.000
##   Narr ~~
##     NAxNarr           0.000                               0.000    0.000
##   NA_var ~~
```

```
##     Narr              1.133    0.276    4.099    0.000    0.246    0.246
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     NA_var            0.000                              0.000    0.000
##     Narr              0.000                              0.000    0.000
##     NAxNarr           0.000                              0.000    0.000
##    .PANASNEG         22.284    0.348   63.962    0.000   22.284    2.895
##    .neurot           30.020    0.282  106.347    0.000   30.020    4.814
##    .AIMN             27.838    0.326   85.331    0.000   27.838    3.863
##    .ces               2.805    0.049   56.958    0.000    2.805    2.578
##    .ptgi              2.990    0.065   46.248    0.000    2.990    2.094
##    .closure           4.810    0.120   40.087    0.000    4.810    1.815
##    .PANASNEG.ces     -0.000    0.391   -0.000    1.000   -0.000   -0.000
##    .neurot.ptgi       0.000    0.406    0.000    1.000    0.000    0.000
##    .AIMN.closure      0.000    0.922    0.000    1.000    0.000    0.000
##    .pcl              32.635    0.702   46.494    0.000   32.635    2.103
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .PANASNEG         37.510    3.146   11.923    0.000   37.510    0.633
##    .neurot           10.048    1.717    5.853    0.000   10.048    0.258
##    .AIMN             18.244    2.387    7.643    0.000   18.244    0.351
##    .ces               0.205    0.061    3.363    0.001    0.205    0.173
##    .ptgi              1.221    0.094   13.000    0.000    1.221    0.599
##    .closure           5.235    0.348   15.062    0.000    5.235    0.745
##    .PANASNEG.ces     37.001   22.945    1.613    0.107   37.001    0.496
##    .neurot.ptgi      74.618    7.909    9.434    0.000   74.618    0.927
##    .AIMN.closure    356.134   45.572    7.815    0.000  356.134    0.859
##    .pcl             126.101   14.338    8.795    0.000  126.101    0.523
##     NA_var           21.721    3.188    6.813    0.000    1.000    1.000
##     Narr              0.979    0.087   11.273    0.000    1.000    1.000
##     NAxNarr          37.602   23.779    1.581    0.114    1.000    1.000
##
## R-Square:
##                    Estimate
##     PANASNEG          0.367
##     neurot            0.742
##     AIMN              0.649
##     ces               0.827
##     ptgi              0.401
##     closure           0.255
##     PANASNEG.ces      0.504
##     neurot.ptgi       0.073
##     AIMN.closure      0.141
##     pcl               0.477
```

Probe the interaction

```
probe2way <- probe2WayMC(product_fit, # fitted lavaan model
                nameX = c("NA_var", "Narr", "NAxNarr"), # X variables
                nameY = "pcl", # Y variable
                modVar = "Narr", # moderator
                valProbe = c(-1, 0, 1)) # levels of the Z value at which to probe the
interaction

probe2way
```

```
## $SimpleIntcept
##   Narr    est    se      z pvalue
## 1   -1 23.546 0.993 23.721      0
## 2    0 32.635 0.702 46.494      0
## 3    1 41.723 1.249 33.407      0
##
## $SimpleSlope
##   Narr   est    se     z pvalue
## 1   -1 0.305 0.245 1.244  0.214
## 2    0 0.736 0.136 5.401  0.000
## 3    1 1.168 0.302 3.864  0.000
```

Using factor scores from Mplus

Finally, we can use Mplus factor scores to plot interactions in R. We'll use the same packages and approaches presented in lecture, which cover both spotlight (pick-a-point) and floodlight (Johnson-Neyman) approaches.

We create the factor scores in Mplus by fitting an LMS model, then request the saved factor scores. We can use the names in the Mplus output to determine the names of the columns in the dataset.

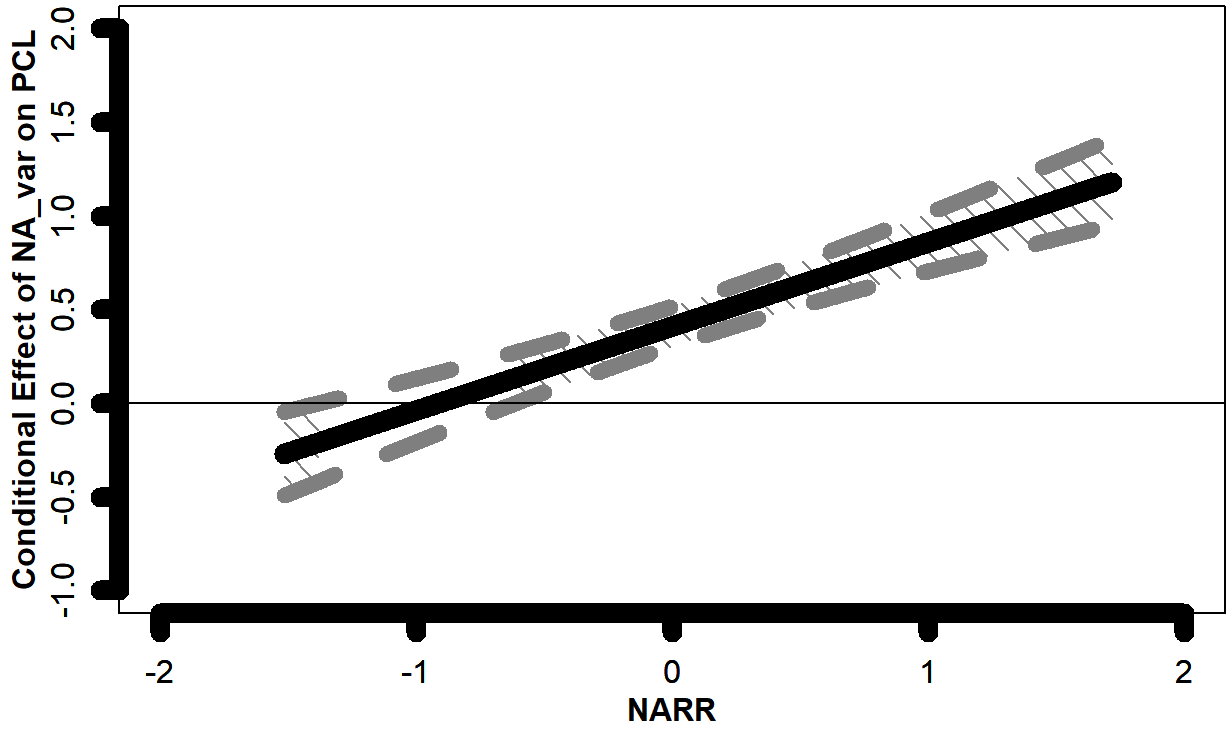
```
narr_scores <- read.csv("pcl_factorscores.csv", header = TRUE)
```

With *probemod* package

```
#install.packages("probemod")
library(probemod)
```

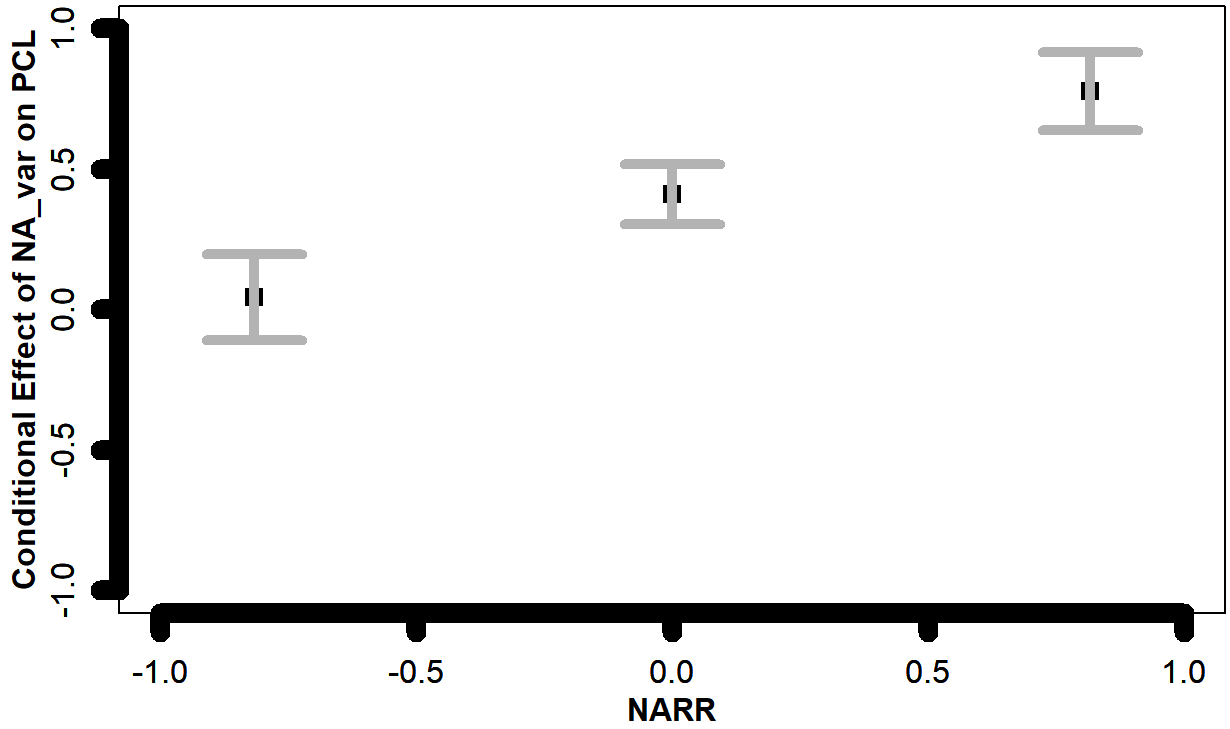
```
mod <- lm(PCL ~ NA_var + NARR + NA_var*NARR,
          data = narr_scores)
```

```
plotmod <- jn(mod, dv = "PCL", iv = "NA_var", mod = "NARR")
plot(plotmod)
```




```
## Values of NARR indicated by the shaded region
##                        x          y        se         t    p          llci
## Lower Bound: -1.3801842 -0.2087620 0.1062463 -1.964888 0.05 -4.175240e-01
## Upper Bound: -0.6127909  0.1356036 0.0690134  1.964888 0.05  2.498002e-16
##                        ulci
## Lower Bound: -2.498002e-16
## Upper Bound:  2.712071e-01
```

```
spot1 <- pickapoint(mod, dv = "PCL", iv = "NA_var", mod = "NARR", method = "meansd")
plot(spot1)
```




```
spot1
```

```
## Call:
## pickapoint(model = mod, dv = "PCL", iv = "NA_var", mod = "NARR",
##     method = "meansd")
## Conditional effects of  NA_var  on  PCL  at values of  NARR
##           NARR     Effect         SE          t            p       llci
##  -0.8163866685 0.04424052 0.07771446  0.5692701 5.694375e-01 -0.1084597
##   0.0002008197 0.41068186 0.05386281  7.6245898 1.309415e-13  0.3048475
##   0.8167883079 0.77712320 0.07102181 10.9420365 4.797968e-25  0.6375733
##       ulci
##  0.1969407
##  0.5165162
##  0.9166731
##
## Values for quantitative moderators are the mean and plus/minus one SD from the mea
n
## Values for dichotomous moderators are the two values of the moderator
```

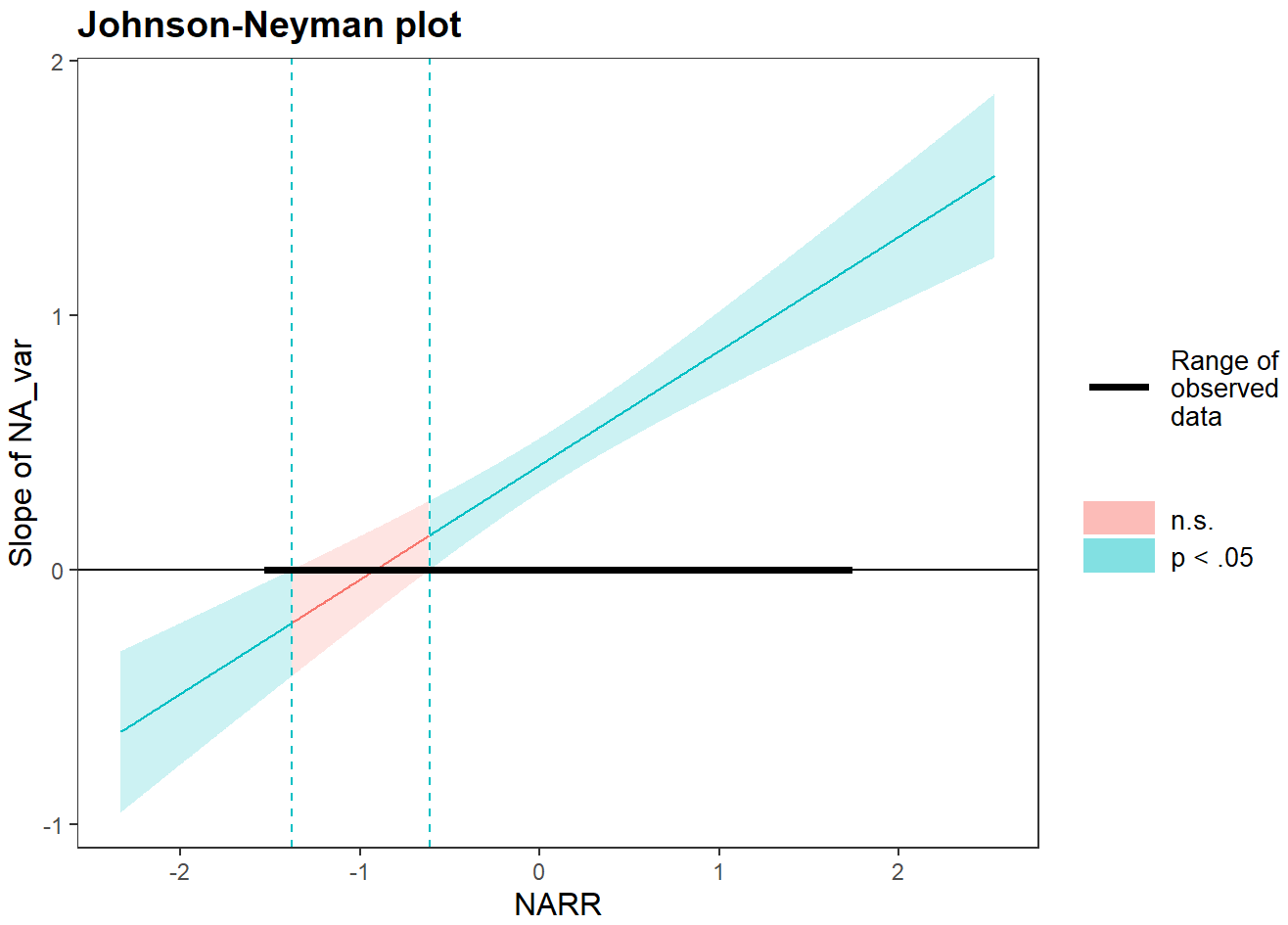
With *interactions* package

```
#install.packages("interactions")
library(interactions)
```

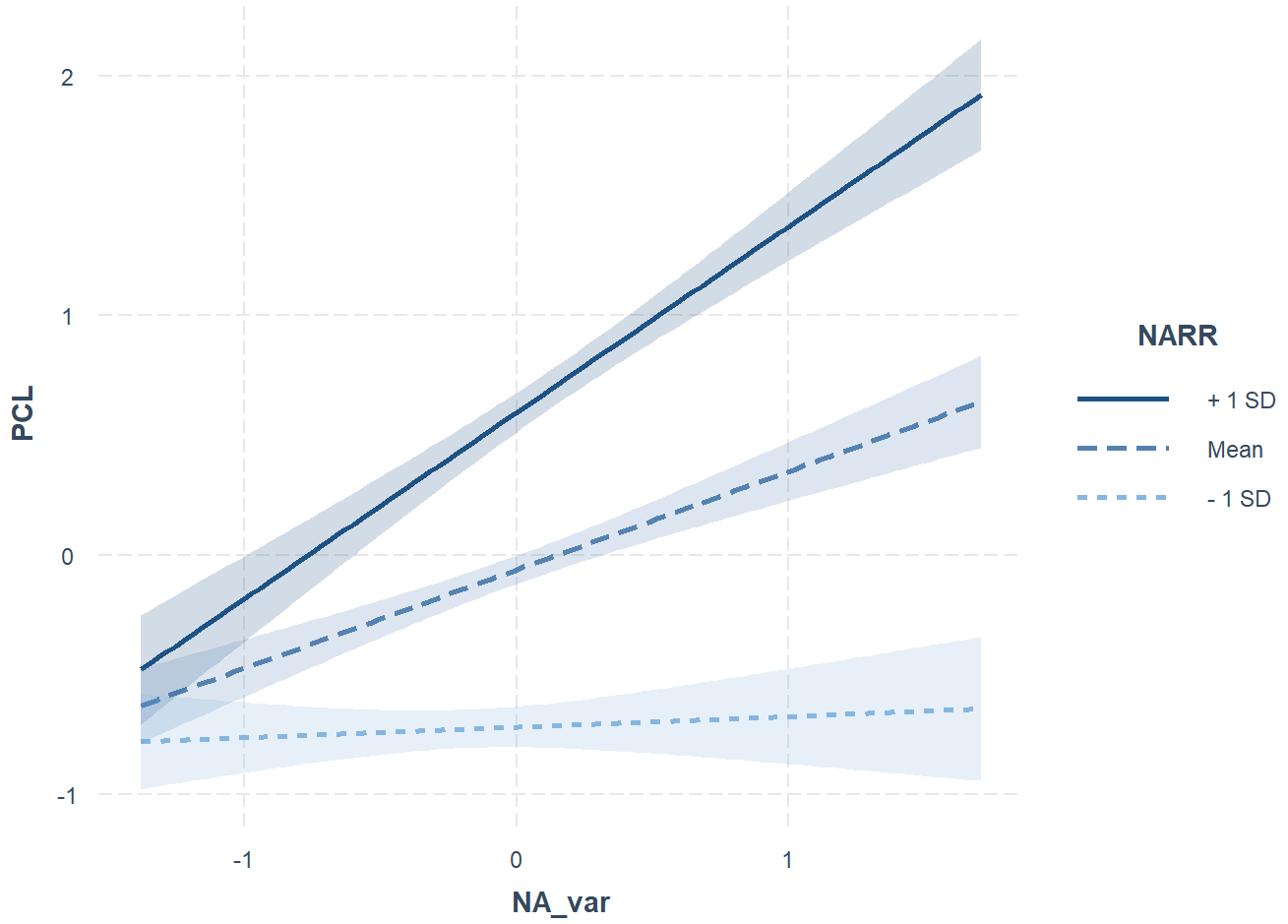
```
## Warning: package 'interactions' was built under R version 4.1.3
```

```
johnson_neyman(mod, pred = NA_var, modx = NARR)
```

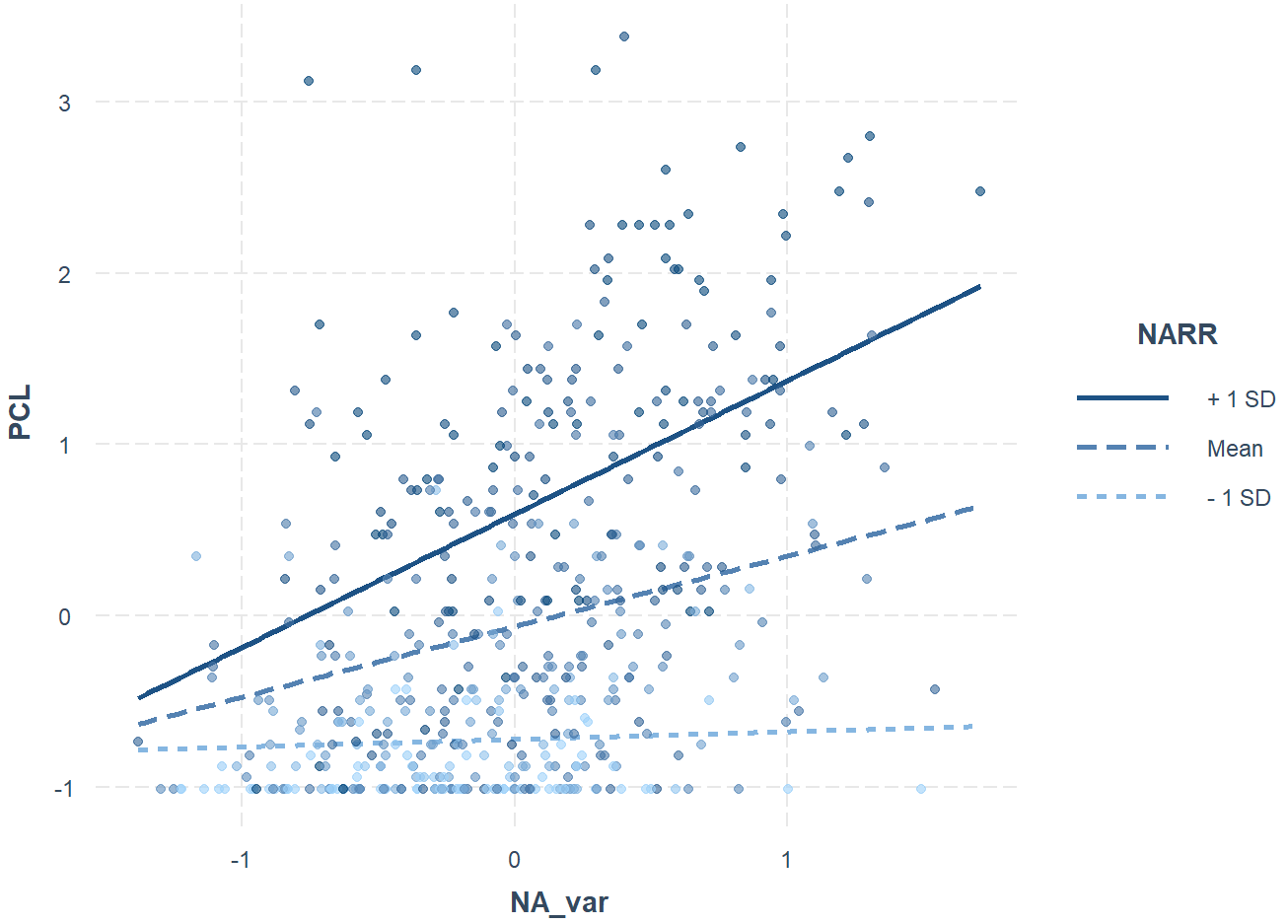
```
## JOHNSON-NEYMAN INTERVAL
##
## When NARR is OUTSIDE the interval [-1.38, -0.61], the slope of NA_var is p
## < .05.
##
## Note: The range of observed values of NARR is [-1.52, 1.72]
```




```
interact_plot(mod, pred = NA_var, modx = NARR, interval = TRUE)
```




```
interact_plot(mod, pred = NA_var, modx = NARR, plot.points = TRUE)
```



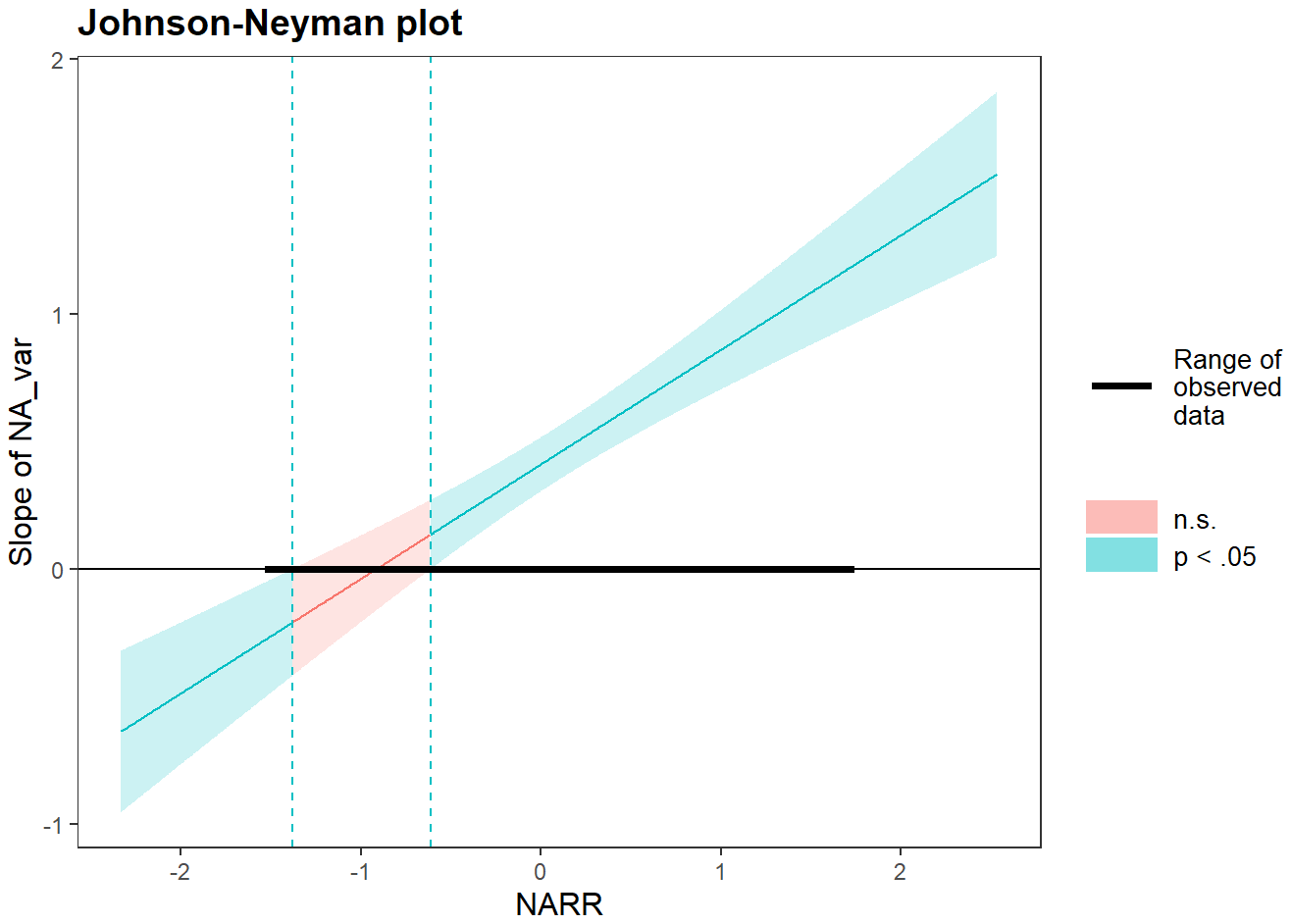
With *sandwich* package

```
library(sandwich)
```

```
## Warning: package 'sandwich' was built under R version 4.1.3
```

```
sim_slopes(mod, pred = NA_var, modx = NARR, jnplot = TRUE)
```

```
## JOHNSON-NEYMAN INTERVAL
##
## When NARR is OUTSIDE the interval [-1.38, -0.61], the slope of NA_var is p
## < .05.
##
## Note: The range of observed values of NARR is [-1.52, 1.72]
```




```
## SIMPLE SLOPES ANALYSIS
##
## Slope of NA_var when NARR = -0.8163866685 (- 1 SD):
##
##   Est.   S.E.   t val.      p
## ------ ------ -------- ------
##   0.04   0.08     0.57   0.57
##
## Slope of NA_var when NARR =  0.0002008197 (Mean):
##
##   Est.   S.E.   t val.      p
## ------ ------ -------- ------
##   0.41   0.05     7.62   0.00
##
## Slope of NA_var when NARR =  0.8167883079 (+ 1 SD):
##
##   Est.   S.E.   t val.      p
## ------ ------ -------- ------
##   0.78   0.07    10.94   0.00
```