To store the orthogonal polynomial regression coefficients (OPRC) an empty matrix is set up with k rows and the number of columns equal to the highest order polynomial, previously specified as maxpoly.

poly\_coef = data.frame(matrix(0, ncol = maxpoly, nrow = k))

The choice of these dimensions becomes clear when the following code is regarded:

for (i in 1:maxpoly) {

pfv = paste("poly\_fit\_", i, sep = "")

poly = assign(pfv, lm(rma\_data\_long$value ~ poly(rma\_data\_long$condition, degree = i, raw = TRUE)))

poly\_coef[, i][1:(i + 1)] = poly$coef

}

In each cycle of the for loop, that runs from 1 to the maximal polynomial order (maxpoly), the following steps are executed:

First, orthogonal polynomial regressions are estimated for the observed values (pulled weights) in long format suing the poly() function within lm() . The degree of the highest polynomial is specified by the index i. The model always contains an intercept as well as coefficients for the data to the i-th polynomial. The result, which is an object of type lm, is stored in an object called poly which is overwritten in each cycle.

In the second step, the coefficients are extracted from poly and stored in the previously prepared matrix (poly\_coef) such that the coefficients are assigned to the i-th column. The subsetting is cruicial here to ensure that the dimensions in the assignment coincide. For example, in the second cycle, the second column of the data frame is chosen. This column is then filled with the intercept coefficient, the first order polynomial coefficient and the second order polynomial coefficient. Since in this cycle there is no higher order, the last element in the data frame remains equal to zero. The resulting data frame looks as follows:

X1 X2 X3

1 19.668109 4.712768 -60.057040

2 -4.650561 10.304780 113.319617

3 0.000000 -2.991068 -49.255216

4 0.000000 0.000000 6.168553

Here, one can see that with each cycle of the loop the number of nonzero coefficients increases by one. So the first row stores the coefficients for the intercept, the second row the coefficients for the first order coefficient. The last row always contains the highest order coefficient (here: 3). The columns represent the resulting regression equations and can be seen as follows:

C1: b0 + b1X

C2: b0 + b1X + b2X^2

C3: b0 + b1X + b2X^2 + b3X^3

First, a vector x is defined as a sequence from one to k, the number of factor levels in steps of k/1000. Next, the outer() function is used to create a matrix that takes x to the power of 0 to maxpoly. The result is stored in the object poly\_values.

x = seq(1, k, length.out = 1000)

poly\_values = outer(x, 0:(k - 1), `^`)

The first six rows of the resulting matrix look as following:

[,1] [,2] [,3] [,4]

[1,] 1 1.000000 1.000000 1.000000

[2,] 1 1.003003 1.006015 1.009036

[3,] 1 1.006006 1.012048 1.018126

[4,] 1 1.009009 1.018099 1.027271

[5,] 1 1.012012 1.024168 1.036471

[6,] 1 1.015015 1.030255 1.045725

One can see that the first column contains the values for the intercept, just like in a regular linear regression setup. These values are created for the polynomial regression curves that are about to be plotted. Next, a data frame is created containing the vector x and the matrix product of the data matrix poly\_values and the regression coefficients.

poly\_curve\_data = data.frame(x, poly\_values %\*% poly\_coef)

The first six rows of the resulting matrix look as following:

x X1 X2 X3

1 1.000000 15.017548 12.02648004 10.17591410

2 1.003003 15.003583 12.03943397 10.27568155

3 1.006006 14.989617 12.05233397 10.37489539

4 1.009009 14.975651 12.06518001 10.47355664

5 1.012012 14.961686 12.07797211 10.57166629

6 1.015015 14.947720 12.09071026 10.66922535

The columns X1, X2 and X3 contain the estimated y-values from the regression equations stated in XXX for the corresponding x value in the column x.

The final step is to transform the data to long format which makes plotting with ggplot2 more convenient.

poly\_curve\_data = gather(poly\_curve\_data, line, y, -x)

The resulting data frame is now in long format:

x curve y

1 1.000000 X1 15.01754820

2 1.003003 X1 15.00358255

3 1.006006 X1 14.98961690

4 1.009009 X1 14.97565125

5 1.012012 X1 14.96168560

6 1.015015 X1 14.94771996

The column x and y contain the x- and y-values, respectively and the column curve specifies the regression curve, that the values belong to.

The final plot is created using ggplot2 (Wickham, 2016).

poly\_plot = ggplot(data = rma\_data\_long, aes(x = condition, y = value)) + geom\_point() + labs(col = "Order of \npolynomial", x = "Condition", y = "Value", title = "Orthogonal polynomial contrasts") +

geom\_path(data = poly\_curve\_data, aes(x, y, color = curve), lwd = 1.2) + scale\_color\_discrete(labels = as.character(1:(maxpoly))) +

theme\_bw() + theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(), legend.key = element\_rect(colour = "black"), plot.title = element\_text(face="bold", hjust = .5))

A detailed resource for the use of ggplot2 is provided by Wickham (2016). Here, it is worth noticing that the code makes use of ggplot2’s ability to plot data from multiple data frames in one plot. On the x-axis there are the factor levels, i.e. the groups from one to four. The y-axis displays the observed values, in this case the individually pulled weight. The data frame rma\_data\_long is used to plot the observed values of the subjects at the different factor levels. The regression curves are plotted using the geom\_path() layer function. The curves are fitted smoothly through the 1000 x-values between 1 and 4 for each regression equation. Here, it is worth noticing that if the number of values for the regression curves is chosen too small, the curves might become wiggly. The color is mapped to the (categorical) curve variable such that each regression curve has a different color. Specifying the color argument withing the aes() function also initializes a legend. The other arguments are used to format the plot.

LITERATURE

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis* (2nd ed. 2016). New York, NY: Springer.