# A first R analysis

*Guido Biele*

*16 november 2018*

## Introduction

This document aims to provide a shirt introduction into statistical analysis with R. The assumption is, that the reader is already familiar with basic concepts in R and the RStudio IDE, and has some basic knowledge about statistics, for example from using SPSS. The goal here is to show how one can

- open an SPSS file
- merge separate data sets
- visually inspect the data
- do a simple data reduction through an exploratory factor analysis
- estimate the association between an exposure and an outcome with a linear regression
- visualize the results of this regression.

## Packages

The name R refers at the same time to 3 things:

1. A programming language
2. A software for statistical analyses
3. An ecosystem of *packages* that implement different statistical analyses.

R packages are written (typically by using R, the programming language) by researchers and available for free. These packages are crucial, because the capabilities of The basic R software are limited.

There are thousands of R packages. Therefore, CRAN task view web sites are useful to give an overview about R packages for different application areas. CRAN stands for "Comprehensive R Archive Network".

For this introduction we will use following packages:

- `haven` allows to read SPSS (or Stata) data into R
- `ggplot` facilitates plotting
- `psych` is a package for factor analysis and related things
- `ggplot` and `cowplot` for general plotting
- `sjPlot` and `ggeffect` for plotting of regression analysis results
- `VGAM` allows doing regression with more unusual response models, like for example the beta binomial distribution.

Installing packages is easy!

```
install.packages("haven")
```

One can also install multiple packages simultaneously as follows.

```
install.packages(c("ggplot2", "cowplot",
                   "psych", "qualityTools"
                   "sjPlot", "ggeffects",
                   "VIM", "apaTables".
                   "VGAM")
                  )
```

*Some code chunks in this tutorial cannot be exectuted with the data in this repository, because I did not create synthesized data in SPSS format. All chunks that cannot be executed have set the option* `eval=FALSE`. The script I used to create synthetic data can be found here:

## The running a regression and plotting results

When we do statistical analysis, we often think about how to choose covariates to include into a regression, which type of regression (linear, logistic, ..) to do, and some pray to God that $p < .05$. However, lots of time is spent on the much less glamorous part of cleaning the data and prepare them for the analysis ( *data wrangling* ). Data wrangling will be an important part of this tutorial. I would also say that R is very good for doing this. But because some could think this is boring, we'll start with a quick regression model.

We load the data. and look at the first few rows.

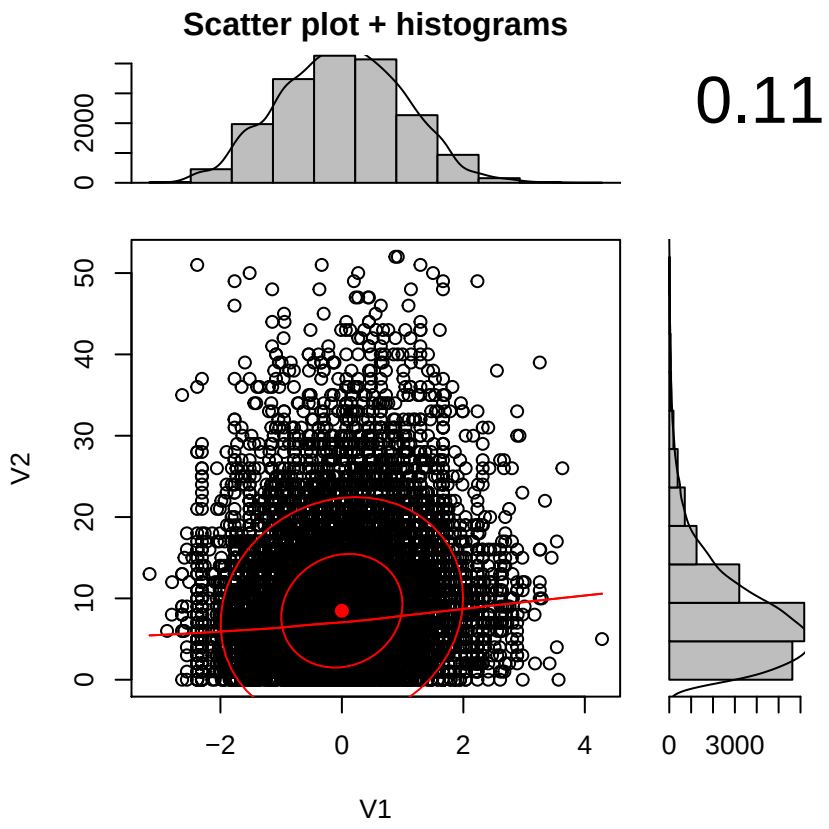
```
load("data/mysdata.Rdata")
head(my_data)
```

```
##   PREG_ID_439 BARN_NR   positive inconsistent OD ADHD CD SMFQ CCC SPRAAK
## 1       41263       1  0.7557911   -0.2130634  3    8  1    5  24      1
## 2       41271       1  0.9597752   -1.1466486  6   44  1    8  29     14
## 3       41279       1  0.9576270    1.3839585  9   11  0    5  21      0
## 4       41283       1 -0.8054542   -1.2791529  4   21  4    4  25      0
## 5       41391       1  1.0403413   -1.5187908 14   22  3    8  27      6
## 6       41519       1  0.8959180    1.6646800  7   22  1    4  29      7
##   gender mAge parity birthmonth         mEDU            fEDU   mAgeg
## 1    boy   34      1         12     Bachelor        Bachelor (30,35]
## 2   girl   26      1         12 general high vocational high (25,30]
## 3    boy   30      0         11     Bachelor        Bachelor (25,30]
## 4    boy   28      2         11     Bachelor        Bachelor (25,30]
## 5    boy   21      0         12 general high          Master (20,25]
## 6    boy   26      1         12     Bachelor vocational high (25,30]
```

Now we run a simple linear regression. Our *exposure* are factor scores for positive and inconsistent parenting (mean = 0, sd = 1). These are calculated from a factor analysis of MoBa questions about parenting style at age 5. Our *outcome* is a sum-score for ADHD symptoms from the questionnaire at age 8. We also adjust for gender.

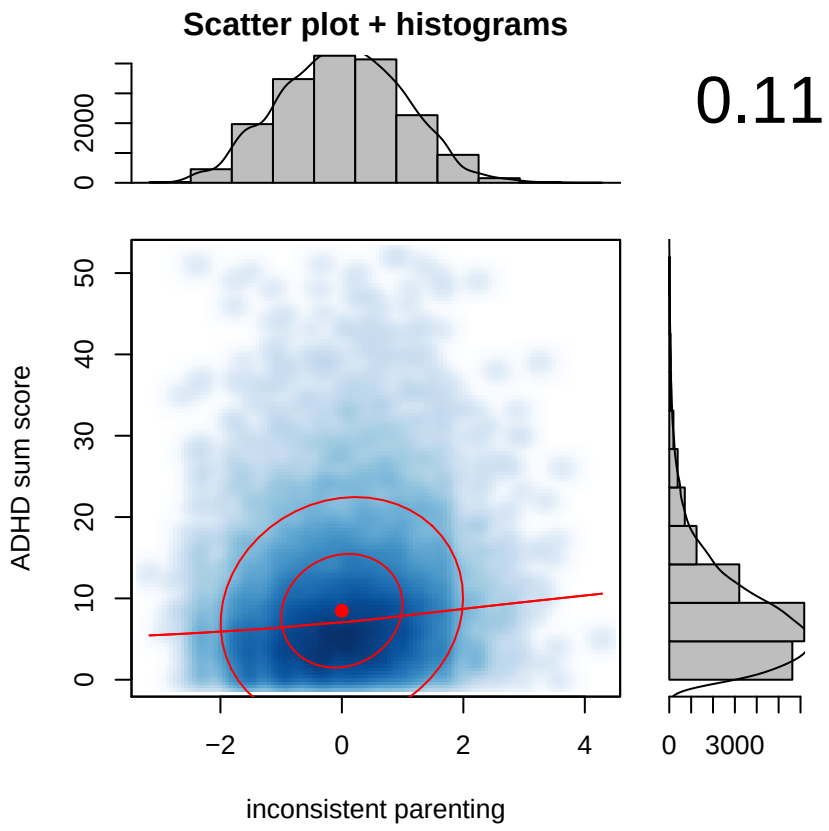
Before we run the regression, we can inspect the correlation of the variables

```
library(psych)
scatter.hist(my_data$inconsistent,
             my_data$ADHD)
```




The default option generally works. But by specifying a few extra things, one gets a better plot:

```
library(psych)
scatter.hist(my_data$inconsistent,
             my_data$ADHD,
             smoother = T,
             ylab = "ADHD sum score",
             xlab = "inconsistent parenting")
```




This is how a linear regression is specified in R. Thew tilde ~ separates criterion and predictor variables.

```
my_fit = lm(ADHD ~ positive + inconsistent + gender,
            data = my_data)
summary(my_fit)
```

```
##
## Call:
## lm(formula = ADHD ~ positive + inconsistent + gender, data = my_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.274  -4.555  -1.500   2.694  44.856
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.64199    0.07126  135.30   <2e-16 ***
## positive     -0.83694    0.05136  -16.30   <2e-16 ***
## inconsistent  0.83702    0.05120   16.35   <2e-16 ***
## gendergirl   -2.38324    0.10206  -23.35   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.789 on 17713 degrees of freedom
## Multiple R-squared:  0.05665,    Adjusted R-squared:  0.05649
## F-statistic: 354.5 on 3 and 17713 DF,  p-value: < 2.2e-16
```

*Try your won regression using another outcome by copying the formula above into the Concole below!*

This result display looks kind of complicated. Lets make a table in APA format from this:

```
library(apaTables)
apa.reg.table(my_fit, filename = "reg_table_APA.doc", table.number = 2)
```

```
##
##
## Table 2
##
## Regression results using ADHD as the criterion
##
##
##     Predictor       b       b_95%_CI sr2 sr2_95%_CI             Fit
##   (Intercept)  9.64**   [9.50, 9.78]
##      positive -0.84** [-0.94, -0.74] .01 [.01, .02]
##  inconsistent  0.84**   [0.74, 0.94] .01 [.01, .02]
##    gendergirl -2.38** [-2.58, -2.18] .03 [.02, .03]
##                                                        R2 = .057**
##                                                   95% CI[.05,.06]
##
##
## Note. A significant b-weight indicates the semi-partial correlation is also significant.
## b represents unstandardized regression weights.
## sr2 represents the semi-partial correlation squared.
## Square brackets are used to enclose the lower and upper limits of a confidence interval.
## * indicates p < .05. ** indicates p < .01.
##
```

A nicely formatted table was also saved in a MS Word file named reg_table_APA.doc!.

But where is it? Type `getwd()`in the console to see what your current working directory is.

It is still hard to understand the impact of parenting style on ADHD. The regression coefficients tell us that with a standard deviation increase of inconsistent parenting, the ADHD sum-score increases by 1 point.

We can calculate predicted values at different levels of the exposure to get a better feeling for what that means:

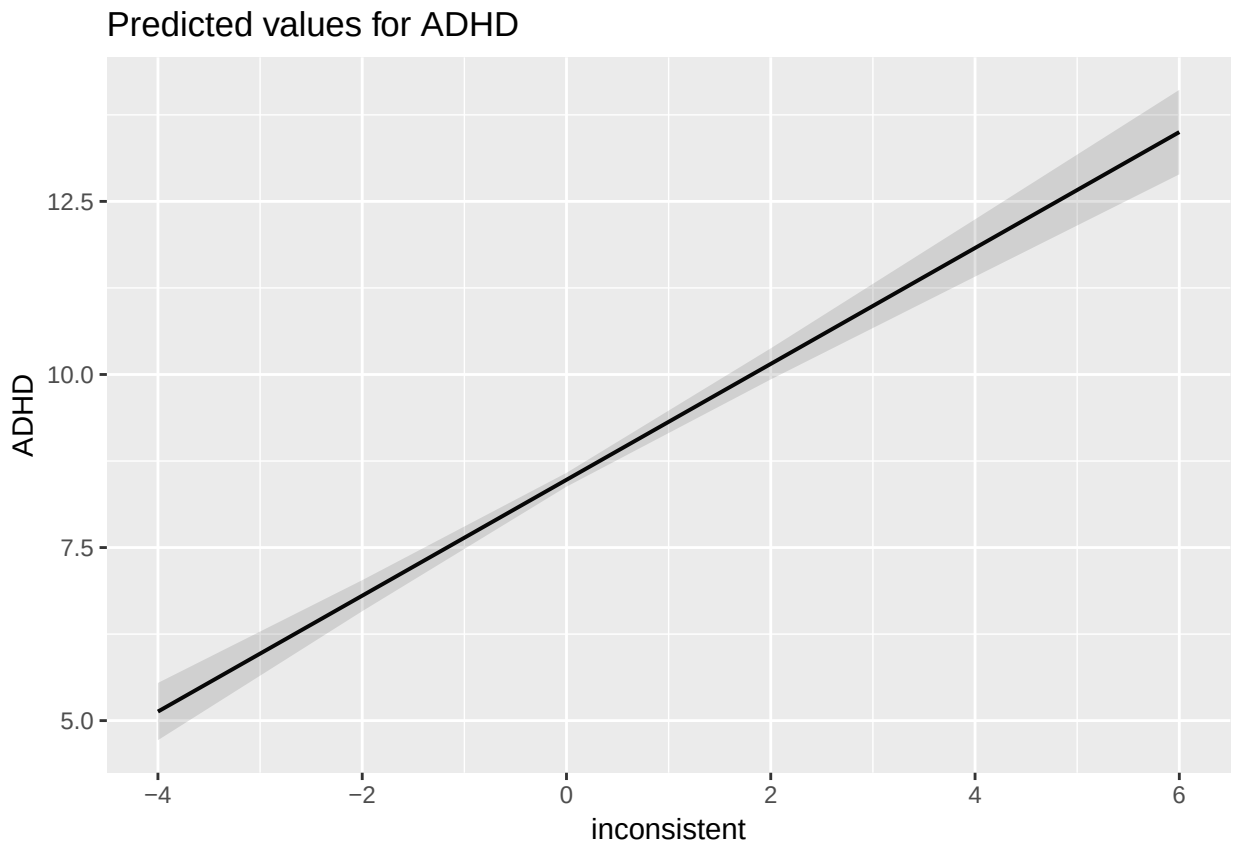
```
library(ggeffects)
ggeffect(my_fit, term = "inconsistent")
```

```
##
## # Predicted values for ADHD
## # x = inconsistent
##
##   x predicted conf.low conf.high
##  -4     5.131    4.717     5.544
##  -2     6.805    6.581     7.029
##   0     8.479    8.379     8.579
##   2    10.153    9.929    10.377
##   4    11.827   11.413    12.241
##   6    13.501   12.891    14.111
```

And even better, we can plot them:

```
library(sjPlot)
plot_model(my_fit,
```

```
          type = "eff",
          terms = "inconsistent")
```




## Loading data

The problem we are looking at is the association between Parenting Style and child mental health symptoms. More specifically, we look at the association between self reported Parenting Style of well educated Norwegian mothers at child age 5 years and oppositional behavior as well as mood and feeling at age 8. This data can be obtained from MoBa questionnaires 5 and 8.

If these data are available in SPSS (or Stata) format, we can read them with the `haven` package, which we first have to load before we can use it:

```
library(haven)
library(VIM)
```

All packages come with a documentation, and if one is lucky also with one or more *vignettes*, which are tutorial-style documents that show how the functions in the package can be used. This and more information about the packages can be found at https://cran.rstudio.com/web/packages/haven/. For other packages, just replace "haven" with the name of the package you are interested in.

OK, now we can load the data:

```
##  [1] "PREG_ID_439" "BARN_NR"     "LL526"       "LL527"       "LL528"
##  [6] "LL529"       "LL530"       "LL531"       "LL532"       "LL533"
## [11] "LL534"
```

```
data_directory = "data/"
Q5aar = read_sav(paste0(data_directory,
                        "PDB439_Skjema5aar_v10.sav"))
Q5aar = data.frame(Q5aar)
```

So what have we done here? We first created a new variable `data_directory` and set it to the path to the directory where the data files are. Then, when loading the data into the data.frame `Q5aar`, we collated the path to the directory with the name of the data file by using the `paste0`command.

The reason for doing this is that the code becomes a bit easier to read (still not great ). For the same reason, I made a line break at the end of `Q5aar = read_sav(paste0(data_directory,`.

Note that if we would already be in the same directory as the data file, we could just have written `Q5aar = read_sav("PDB439_Skjema5aar_v10.sav")`. You can use the commands `getwd()` and `setwd()` to check and set you working directory (wd).

Next we select the variables we need. This are the pregnancy ID and the child number, as well as the items for measuring parental style (we get SPSS variable names from here).

```
Q5aar = Q5aar[,c("PREG_ID_439","BARN_NR",
                 paste0("LL",526:534))]
```

Lets look the data we have.

```
head(Q5aar)
```

```
##      PREG_ID_439 BARN_NR LL526 LL527 LL528 LL529 LL530 LL531 LL532 LL533
## 7262       41263       1     5     3     4     1     5     5     5     4
## 7265       41271       1     5     1     4     2     5     5     5     4
## 7271       41279       1     4     2     5     4     5     5     5     5
## 7275       41283       1     4     2     4     2     5     4     4     4
## 7339       41391       1     5     1     4     1     5     5     5     4
## 7341       41393       1     5     2     5     2     4     5     4     4
##      LL534
## 7262     1
## 7265     1
## 7271     3
## 7275     1
## 7339     1
## 7341     2
```

OK, these variable names are not very clear, lets rename them:

```
names(Q5aar)[3:11] = paste0("PS.item.",1:9)
names(Q5aar)
```

```
##  [1] "PREG_ID_439" "BARN_NR"     "PS.item.1"   "PS.item.2"   "PS.item.3"
##  [6] "PS.item.4"   "PS.item.5"   "PS.item.6"   "PS.item.7"   "PS.item.8"
## [11] "PS.item.9"
```

It is important to understand the coding of the variables. The `haven` package makes SPSS' *Variable label* and *Value label* information available as attributes of the variables in the data.frame. We can access those as follows:

```
attr(Q5aar$PS.item.1,"label")
```

```
## [1] "W_53_1:SKJEMA_5AARB; Du lar barnet ditt forstå når han/hun gjør en god innsats; 53. Hvor ofte h
```

```
attr(Q5aar$PS.item.1,"labels")
```

```
## Mer enn ett kryss             Aldri      Nesten aldri       Noen ganger
##                 0                 1                 2                 3
##              Ofte            Alltid
##                 4                 5
```

Here we see one annoying detail of SPSS Moba data: If the mother has ticked more than one box, the data are for our purposes missing, and not zero. We need to fix that.

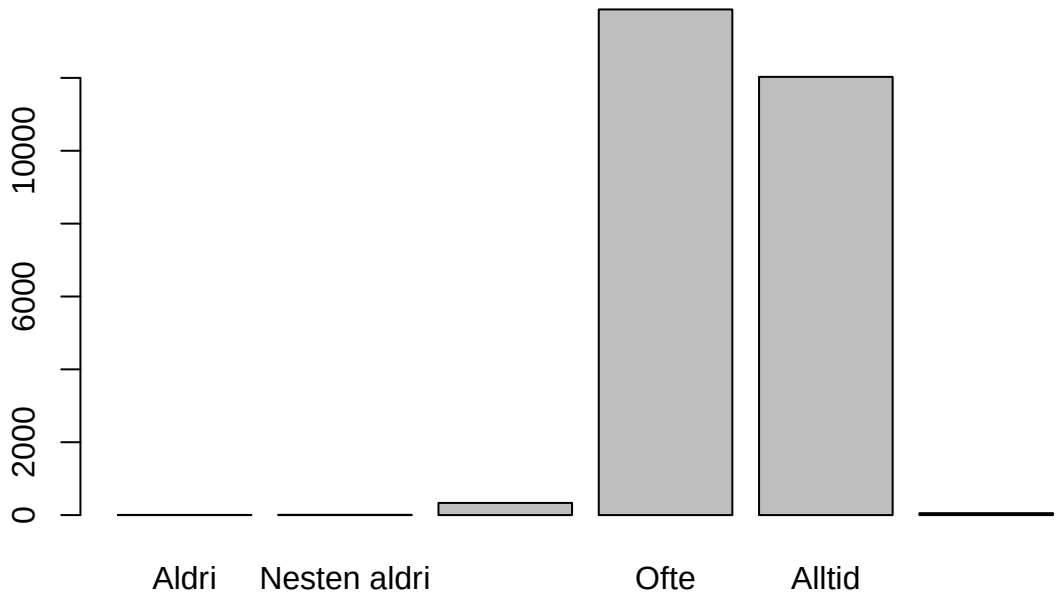
To do this, we do a quick for loop:

```
item_names = names(Q5aar)[-c(1,2)]
item_quest = c()
for (v in item_names) {
  item_quest = c(item_quest,
                 strsplit(attr(Q5aar[,v],"label"),
                          split = ";")[[1]][2])
  attr(Q5aar[, v],"labels") = attr(Q5aar[, v],"labels")[-1]
  # need to use the "which", otherwise R stumbles over missing values
  value_is_0 = which(Q5aar[,v] == 0)
  Q5aar[value_is_0, v] = NA
}
table(as_factor(Q5aar$PS.item.1))
```

```
##
##        Aldri Nesten aldri  Noen ganger         Ofte       Alltid
##            5            7          334        13879        12028
```

Missing data is always a problem, especially with questionnaire data from long lasting studies such as MoBa. Lets look at missing data by showing a table that counts them:

```
table_plus_missing = table(addNA(as_factor(Q5aar$PS.item.1)))
barplot(table_plus_missing)
```



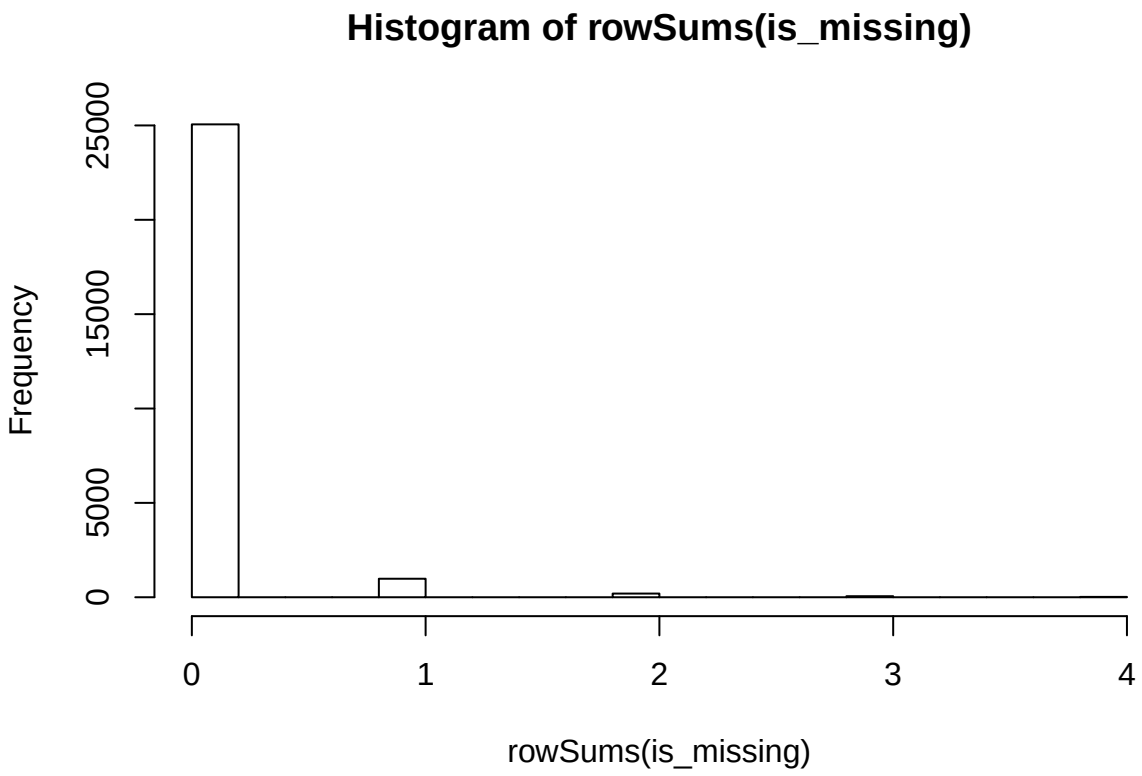

```
table_plus_missing
```

```
##
##        Aldri Nesten aldri  Noen ganger         Ofte       Alltid
##            5            7          334        13879        12028
##         <NA>
##           53
```

There is lots of missing data, probably because MoBa changed the Parenting Style items throughout.

Lets look a little closer at the missing data:

```
is_missing = is.na(Q5aar[,-c(1,2)])
hist(rowSums(is_missing))
```




Indeed there is a substantial number of cases for which all Parenting style items are missing. Let's remove those participants where more than 50% of parenting style items are missing, and impute the missing values for the remaining participants. We use the function `hotdeck`from the package `VIM` to impute values

```
library(VIM)
Q5aar = Q5aar[rowMeans(is_missing[,item_names]) < .5,]
ps_imputations = hotdeck(as.matrix(Q5aar[,item_names]))
Q5aar[,item_names] = ps_imputations[,item_names]
```

## A quick exploratory factor analysis of parenting style items in MoBa

Our goal is to reduce the 9 items to a smaller set of variables describing parenting style. One way to do this is to use exploratory factor analysis (EFA) to learn the factor structure of the data. As a first step we'll check how many factors we need, using the `fa.parallel.poly` command from the `psych` package.

As with many things in R, using up to date methods is not hard, as you'll see. The problem is more in knowing that these methods exist. If you do not know which method to use, try googeling it. The challenge is in finding out which of the many similar packages one should use. Here are a few tips:

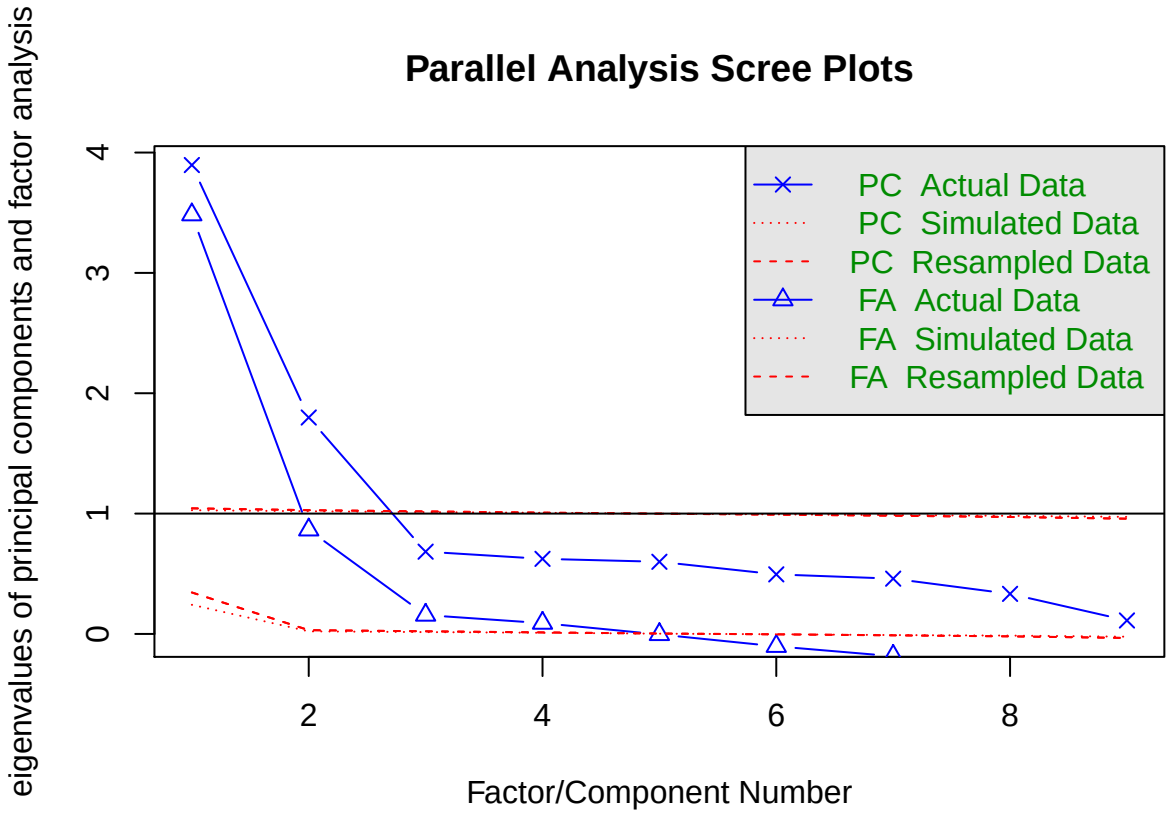
- Google R tutorial exploratory factor analysis,
- If a paper about a package is published in the Journal of Statistical Software, this is probably one of the better packages. Don't worry about the name of the Journal! The articles typically include a tutorial about how to use the key functions.

Now lets move on with the exploratory factor analysis. One thing we need to specify is that polychoric correlations are used, because we have ordinal data.

```r
library(psych)
describe(Q5aar[,item_names])
```

```
##           vars     n mean   sd median trimmed  mad min max range  skew
## PS.item.1    1 26306 4.44 0.53      4    4.45 0.00   1   5     4 -0.12
## PS.item.2    2 26306 2.14 0.77      2    2.14 1.48   1   5     4  0.14
## PS.item.3    3 26306 4.19 0.46      4    4.15 0.00   1   5     4  0.57
## PS.item.4    4 26306 2.05 0.84      2    2.02 1.48   1   5     4  0.34
## PS.item.5    5 26306 4.77 0.45      5    4.85 0.00   1   5     4 -1.81
## PS.item.6    6 26306 4.64 0.49      5    4.68 0.00   1   5     4 -0.70
## PS.item.7    7 26306 4.47 0.57      5    4.50 0.00   1   5     4 -0.51
## PS.item.8    8 26306 4.22 0.59      4    4.25 0.00   1   5     4 -0.22
## PS.item.9    9 26306 2.33 0.77      2    2.37 1.48   1   5     4 -0.10
##           kurtosis   se
## PS.item.1    -0.96 0.00
## PS.item.2    -0.43 0.00
## PS.item.3     1.23 0.00
## PS.item.4    -0.43 0.01
## PS.item.5     3.51 0.00
## PS.item.6    -1.16 0.00
## PS.item.7    -0.54 0.00
## PS.item.8     0.25 0.00
## PS.item.9    -0.44 0.00
```

```r
fa.parallel(Q5aar[,item_names], cor = "poly")
```




```
## Parallel analysis suggests that the number of factors =  4  and the number of components =  2
```

4 factors. I had hoped fewer, but lets continue and do an exploratory factor analysis.

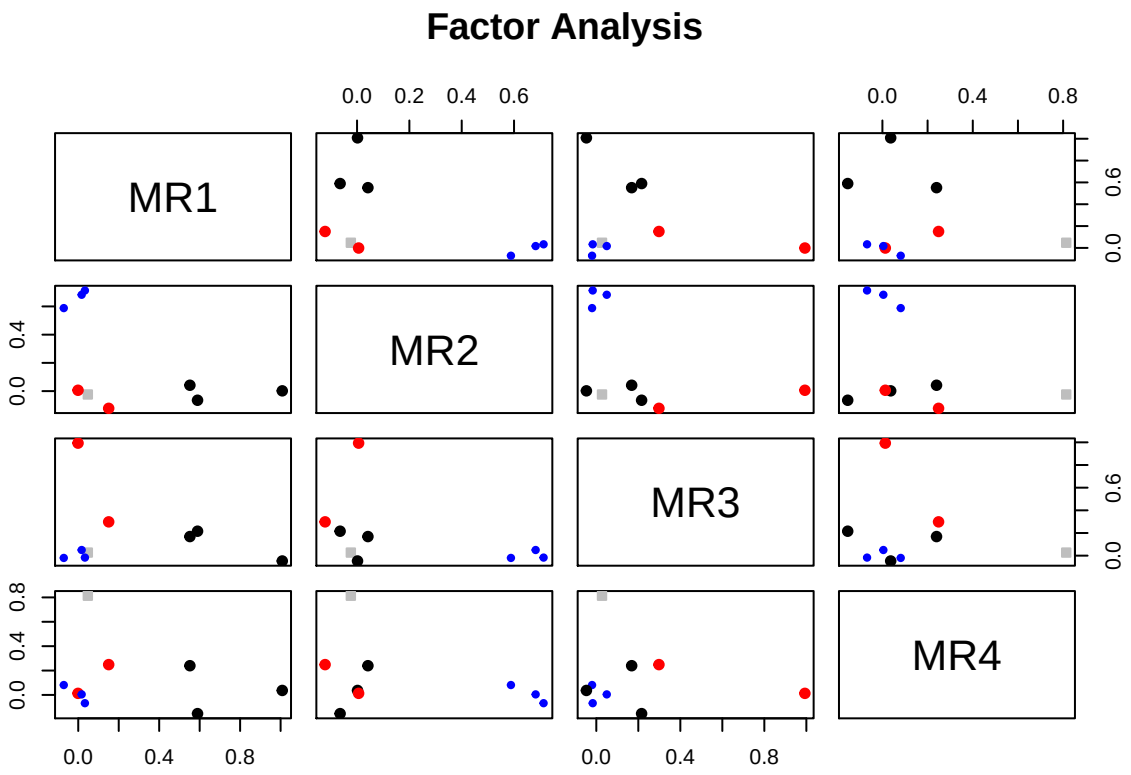
```
fa_ps = fa(Q5aar[,item_names],
           nfactors = 4,
           cor = "poly")
fa_ps
```

```
## Factor Analysis using method =  minres
## Call: fa(r = Q5aar[, item_names], nfactors = 4, cor = "poly")
##
##  Warning: A Heywood case was detected.
## Standardized loadings (pattern matrix) based upon correlation matrix
##             MR1   MR2   MR3   MR4   h2      u2 com
## PS.item.1  0.05 -0.02  0.03  0.81 0.76  0.2447 1.0
## PS.item.2  0.03  0.71 -0.02 -0.07 0.53  0.4724 1.0
## PS.item.3  0.15 -0.12  0.30  0.25 0.40  0.6004 2.8
## PS.item.4 -0.07  0.59 -0.02  0.08 0.34  0.6562 1.1
## PS.item.5  0.59 -0.06  0.22 -0.15 0.42  0.5806 1.4
## PS.item.6  1.01  0.00 -0.05  0.04 1.01 -0.0148 1.0
## PS.item.7  0.55  0.04  0.17  0.24 0.73  0.2687 1.6
## PS.item.8  0.00  0.01  0.99  0.01 1.00  0.0035 1.0
## PS.item.9  0.02  0.68  0.05  0.00 0.46  0.5415 1.0
##
##                        MR1  MR2  MR3  MR4
## SS loadings           1.95 1.36 1.35 0.99
## Proportion Var        0.22 0.15 0.15 0.11
## Cumulative Var        0.22 0.37 0.52 0.63
## Proportion Explained  0.35 0.24 0.24 0.18
## Cumulative Proportion 0.35 0.59 0.82 1.00
##
##  With factor correlations of
##       MR1   MR2   MR3   MR4
## MR1  1.00 -0.15  0.61  0.74
## MR2 -0.15  1.00 -0.10 -0.21
## MR3  0.61 -0.10  1.00  0.50
## MR4  0.74 -0.21  0.50  1.00
##
## Mean item complexity =  1.3
## Test of the hypothesis that 4 factors are sufficient.
##
## The degrees of freedom for the null model are  36  and the objective function was  4.19 with Chi Squa
## The degrees of freedom for the model are 6  and the objective function was  0.06
##
## The root mean square of the residuals (RMSR) is  0.01
## The df corrected root mean square of the residuals is  0.03
##
## The harmonic number of observations is  26306 with the empirical chi square  257.29  with prob <  1.1
## The total number of observations was  26306  with Likelihood Chi Square =  1643.02  with prob <  0
##
## Tucker Lewis Index of factoring reliability =  0.911
## RMSEA index =  0.102  and the 90 % confidence intervals are  0.098 0.106
## BIC =  1581.95
## Fit based upon off diagonal values = 1
```

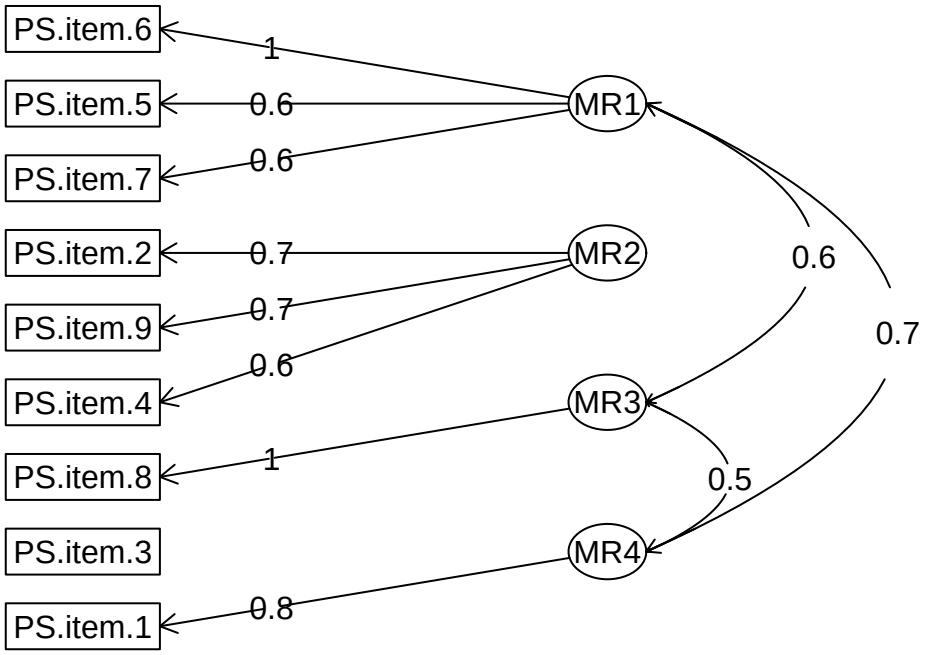
We can at first look at the RMSEA. The value is 0.06, which is not great but OK.

This are the factor loadings:

```
plot(fa_ps)
```

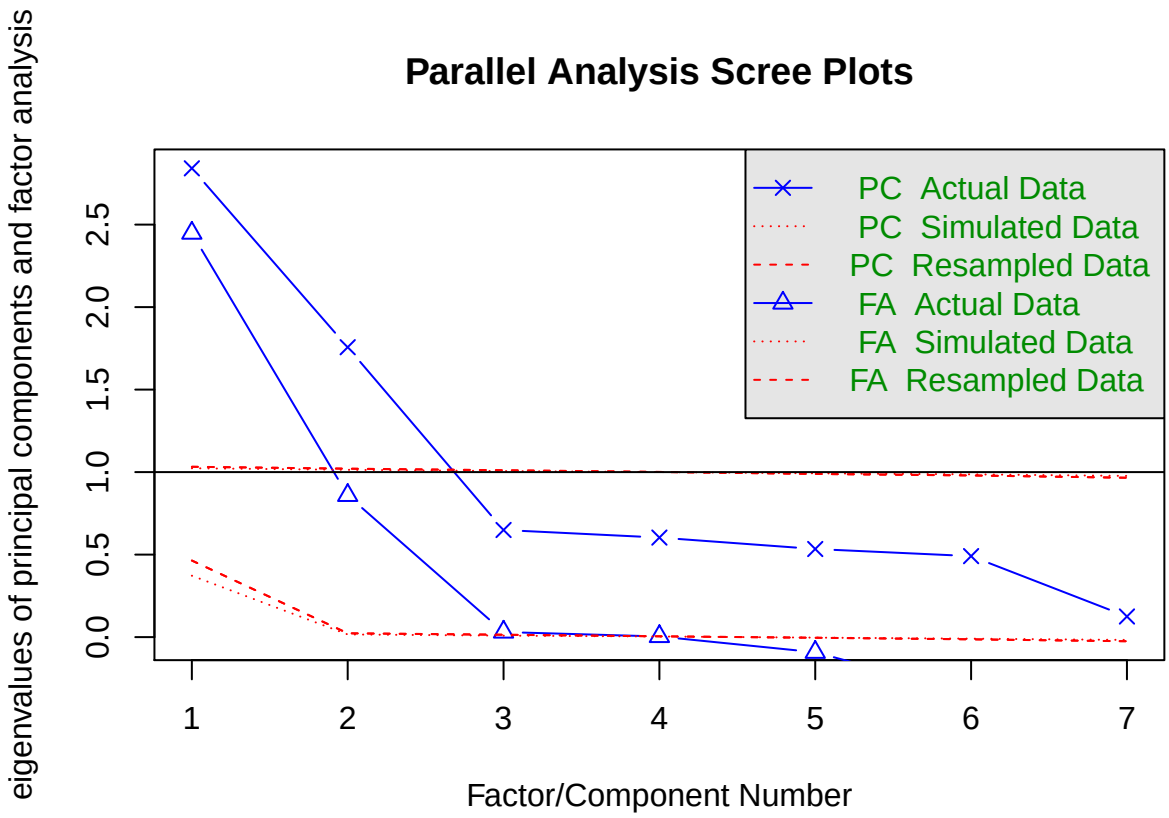



```
fa.diagram(fa_ps)
```




This is not very clear, because a number of items (especially item 3) have high loadings for multiple factors. Maybe even more problematically, items 1 and 8 stand apart. Lets try to redo this without these items.

```
reduced_items = item_names[-c(1,8)]
fa.parallel(Q5aar[,reduced_items], cor = "poly")
```




```
## Parallel analysis suggests that the number of factors =  3  and the number of components =  2
```

OK, now we only need 3 or maybe only 2 factors. Lets first try with 2.

```
fa_ps = fa(Q5aar[,reduced_items],
           nfactors = 2,
           cor = "poly")
fa_ps
```

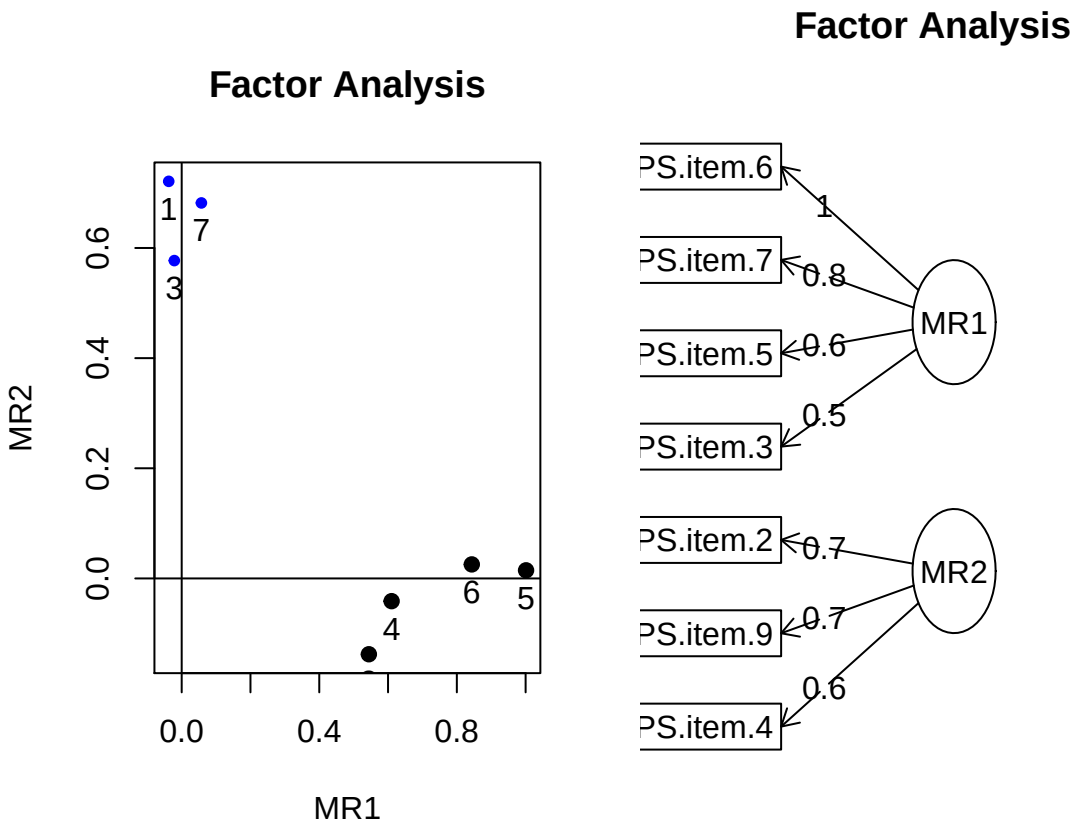
```
## Factor Analysis using method =  minres
## Call: fa(r = Q5aar[, reduced_items], nfactors = 2, cor = "poly")
##
##  Warning: A Heywood case was detected.
## Standardized loadings (pattern matrix) based upon correlation matrix
##              MR1   MR2   h2     u2 com
## PS.item.2 -0.04  0.72 0.53 0.4701 1.0
## PS.item.3  0.54 -0.14 0.34 0.6609 1.1
## PS.item.4 -0.02  0.58 0.34 0.6625 1.0
## PS.item.5  0.61 -0.04 0.38 0.6177 1.0
## PS.item.6  1.00  0.01 1.00 0.0019 1.0
## PS.item.7  0.84  0.03 0.71 0.2948 1.0
## PS.item.9  0.06  0.68 0.46 0.5444 1.0
##
##                        MR1  MR2
## SS loadings           2.40 1.35
## Proportion Var        0.34 0.19
## Cumulative Var        0.34 0.54
```

```
## Proportion Explained  0.64 0.36
## Cumulative Proportion 0.64 1.00
##
##  With factor correlations of
##       MR1   MR2
## MR1  1.00 -0.16
## MR2 -0.16  1.00
##
## Mean item complexity =  1
## Test of the hypothesis that 2 factors are sufficient.
##
## The degrees of freedom for the null model are  21  and the objective function was  2.75 with Chi Squa
## The degrees of freedom for the model are 8  and the objective function was  0.04
##
## The root mean square of the residuals (RMSR) is  0.02
## The df corrected root mean square of the residuals is  0.03
##
## The harmonic number of observations is  26306 with the empirical chi square  312.65  with prob <  8.3
## The total number of observations was  26306  with Likelihood Chi Square =  1103.76  with prob <  5.9e
##
## Tucker Lewis Index of factoring reliability =  0.96
## RMSEA index =  0.072  and the 90 % confidence intervals are  0.069 0.076
## BIC =  1022.34
## Fit based upon off diagonal values = 1
```

RMSEA and even more the TLI suggest that 2 factors should be OK.

This are the new factor loadings:

```
par(mfrow = c(1,2))
plot(fa_ps)
fa.diagram(fa_ps)
```




This looks like a clear factor structure. Lets look at the items to understand what those factor might measure.

```
factor_loadings = loadings(fa_ps)
simple_loadings = apply(factor_loadings,
                        1,
                        function(x)
                          (abs(x) == max(abs(x))))
item_quest_reduced = item_quest[-c(1,8)]

items_factor1 = which(simple_loadings[1,] == T)
items_factor2 = which(simple_loadings[2,] == T)

cat(paste0("Factor1:\n",
           paste(item_quest_reduced[items_factor1],
                 collapse = "\n"),
           "\n"))
```

```
## Factor1:
##  Du har en hyggelig samtale med barnet ditt
##  Du spør barnet ditt om hvordan hans/hennes dag i barnehagen har vært
##  Du sier noe pent til barnet ditt når han/hun har gjort noe bra
##  Du roser barnet ditt om han/hun oppfører seg pent
```

```
cat(paste0("Factor2:\n",
           paste(item_quest_reduced[items_factor1],
                 collapse = "\n"),
           "\n"))
```

```
## Factor2:
##  Du har en hyggelig samtale med barnet ditt
##  Du spør barnet ditt om hvordan hans/hennes dag i barnehagen har vært
##  Du sier noe pent til barnet ditt når han/hun har gjort noe bra
##  Du roser barnet ditt om han/hun oppfører seg pent
```

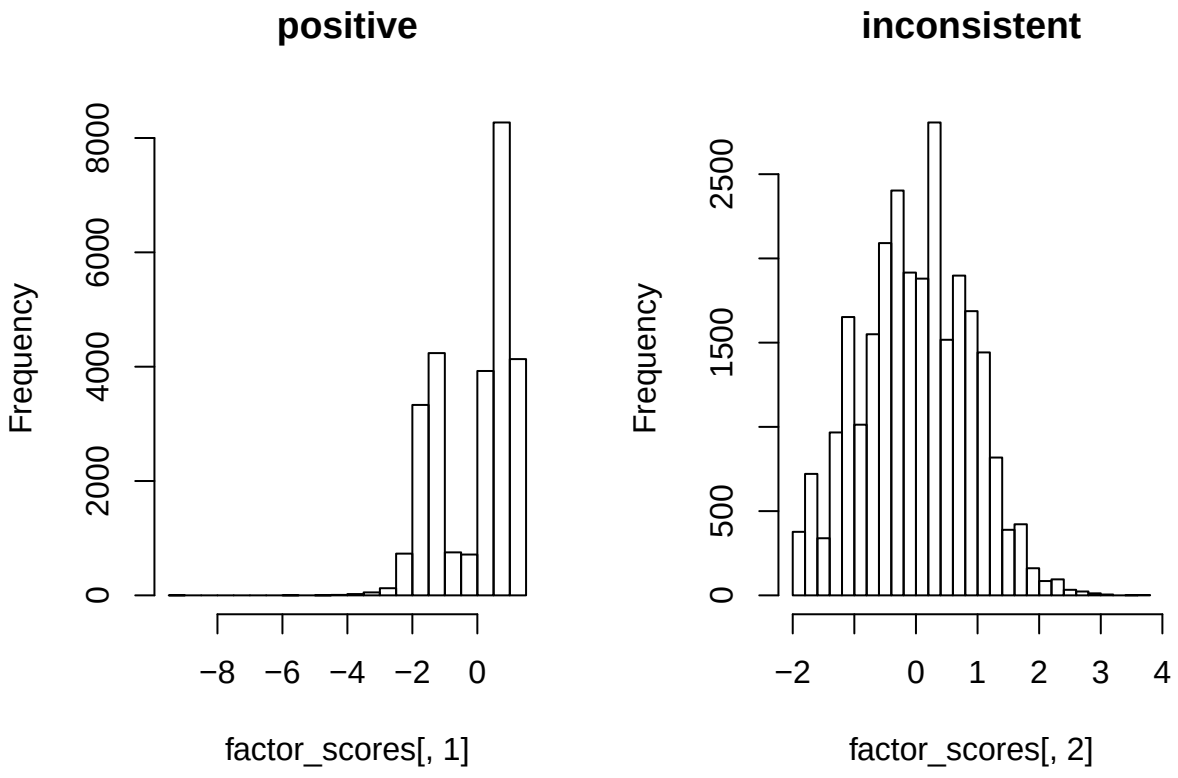
This looks as if there is one factor around positive reinforcement and communication with the child. Lets call this *positive parenting*. The other factor appears to be about the inability to follow through with threats of punishment, lets call this *inconsistent parenting* ;-). If we scroll back and look at correlation of the two factors, it looks as if *positive parenting* has a weak negative correlation with *inconsistent parenting*.

Now, if we want to use the parenting dimensions as predictors, we need to extract the factor scores. Factor scores are likely stored in the `fa_ps` object, which has the results of out exploratory factor analysis. If we want to know what this object contains, we can simply click on in in the "Environment" tab at the right hand side of the RStudio IDE.

```
factor_scores = fa_ps$scores
colnames(factor_scores) = c("positive","inconsistent")
```

Lets quickly look at the factor scores, before we add them to the data.frame with the 5 year data.:

```
par(mfrow = c(1,2))
hist(factor_scores[,1],
     breaks = 25,
     main = "positive")
hist(factor_scores[,2],
     breaks = 25,
     main = "inconsistent")
```



```
Q5aar = cbind(Q5aar,factor_scores)
```

If we had a great parenting scale, the histograms should look normally distributed. Especially the histogram for positive parenting deviates from that. It shows that (a) there are a few extreme outliers to the left and (b) there seems to be something like a ceiling effect. That is, a good number of mothers report to be **VERY** positive in their parenting. This is another instance where it would be great to have a social desirability scale in MoBa.

## Some more data wrangling, including calculation of sum scores

Anyhow, lets continue and put together outcome data. For this, we load the MoBa 8 years data file.

```
data_directory = "data/"
Q8aar = read_sav(paste0(data_directory,
                        "PDB439_Skjema8aar_v10.sav"))
Q8aar = Q8aar[,c("PREG_ID_439","BARN_NR",
                 paste("NN",c(68:80,         # SMFQ
                              211:226,       # CCC-2 sort
                              227:233, 374,  # Språåk 20
                              111:118,       # CD
                              119:136,       # ADHD
                              137:144),      # OD
                        sep = ""))]

Q8aar = data.frame(Q8aar)
```

Now we are coming to a part, which always takes more time than we would wish: cleaning the data. We want to calculate sum-scores, but this is complicated by two facts: MoBa data come with

- `0` instead of `NA` for data with unambiguous responses
- all scales start at 1
- some of the data are missing.

We will address these problems in that order, starting with renaming some variables. To make this easier, we'll create simple functions where this is useful.

First we rename all variables:

```
rename_variables = function(df,old_names,new_names) {
  names(df)[names(df) %in% old_names] = new_names
  return(df)
}

Q8aar = rename_variables(Q8aar,
                         paste0("NN",68:80),
                         paste0("SMFQ.i",1:13,"_8y"))
Q8aar = rename_variables(Q8aar,
                         paste0("NN",c(227:233,374)),
                         paste0("SPRAAK20.i",1:8,"_8y"))
Q8aar = rename_variables(Q8aar,
                         paste0("NN",211:226),
                         paste0("CCCs.i",1:16,"_8y"))
for (item in paste0("CCCs.i",c(10:16),"_8y"))
  Q8aar[,item] = abs(Q8aar[,item]-5)
Q8aar = rename_variables(Q8aar,
                         paste0("NN",111:118),
```

```
                          paste0("CD.i",1:8,"_8y"))
Q8aar = rename_variables(Q8aar,
                          paste0("NN",119:136),
                          paste0("ADHD.i",1:18,"_8y"))
Q8aar = rename_variables(Q8aar,
                          paste0("NN",137:144),
                          paste0("OD.i",1:8,"_8y"))
Q8aar = rename_variables(Q8aar,
                          paste0("NN",145:149),
                          paste0("SCARED.i",1:5,"_8y"))
```

Next we replace all 0 values with NA and subtract 1 from all items. We use the `grep` command, which makes it easy for us to find all variables that match a particular pattern. We also temporarily create a new `data.frame` tmp_items, in which we keep the item data.

```
all_items = grep("\\.i[0-9]",names(Q8aar), value = T)
tmp_items = Q8aar[,all_items]
tmp_items[tmp_items == 0] = NA
tmp_items = tmp_items-1
```

We can use imputation to replace missing data. I prefer to do this only for cases, for which we have at least 50% of the data. So we remove participants with more than 50% missing data.

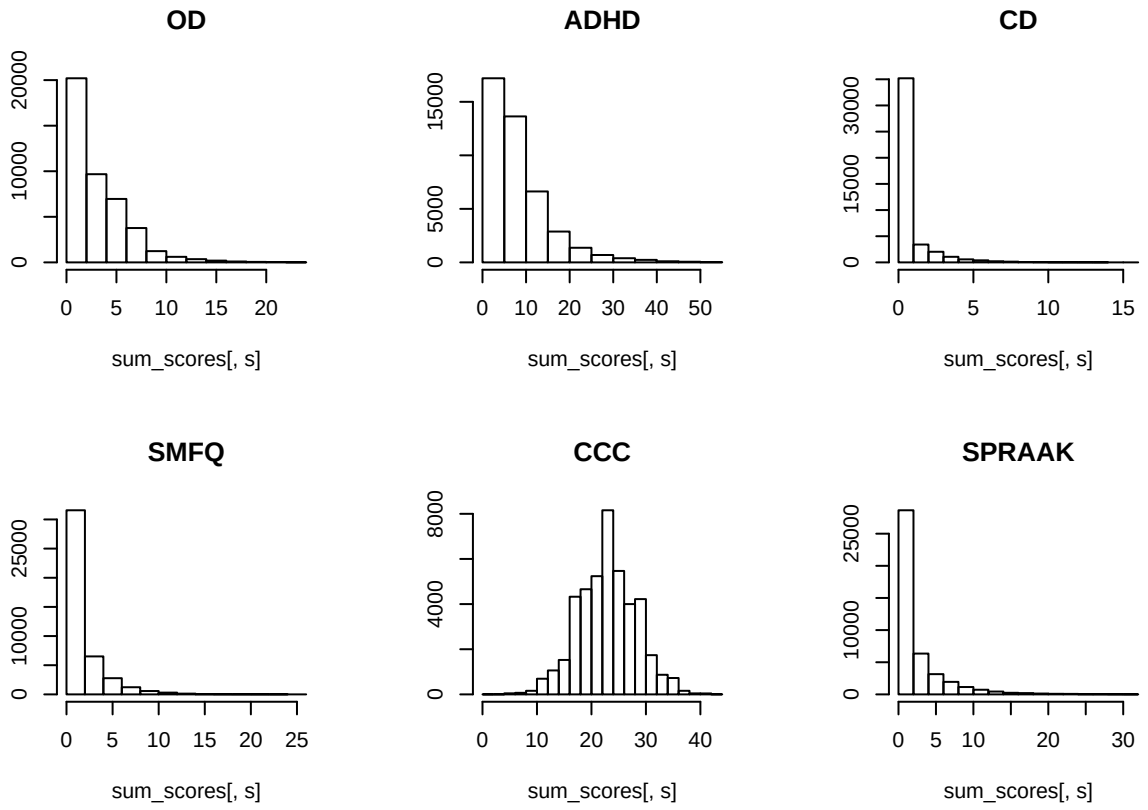
```
proportion_missing = rowMeans(is.na(tmp_items))
Q8aar = Q8aar[proportion_missing < .5,]
tmp_items = tmp_items[proportion_missing < .5,]
```

Now we use the function `hotdeck`from the package `VIM` to impute values. (`hotdeck` is fast). Other approaches like nearest neighbor imputation of multiple chained imputation are more accurate, by they are to slow when analyzing bigger data-sets in a tutorial.

```
tmp_items_imputed = hotdeck(as.matrix(tmp_items))
tmp_items_imputed = tmp_items_imputed[,1:ncol(tmp_items)]
```

Finally, we can calculate sum scores.

```
scales = c("OD","ADHD","CD","SMFQ","CCC","SPRAAK")
sum_scores = matrix(NA,
                    nrow = nrow(Q8aar),
                    ncol = length(scales))
colnames(sum_scores) = scales
par(mfrow = c(2,3))
for (s in scales) {
  items = grep(s,names(tmp_items_imputed), value = T)
  sum_scores[,s] = rowSums(tmp_items_imputed[,items])
  hist(sum_scores[,s],
       main = s,
       ylab = "")
}
```




```
head(sum_scores)
```

```
##      OD ADHD CD SMFQ CCC SPRAAK
## [1,]  2    5  0    1  18      2
## [2,]  1   11  0    2  28      0
## [3,]  2    8  0    1  27      2
## [4,]  0    0  0    1  23      0
## [5,] 12   16  2    5  20     14
## [6,]  6    9  0    0  23      1
```

This looks more or less as expected. Now we put the Q8aar data together, and merge 5 and 8 years data.

```
Q8aar = cbind(Q8aar, sum_scores)
my_data = merge(Q5aar[,c("PREG_ID_439","BARN_NR",colnames(factor_scores))],
                Q8aar[,c("PREG_ID_439","BARN_NR",scales)],
                by = c("PREG_ID_439","BARN_NR"))
head(my_data)
```

```
##   PREG_ID_439 BARN_NR   positive inconsistent OD ADHD CD SMFQ CCC SPRAAK
## 1      100002       1 -1.0746229   -0.8082697  3    4  0    1  25      4
## 2      100003       1  0.7753005   -0.8384897  4    3  0    1  29      0
## 3      100004       1 -1.5912295   -0.3571006  2    5  0    1  24      1
## 4      100006       1  0.8029876   -0.6750237  3    6  0    1  19      0
## 5      100007       1  1.2734772   -0.7874821  1    0  1    0  25      0
## 6      100009       1 -1.5745820    0.2151237  3    8  1    3  19      7
```

```
my_data$positive = as.numeric(scale(my_data$positive))
my_data$inconsistent = as.numeric(scale(my_data$inconsistent))
```

```
my_data = my_data[my_data$positive > -4,]
```

## Linear (and other) regressions in R

*For the next section, you can run your own regression by changing the outcome. Use the command* `names(my_data)` *to see, which variables are available.*

Finally, we can do simple linear regression models:

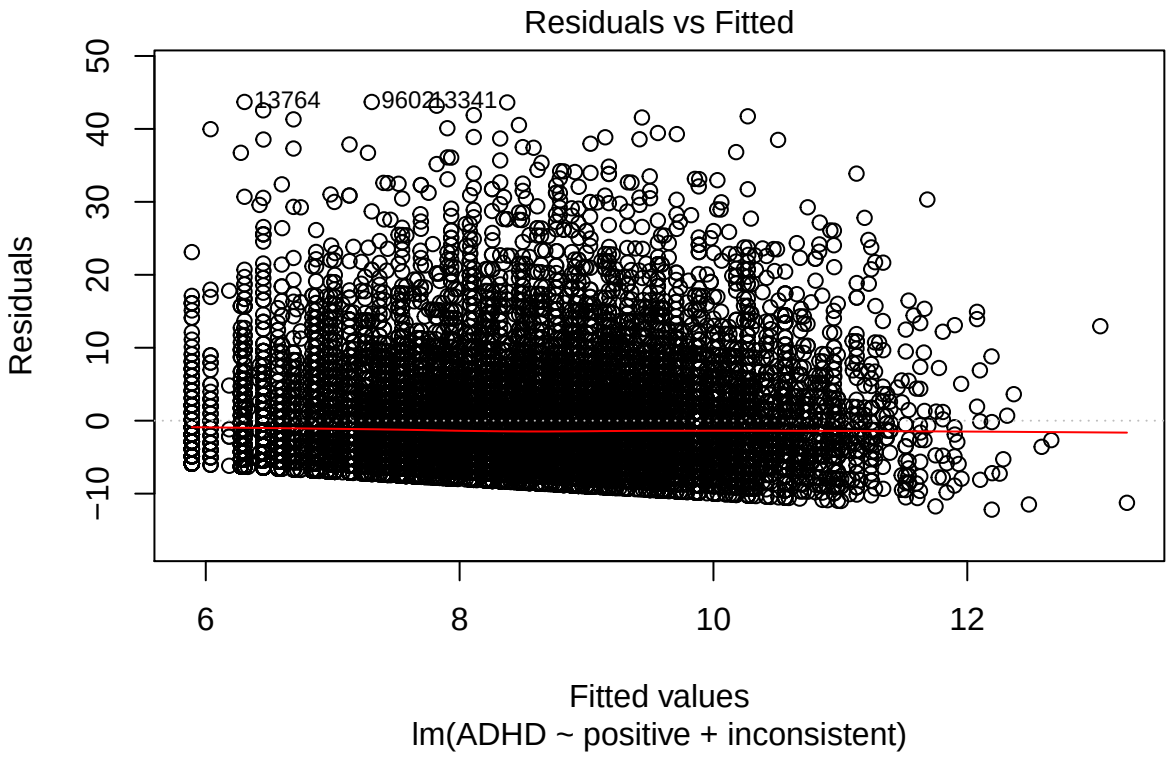
```
ADHD_model = lm(ADHD ~ positive + inconsistent, my_data)
summary(ADHD_model)
```

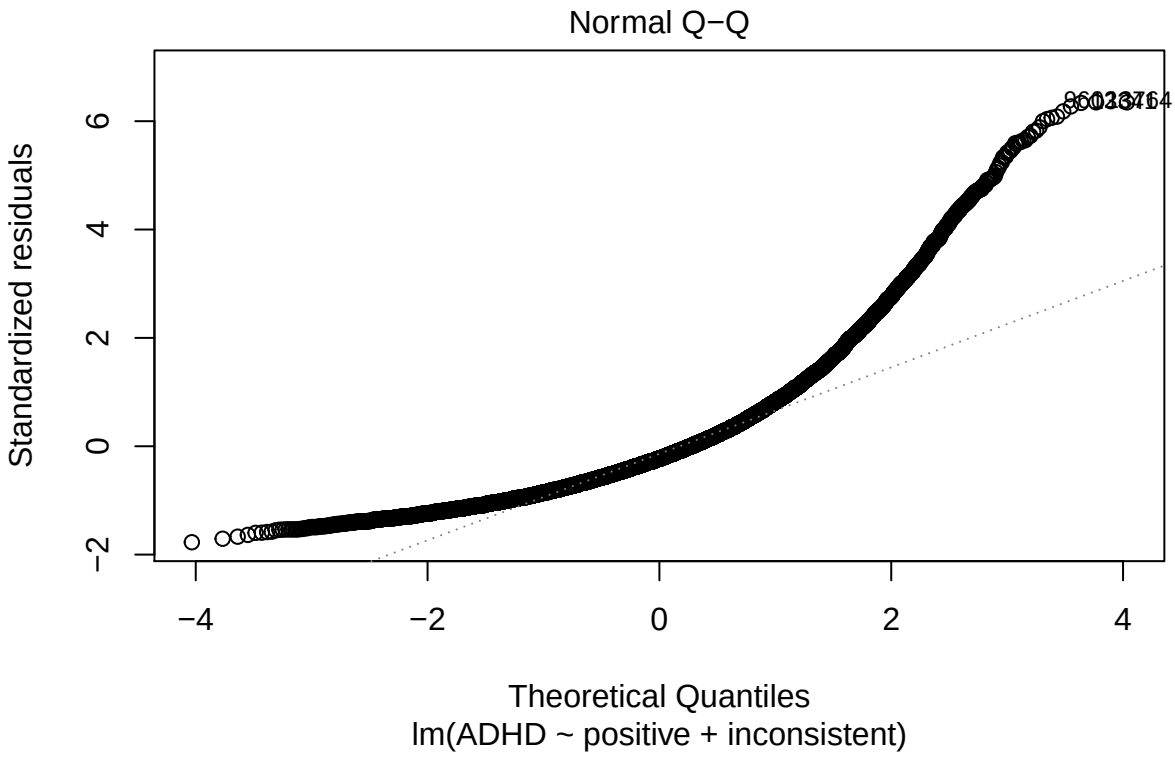
```
##
## Call:
## lm(formula = ADHD ~ positive + inconsistent, data = my_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.193  -4.665  -1.562   2.744  43.695
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.48534    0.05103 166.284  < 2e-16 ***
## positive     -0.39186    0.05273  -7.431 1.13e-13 ***
## inconsistent  1.01118    0.05246  19.276  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.885 on 18199 degrees of freedom
## Multiple R-squared:  0.02782,    Adjusted R-squared:  0.02771
## F-statistic: 260.4 on 2 and 18199 DF,  p-value: < 2.2e-16
```

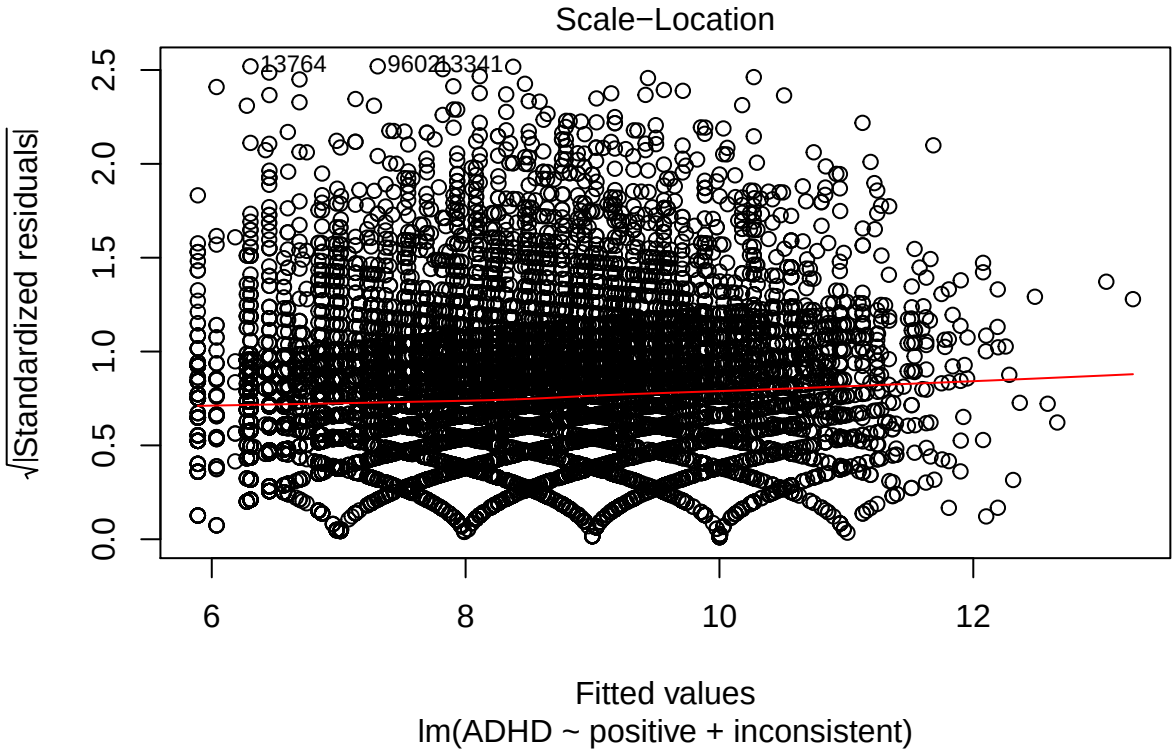
This is exciting! Parenting style at age 5 predicts ADHD symptoms at age 8!

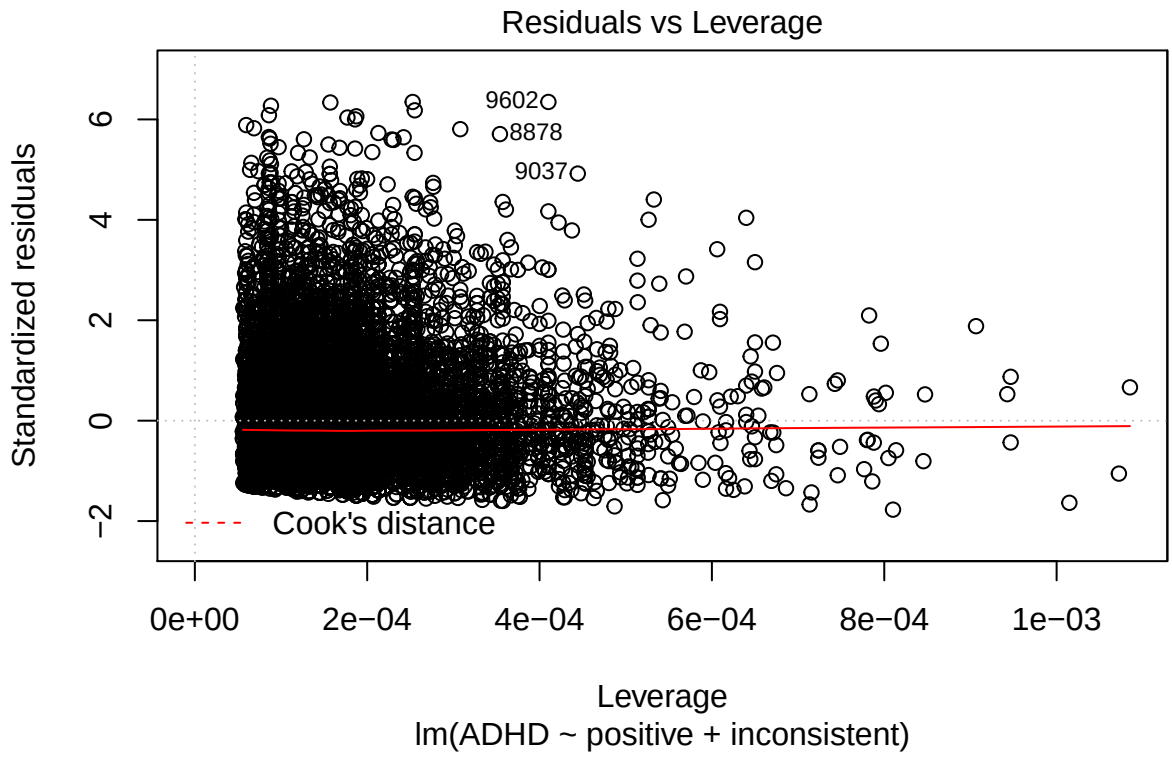
Before getting carried away, it makes sense to look at a few diagnostic plots:

```
par(ask=F)
plot(ADHD_model)
```


Residuals vs Fitted
Residuals
50
40
30
20
10
0
−10
13764
960213341
6
8
10
12
Fitted values
lm(ADHD ~ positive + inconsistent)


Normal Q−Q
Standardized residuals
−2
0
2
4
6
−4
−2
0
2
4
Theoretical Quantiles
lm(ADHD ~ positive + inconsistent)


Scale−Location
13764
9602133410
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
6
8
10
12
Fitted values
lm(ADHD ~ positive + inconsistent)




Here we can go through an explanation of the diagnostic plots.

As expected, this shows that a simple linear regression is probably not appropriate here.

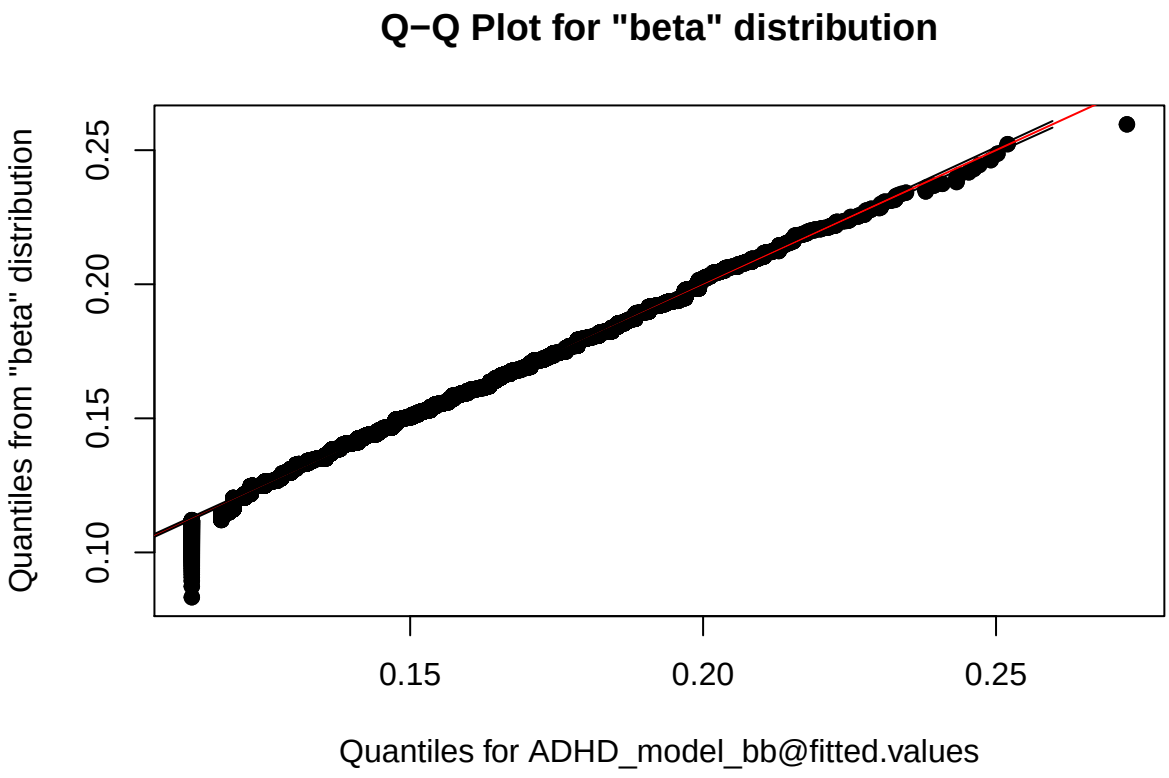
### A beta-binomial regression

A beta-binomial regression would be more appropriate here, because the outcome variable can only be between 0 and an upper bound. There are several possibilities to do such regressions in R. On mature package that can do this is `VGAM`. Alternatives include `brms` and `rstanarm`.

```
library(VGAM)
library(qualityTools)
ADHD_model_bb = vglm(cbind(ADHD, 54-ADHD) ~ poly(positive,2) + poly(inconsistent,2),
                         betabinomial,
                         data = my_data,
                         trace = TRUE)
```

```
## VGLM    linear loop  1 :  loglikelihood = -56963.9582
## VGLM    linear loop  2 :  loglikelihood = -56956.2402
## VGLM    linear loop  3 :  loglikelihood = -56956.2376
## VGLM    linear loop  4 :  loglikelihood = -56956.2376
```

```
qqPlot(ADHD_model_bb@fitted.values,"beta", start = list(shape1 = 10, shape2 = 2))
```




```
summary(ADHD_model_bb)
```

```
##
## Call:
## vglm(formula = cbind(ADHD, 54 - ADHD) ~ poly(positive, 2) + poly(inconsistent,
##     2), family = betabinomial, data = my_data, trace = TRUE)
##
##
## Pearson residuals:
##                Min      1Q   Median     3Q    Max
## logit(mu)  -2.6744 -0.6208 -0.05143 0.5334  6.634
## logit(rho) -0.8089 -0.7127 -0.47817 0.1730 20.304
##
## Coefficients:
##                          Estimate Std. Error  z value Pr(>|z|)
## (Intercept):1          -1.675039   0.006806 -246.109  < 2e-16 ***
## (Intercept):2          -2.254103   0.013106 -171.992  < 2e-16 ***
## poly(positive, 2)1     -7.962574   0.876562   -9.084  < 2e-16 ***
## poly(positive, 2)2     -3.920274   0.862231   -4.547 5.45e-06 ***
## poly(inconsistent, 2)1 18.567372   0.885754   20.962  < 2e-16 ***
## poly(inconsistent, 2)2  0.513920   0.846108    0.607    0.544
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of linear predictors:  2
##
```

```
## Names of linear predictors: logit(mu), logit(rho)
##
## Log-likelihood: -56956.24 on 36398 degrees of freedom
##
## Number of iterations: 4
##
## No Hauck-Donner effect found in any of the estimates
```

Even though the QQ plot shows that beta-binomial model is more appropriate, the tutorial continues with the result of the regression model. This is because some of the topics we are going to cover are easy to implement for a linear regression model, but take a bit more time for alternative models.

### Processing regression results: Tables and plots

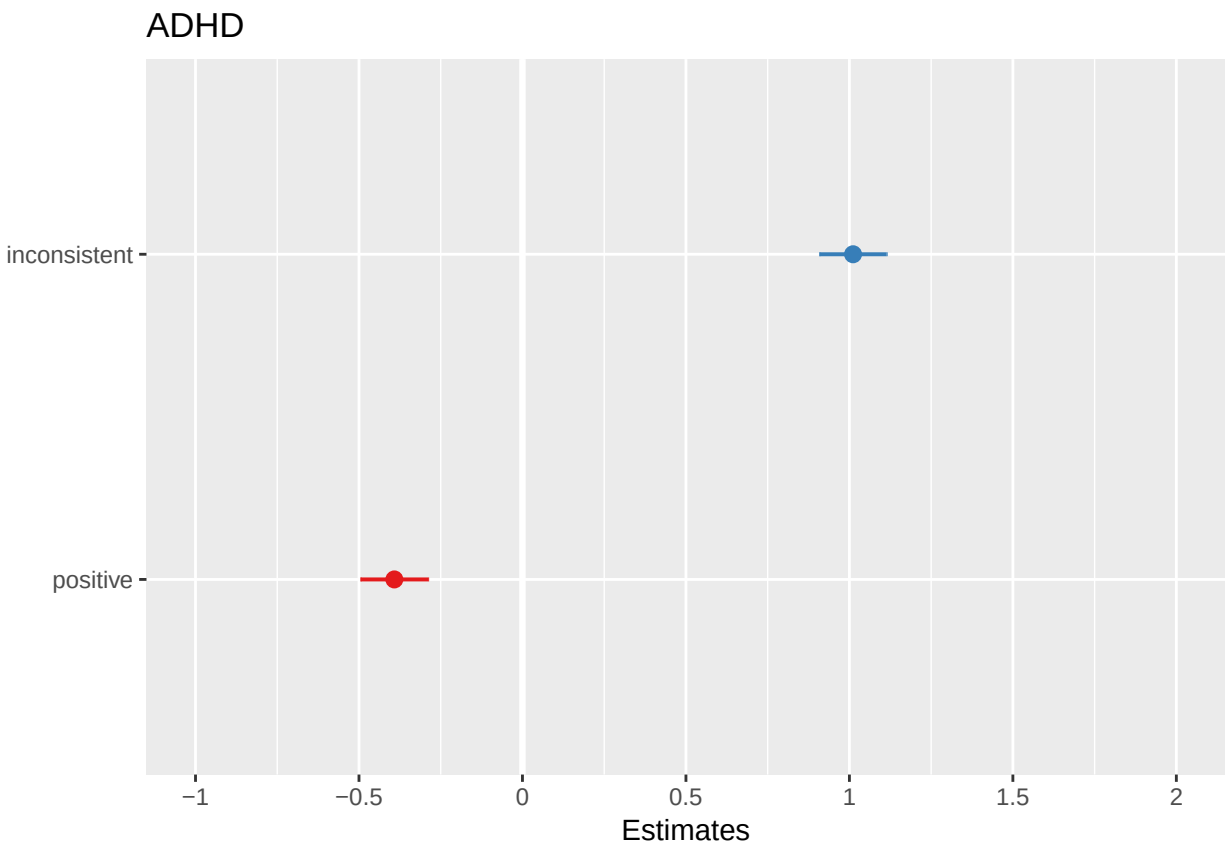
With the package `apaTables`we can format results tables from regressions as described in the APA Manual 6.

```
library(apaTables)
apa.reg.table(ADHD_model, filename = "Table2_APA.doc", table.number = 2)
```

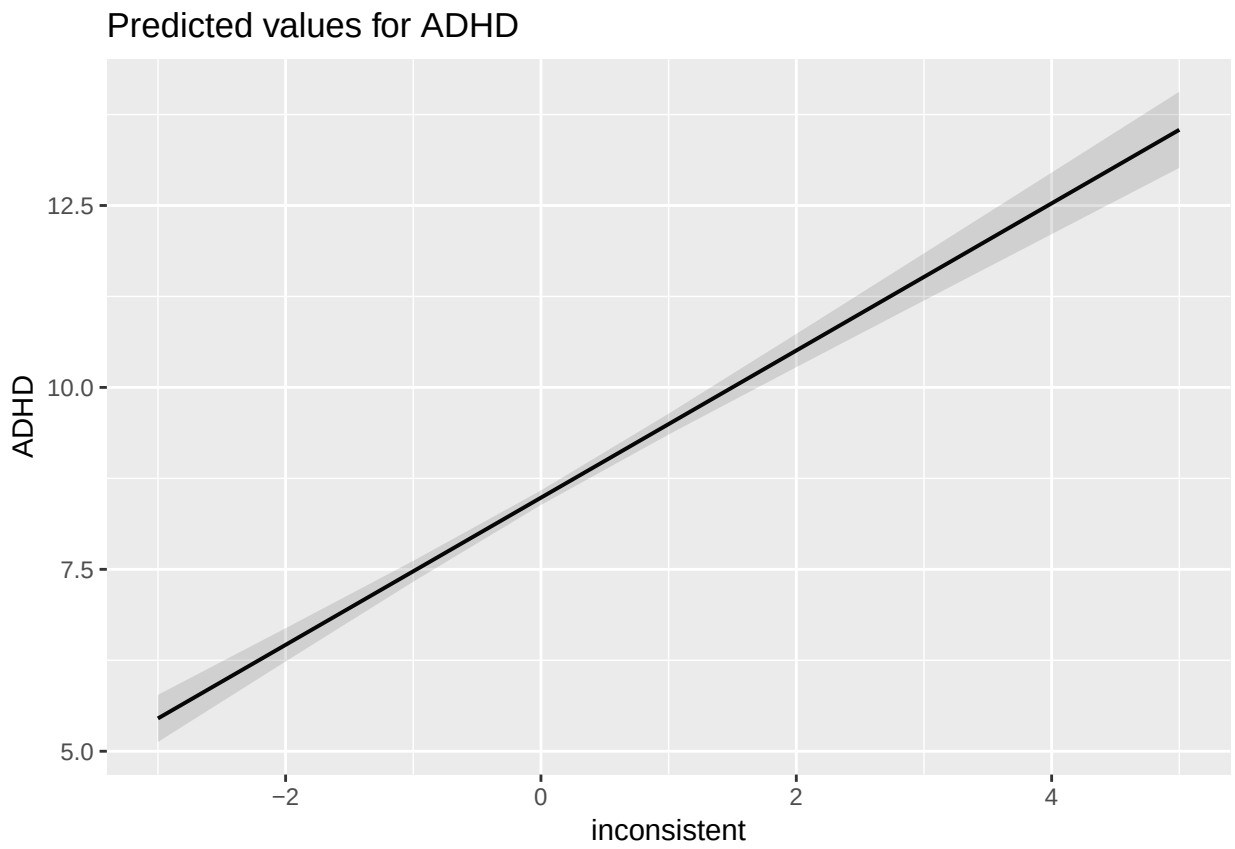
```
##
##
## Table 2
##
## Regression results using ADHD as the criterion
##
##
##     Predictor       b       b_95%_CI  beta    beta_95%_CI sr2 sr2_95%_CI
##   (Intercept)  8.49**   [8.39, 8.59]
##      positive -0.39** [-0.50, -0.29] -0.06 [-0.07, -0.04] .00 [.00, .00]
##  inconsistent  1.01**   [0.91, 1.11]  0.14   [0.13, 0.16] .02 [.02, .02]
##
##
##
##       r             Fit
##
##  -.09**
##   .16**
##             R2 = .028**
##         95% CI[.02,.03]
##
##
## Note. A significant b-weight indicates the beta-weight and semi-partial correlation are also signifi
## b represents unstandardized regression weights. beta indicates the standardized regression weights.
## sr2 represents the semi-partial correlation squared. r represents the zero-order correlation.
## Square brackets are used to enclose the lower and upper limits of a confidence interval.
## * indicates p < .05. ** indicates p < .01.
##
```

It is also relatively easy to plot model results. First, we simply plot the regression coefficients and confidence intervals.

```
library(sjPlot)
library(ggeffects)
library(ggplot2)
plot_model(ADHD_model)
```

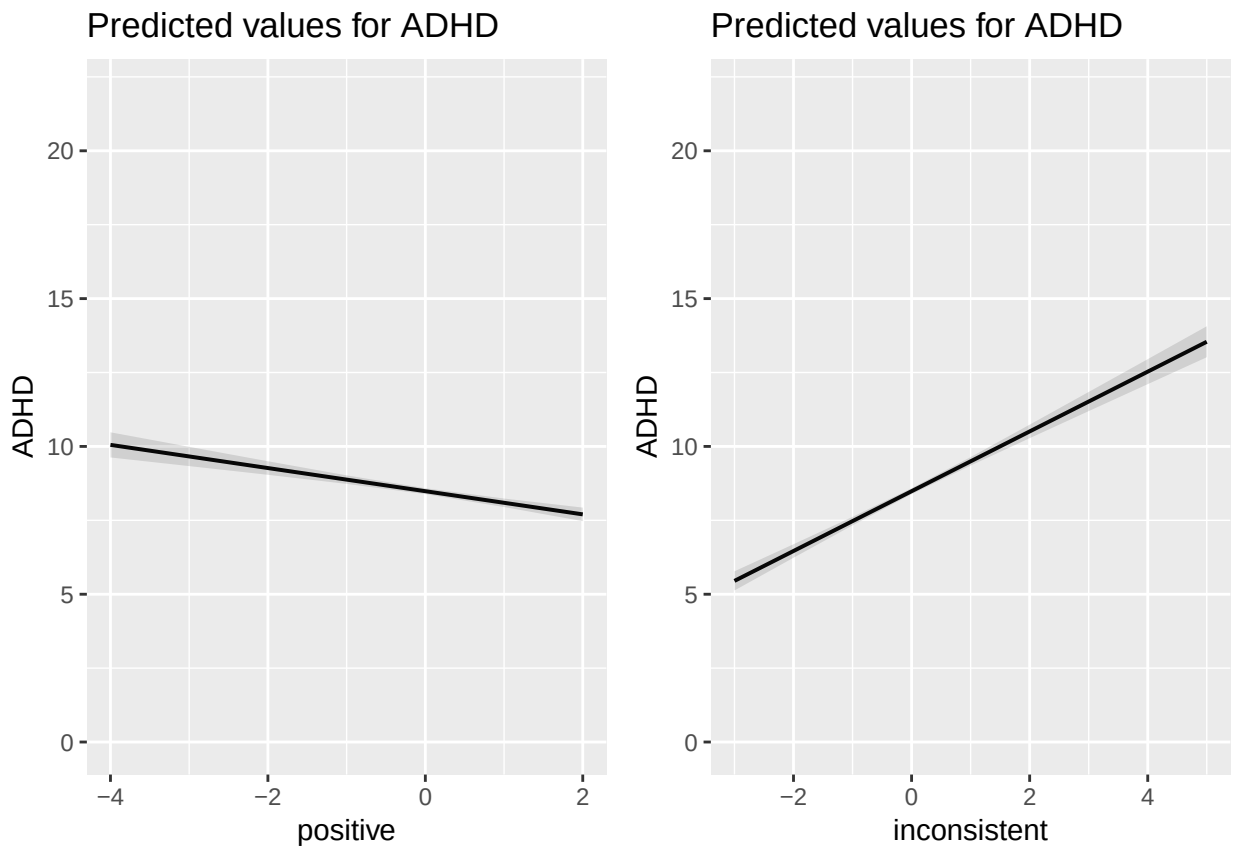



```
plot_model(ADHD_model, type = "eff", terms = "inconsistent")
```




We can also plot the effects of the two predictors besides each other to compare effects:

```
library(ggplot2)
ylim = c(0,quantile(my_data$ADHD,c(.95)))

effect_pos = plot_model(ADHD_model, type = "eff", terms = "positive") +
             coord_cartesian(ylim = ylim)
effect_inc = plot_model(ADHD_model, type = "eff", terms = "inconsistent") +
             coord_cartesian(ylim = ylim)

cowplot::plot_grid(effect_pos,
                   effect_inc)
```




### Adding and plotting non-linear effects

This shows a linear effect of parenting style. Is it possible that there are non-linear effects? We can easily test this by adding squared terms to the model.

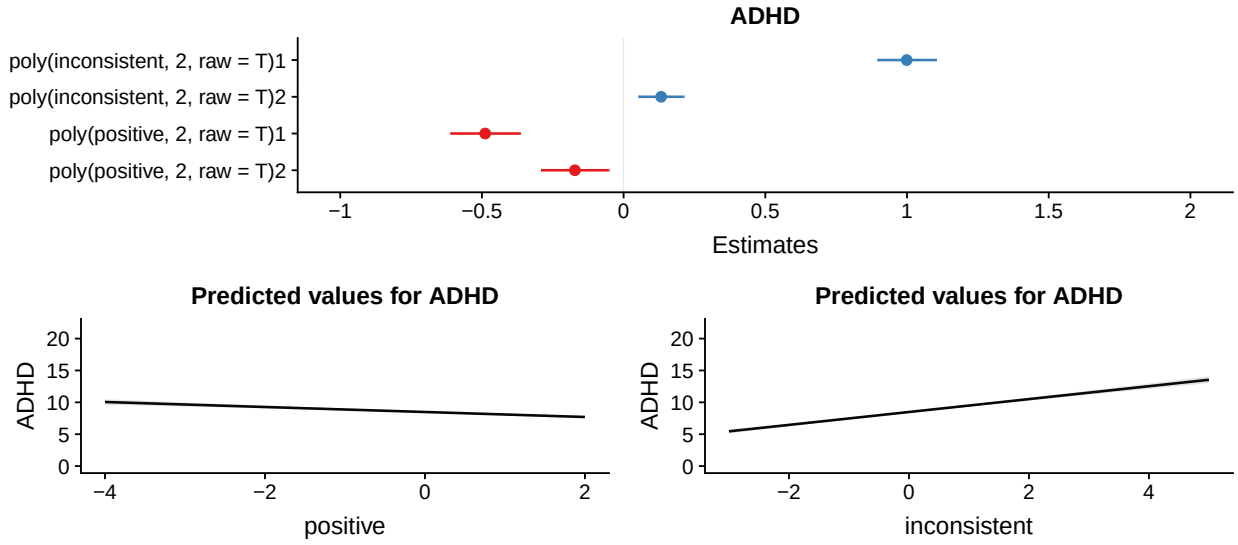
```
ADHD_model2 = lm(ADHD ~ poly(positive,2,raw = T) + poly(inconsistent,2, raw = T), my_data)
summary(ADHD_model2)
```

```
##
## Call:
## lm(formula = ADHD ~ poly(positive, 2, raw = T) + poly(inconsistent,
##     2, raw = T), data = my_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.938  -4.668  -1.590   2.763  43.936
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      8.52312    0.08707  97.892  < 2e-16 ***
## poly(positive, 2, raw = T)1     -0.48830    0.06244  -7.820 5.57e-15 ***
## poly(positive, 2, raw = T)2     -0.17183    0.06078  -2.827  0.00470 **
## poly(inconsistent, 2, raw = T)1  0.99904    0.05253  19.019  < 2e-16 ***
## poly(inconsistent, 2, raw = T)2  0.13246    0.04066   3.258  0.00113 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 6.882 on 18197 degrees of freedom
## Multiple R-squared:  0.02876,    Adjusted R-squared:  0.02854
## F-statistic: 134.7 on 4 and 18197 DF,  p-value: < 2.2e-16
```

We'll use a new library, `cowplot` to put multiple ggplot figures together.

```
library(cowplot)
coef_plot = plot_model(ADHD_model2)
effect_pos = plot_model(ADHD_model, type = "eff", terms = "positive") +
             coord_cartesian(ylim = ylim)
effect_inc = plot_model(ADHD_model, type = "eff", terms = "inconsistent") +
             coord_cartesian(ylim = ylim)

bottom_plots = plot_grid(effect_pos,
                         effect_inc)
plot_grid(coef_plot,
          bottom_plots,
          nrow = 2)
```




### Adding interaction effects

Finally, we can also add an interaction effect:

```
ADHD_model2 = lm(ADHD ~ poly(positive,2,raw = T) +
                        poly(inconsistent,2, raw = T) +
                        positive:inconsistent,
                 my_data)
summary(ADHD_model2)
```

```
##
## Call:
## lm(formula = ADHD ~ poly(positive, 2, raw = T) + poly(inconsistent,
##     2, raw = T) + positive:inconsistent, data = my_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -13.985  -4.685  -1.595   2.762  43.922
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      8.51708    0.08854  96.192  < 2e-16 ***
## poly(positive, 2, raw = T)1     -0.48684    0.06256  -7.781 7.55e-15 ***
## poly(positive, 2, raw = T)2     -0.16563    0.06299  -2.629  0.00856 **
## poly(inconsistent, 2, raw = T)1  0.99854    0.05255  19.003  < 2e-16 ***
## poly(inconsistent, 2, raw = T)2  0.13717    0.04255   3.223  0.00127 **
## positive:inconsistent            0.02093    0.05575   0.375  0.70743
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.882 on 18196 degrees of freedom
## Multiple R-squared:  0.02877,    Adjusted R-squared:  0.0285
## F-statistic: 107.8 on 5 and 18196 DF,  p-value: < 2.2e-16
```

This does not seem to make a big difference.

This analysis suggests that mothers who report inconsistent parenting at child age 3 (they do not follow trough with a threatened punishment) later report more ADHD problems with their children. There are several reasons that we should not immediately endorse a causal interpretation. This could in fact be revers causation, if children with problems at age 8 also had problems at age 5, and it was these problems at age five that caused inconsistent parenting. Some would certainly argue that this is all genetic: the same genes that are responsible for mental health problems also cause inconsistent parenting (e.g. parents with mental health problems might be more prone to be inconsistent.). Genes are an example of confounders: common causes of exposure and outcomes. We do not have genetic information here, but we can easily think of other potential common causes like parental education and age, parity. In addition, we might want to adjust for other covariates like the childs gender.

## Investigating potential confounders through plotting

The next lines of R code load these variables and adds them to our data.

```
data_directory = "data/"
Q1 = read_sav(paste0(data_directory,
                     "PDB439_Skjema1_v10.sav"))
Q1 = data.frame(Q1[,c("PREG_ID_439",
                      paste("AA",c(11,1123:1127,1300:1303),
                            sep = ""))])
names(Q1)[-1] = c("Q1_YEAR","CivilStatus","mEDUcomp","mEDUcurr",
                  "fEDUcomp","fEDUcurr","n_people_19p_Q1",
                  "n_children_12_18_Q1","n_children_6_11_Q1",
                  "n_children_0_5_Q1"))

Q1$CivilStatus = factor(Q1$CivilStatus,
                        levels = 1:6,
                        labels = c("married","separated","cohabitating",
                                   "widow","single","Other"))

EDUlabels = c("basic","1-2 high school","vocational high","general high", "Bachelor","Master" )

Q1$mEDU = Q1$mEDUcomp
idx = is.na(Q1$mEDUcomp) & !is.na(Q1$mEDUcurr) & Q1$mEDUcurr != 1
```

```
Q1$mEDU[idx] = Q1$mEDUcurr[idx] - 1
Q1$mEDU = ordered(Q1$mEDU,labels = EDUlabels)
Q1$fEDU = Q1$fEDUcomp
idx = is.na(Q1$fEDUcomp) & !is.na(Q1$fEDUcurr) & Q1$fEDUcurr != 1
Q1$fEDU[idx] = Q1$fEDUcurr[idx] - 1
Q1$fEDU = ordered(Q1$fEDU,labels = EDUlabels)

rm(EDUlabels,idx)


data_directory = "data/"
MFR = read_sav(paste0(data_directory,
                      "PDB439_MFR_520_v10.sav"))
MFR = data.frame(MFR[,c("PREG_ID_439","BARN_NR",
                      "KJONN","MORS_ALDER",
                      "PARITET_5","FMND")])
names(MFR)[-c(1,2)] = c("gender","mAge",
                      "parity","birthmonth")
MFR$gender = factor(MFR$gender,levels = 1:2,labels = c("boy","girl"))
MFR$birthmonth = as.numeric(MFR$birthmonth)
```

```
my_data = merge(my_data,
                MFR,
                by = c("PREG_ID_439","BARN_NR"),
                all.x = T,
                all.y = F)
my_data = merge(my_data,
                Q1,
                by = c("PREG_ID_439"),
                all.x = T,
                all.y = F)
```

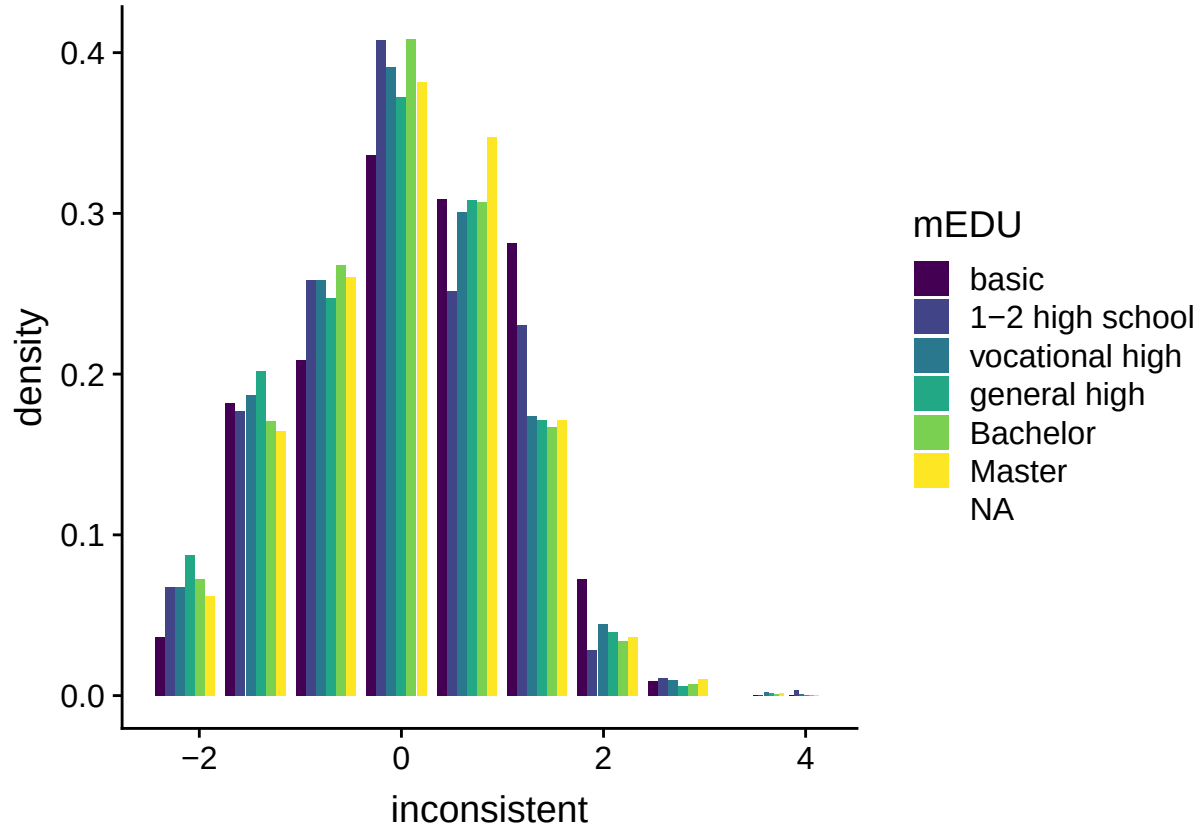
Now we are ready to look at a few tables and plots. First, we use the `group_by`function of the `dplyr`package to show the mean exposures grouped by education.

```
library(dplyr)
options(digits = 2)
my_stats = my_data %>%
  group_by(mEDU) %>%
  summarise(m = mean(inconsistent),
            sd = sd(inconsistent),
            N = n()
            )
my_stats
```

```
## # A tibble: 7 x 4
##   mEDU                    m    sd     N
##   <ord>               <dbl> <dbl> <int>
## 1 basic             0.218   1.06    158
## 2 1-2 high school   0.0238  1.04    405
## 3 vocational high -0.00326 1.03   1512
## 4 general high    -0.0549   1.03   1974
## 5 Bachelor        -0.0212   0.984  7978
## 6 Master           0.0386   0.990  5724
## 7 <NA>             0.0193   1.07    451
```

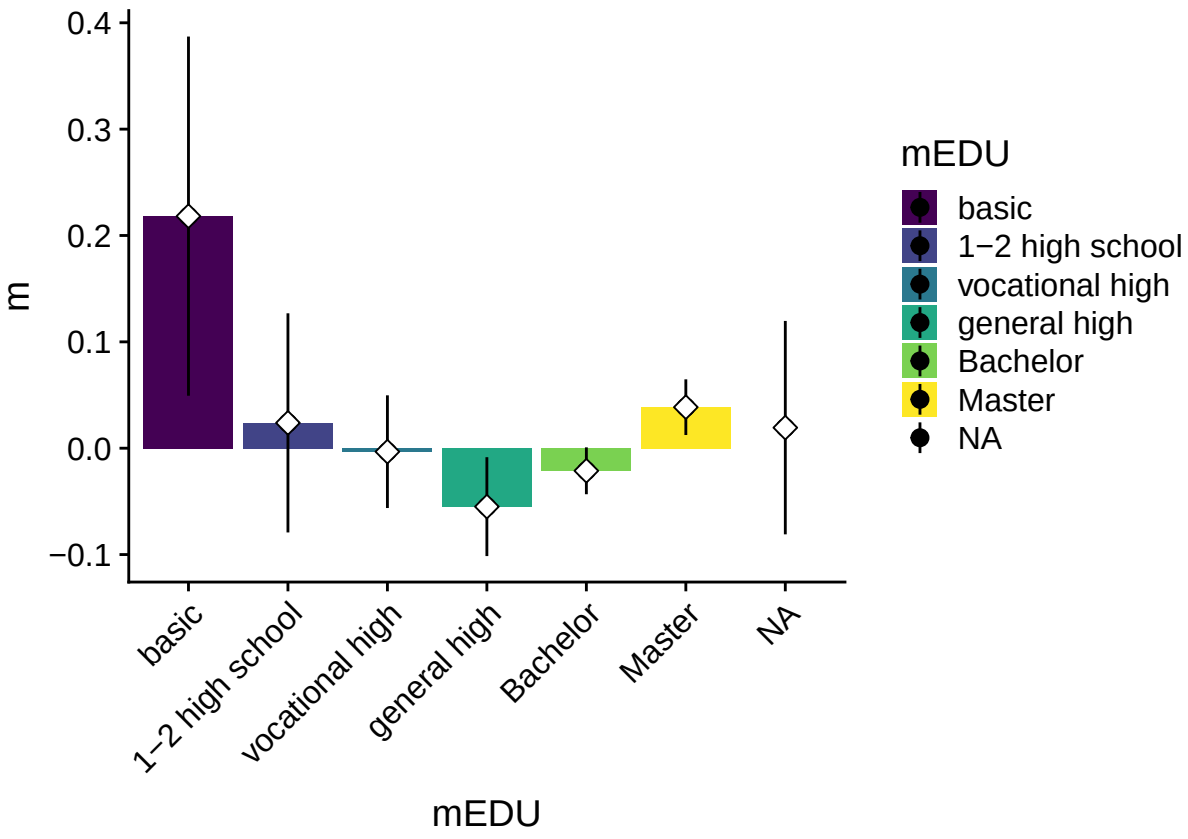
This does not look very clear. We can also plot the raw data using the nice ggplot package. Lets start with a histogram.

```
library(ggplot2)
ggplot(my_data, aes(x = inconsistent, fill = mEDU)) +
  geom_histogram(aes(y = ..density..),position="dodge2", bins = 10)
```



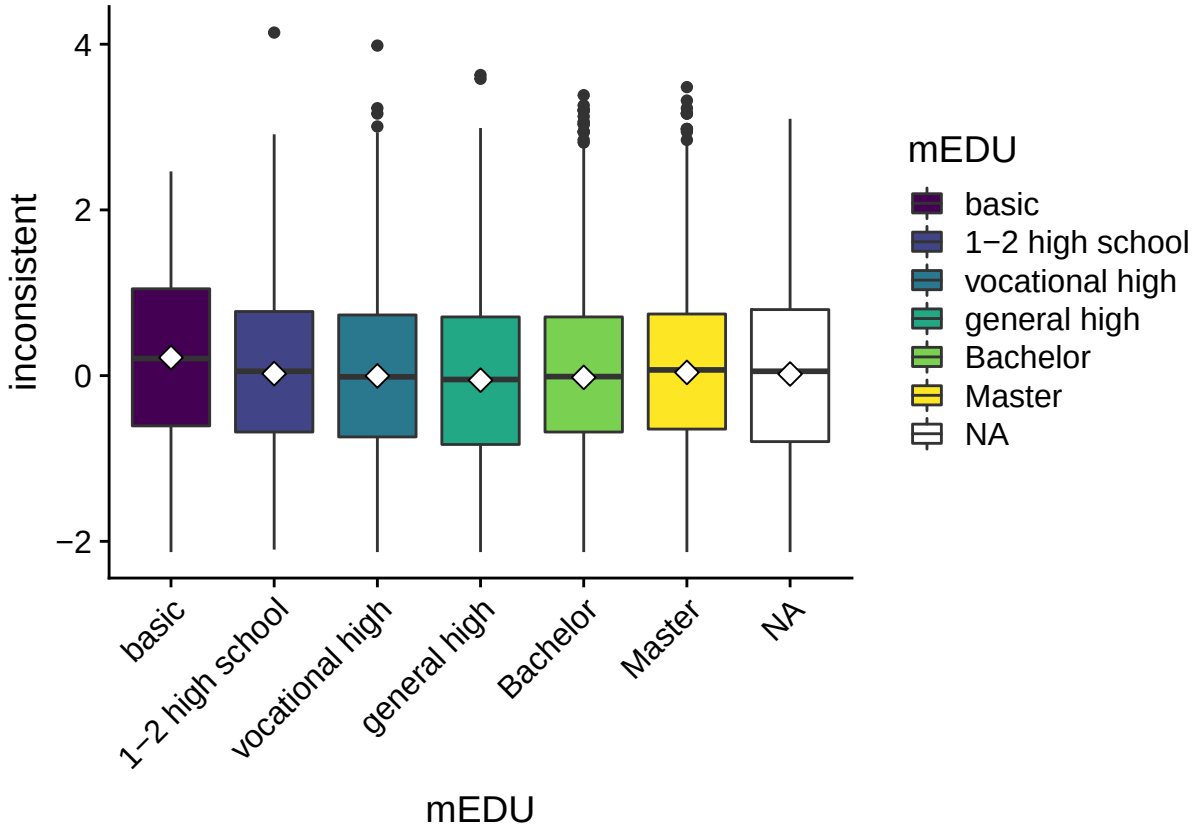

We can simply plot the `my_stats` table we made above, after adding confidence intervals:

```
my_stats$CIlower = my_stats$m - my_stats$sd/sqrt(my_stats$N)*2
my_stats$CIupper = my_stats$m + my_stats$sd/sqrt(my_stats$N)*2

ggplot(my_stats,aes(x = mEDU, y = m, fill = mEDU)) + geom_bar(stat = "identity") +
  geom_pointrange(aes(x = mEDU, ymin = CIlower, ymax = CIupper)) +
  stat_summary(fun.y="median", geom="point", shape=23, size=3, fill="white") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




Now it looks as if the differences between groups are substantial. However, remember that the exposure variables has an sd of 1, so the differences are maybe not so important. To get a better view of the variations within and between groups, we can plot boxplots:

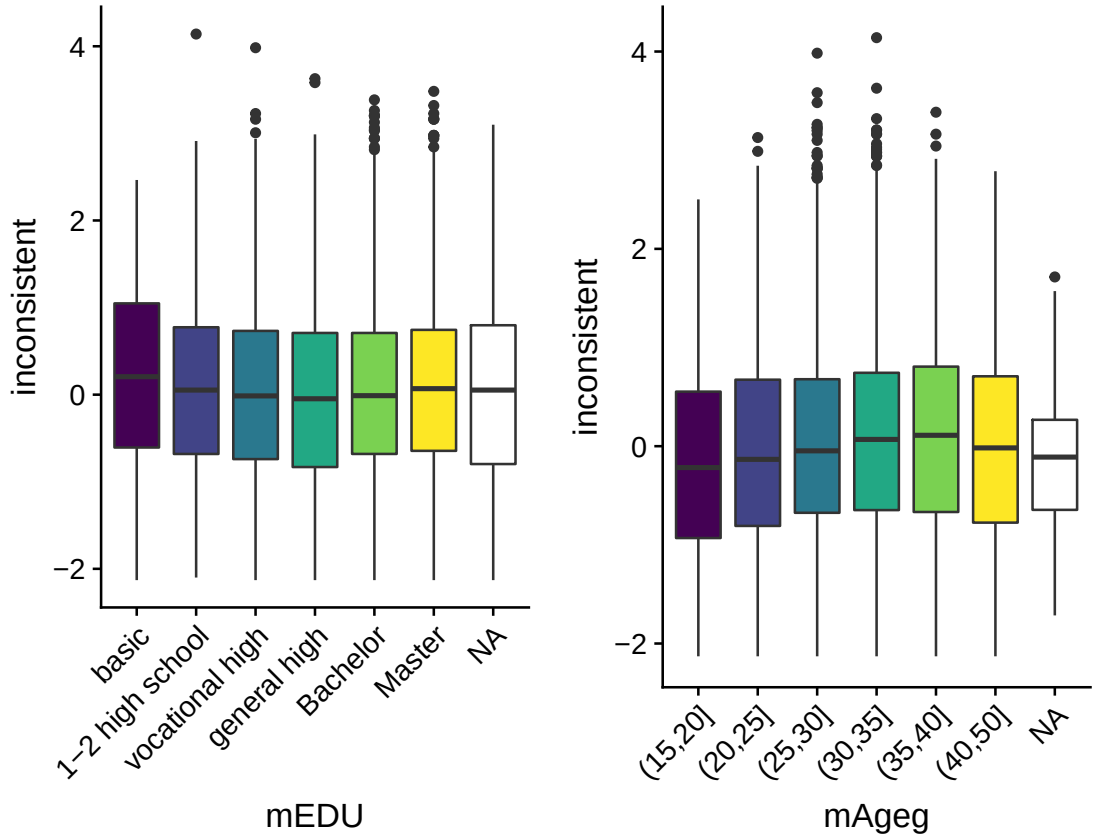
```
ggplot(my_data, aes(x = mEDU, y = inconsistent, fill = mEDU)) +
  geom_boxplot() +
  stat_summary(fun.y="mean", geom="point", shape=23, size=3, fill="white") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



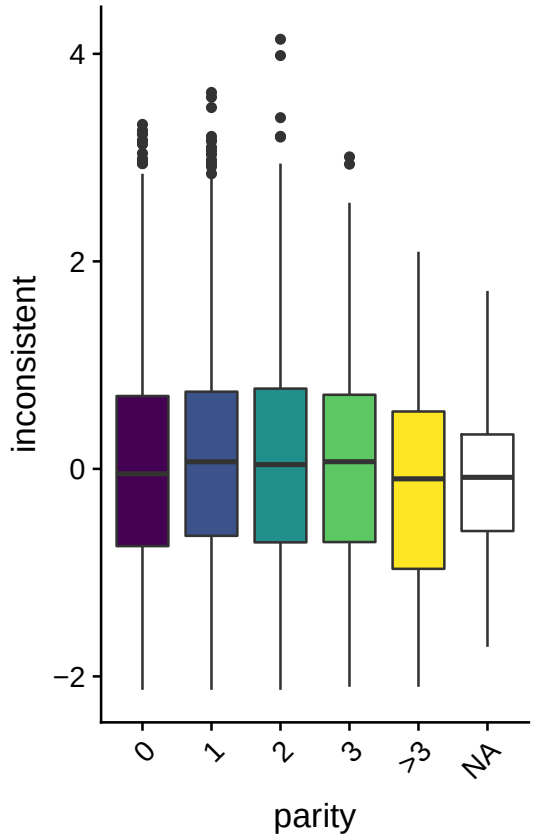

This shows that even though there is are reliable differences between groups, the variation within groups is much larger than the variation between groups.

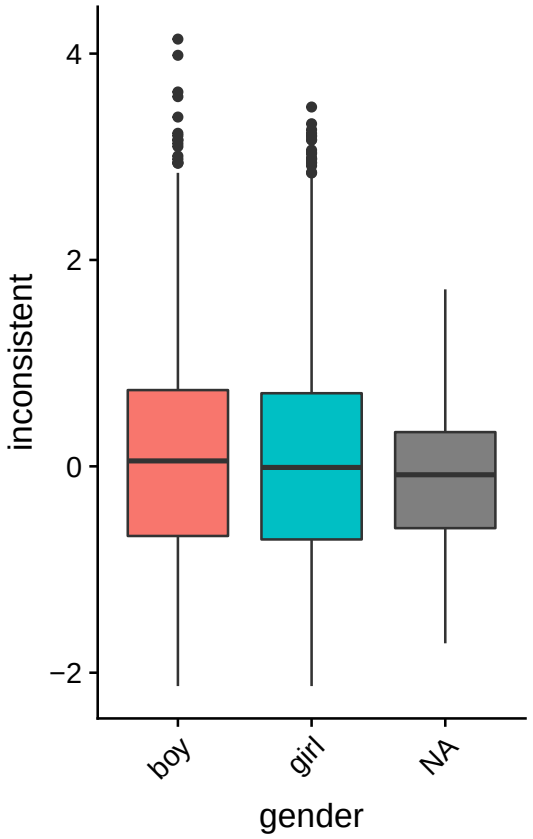
Now lets look at the association with all potential measured confounders (this looks like a lot of code, but it is really only repetition of the same code!):

```
breaks = c(seq(15,40, by = 5),50)
my_data$mAgeg = cut(my_data$mAge, breaks = breaks,ordered_result = T)
my_data$parity = as_factor(my_data$parity,ordered = T)
my_data$birthmonth = ordered(my_data$birthmonth)

by_edu = ggplot(my_data, aes(x = mEDU, y = inconsistent, fill = mEDU)) +
  geom_boxplot(show.legend = F) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
by_age = ggplot(my_data, aes(x = mAgeg, y = inconsistent, fill = mAgeg)) +
  geom_boxplot(show.legend = F) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
by_parity = ggplot(my_data, aes(x = parity, y = inconsistent, fill = parity)) +
  geom_boxplot(show.legend = F) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
by_gender = ggplot(my_data, aes(x = gender, y = inconsistent, fill = gender)) +
  geom_boxplot(show.legend = F) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
cowplot::plot_grid(by_edu,
                   by_age,
                   by_parity,
                   by_gender)
```


inconsistent
4
2
0
−2
basic
1−2 high school
vocational high
general high
Bachelor
Master
NA
mEDU


inconsistent
4
2
0
−2
(15,20]
(20,25]
(25,30]
(30,35]
(35,40]
(40,50]
NA
mAgeg

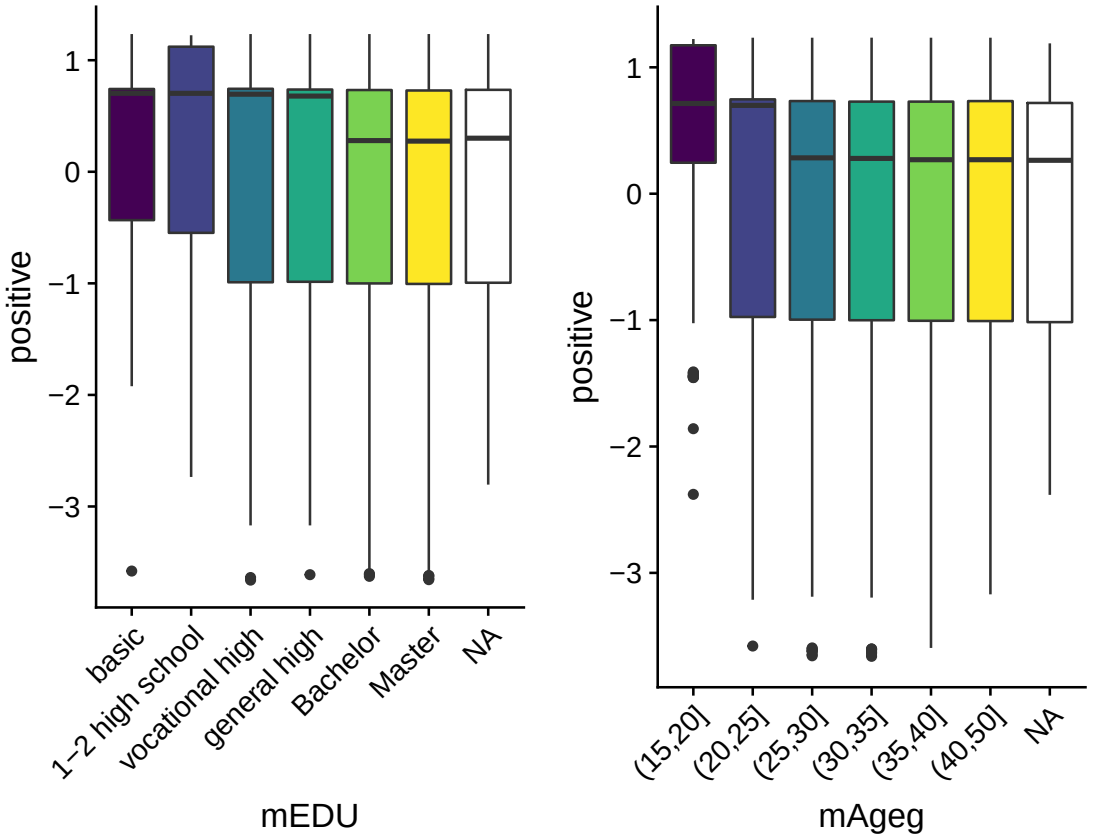

inconsistent
4
2
0
−2
0
1
2
3
>3
NA
parity


inconsistent
4
2
0
−2
boy
girl
NA
gender

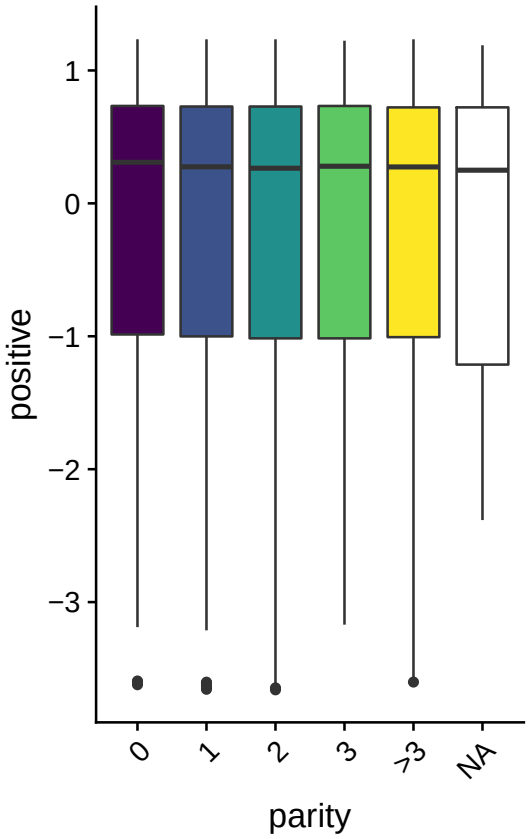
Try to do the same plot but stratify by `birthmonth`.

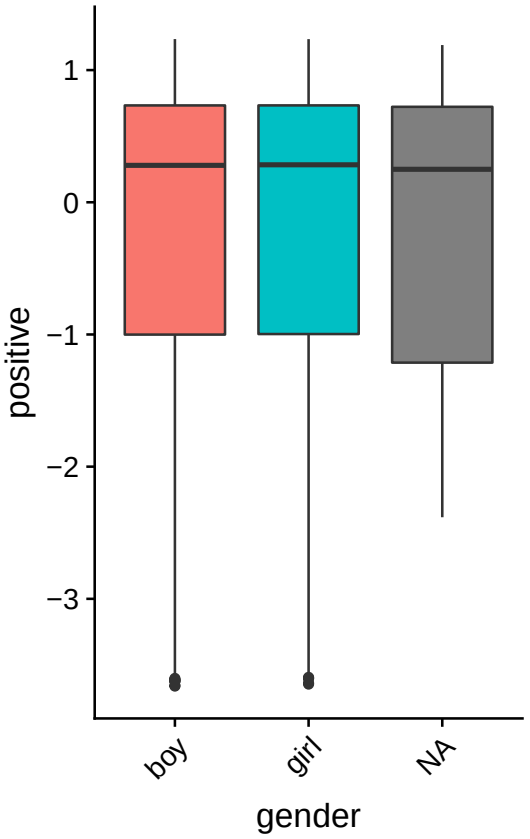
Here is the same plot for positive parenting:

```
by_edu = ggplot(my_data, aes(x = mEDU, y = positive, fill = mEDU)) +
  geom_boxplot(show.legend = F) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

by_age = ggplot(my_data, aes(x = mAgeg, y = positive, fill = mAgeg)) +
  geom_boxplot(show.legend = F) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

by_parity = ggplot(my_data, aes(x = parity, y = positive, fill = parity)) +
  geom_boxplot(show.legend = F) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

by_gender = ggplot(my_data, aes(x = gender, y = positive, fill = gender)) +
  geom_boxplot(show.legend = F) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

plot_grid(by_edu,
          by_age,
          by_parity,
          by_gender)
```


positive
1
0
−1
−2
−3
basic
1−2 high school
vocational high
general high
Bachelor
Master
NA
mEDU


positive
1
0
−1
−2
−3
(15,20]
(20,25]
(25,30]
(30,35]
(35,40]
(40,50]
NA
mAgeg

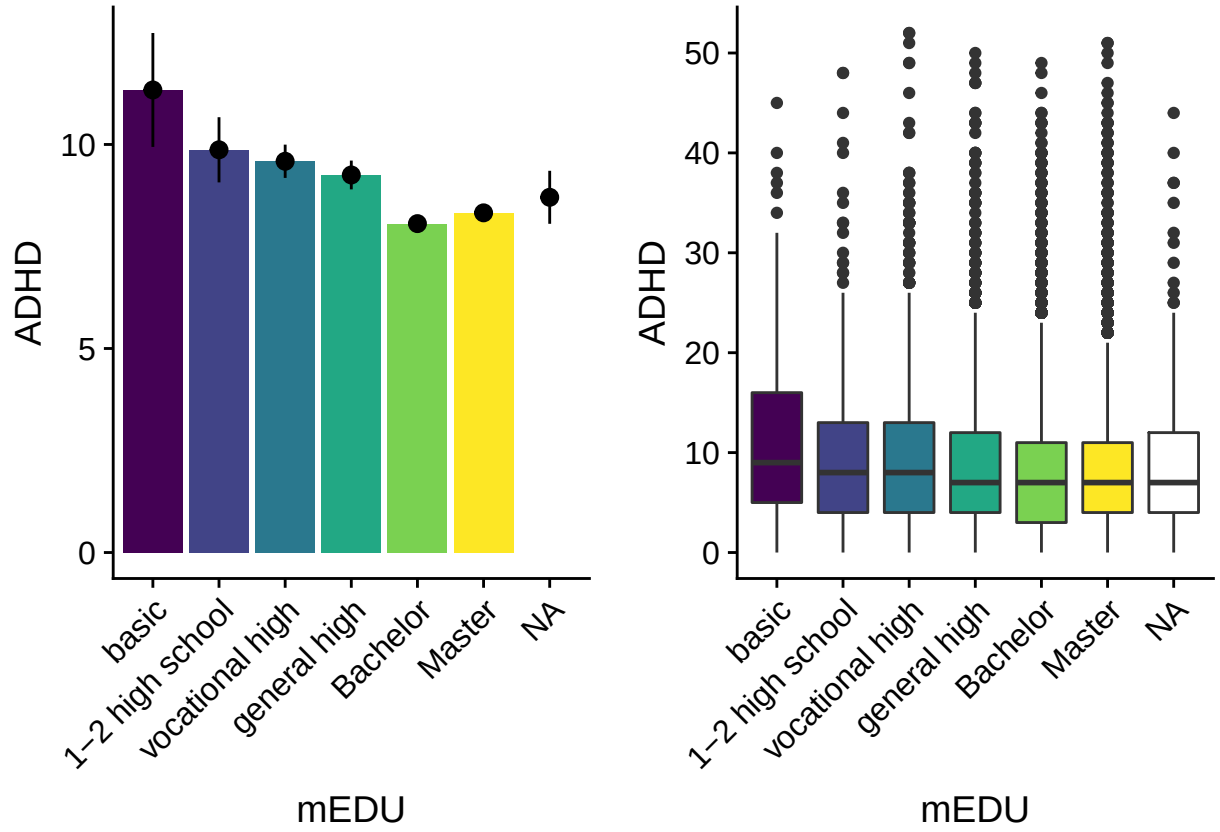

positive
1
0
−1
−2
−3
0
1
2
3
>3
NA
parity


positive
1
0
−1
−2
−3
boy
girl
NA
gender

These associations also do not seem particularly strong.

We conclude this plotting tour with a look at the association between confounders and ADHD symptoms.

```
my_stats = my_data %>%
  group_by(mEDU) %>%
  summarise(m = mean(ADHD),
            sd = sd(ADHD),
            N = n()
            )

my_stats$CIlower = my_stats$m - my_stats$sd/sqrt(my_stats$N)*2
my_stats$CIupper = my_stats$m + my_stats$sd/sqrt(my_stats$N)*2

mean_ci_plot = ggplot(my_stats,aes(x = mEDU, y = m, fill = mEDU)) +
               geom_bar(stat = "identity", show.legend = F) +
               geom_pointrange(aes(x = mEDU, ymin = CIlower, ymax = CIupper), show.legend = F)  +
               ylab("ADHD") +
               theme(axis.text.x = element_text(angle = 45, hjust = 1))

box_plot = ggplot(my_data, aes(x = mEDU, y = ADHD, fill = mEDU)) +
           geom_boxplot(show.legend = F) +
           theme(axis.text.x = element_text(angle = 45, hjust = 1))

plot_grid(mean_ci_plot,
          box_plot)
```



OK, now that we have seen that exposure and outcome are (weakly) associated with a likely confounder, it makes sense to include them into the model.

## A final regression model: Ordinal predictors

```
ADHD_model4 = lm(ADHD ~
                   poly(positive,2, raw = T) +
                   poly(inconsistent,2, raw = T) +
                   mEDU +
                   poly(mAge,2, raw = T) +
                   parity +
                   gender,
                 data = my_data)
summary(ADHD_model4)
```

```
##
## Call:
## lm(formula = ADHD ~ poly(positive, 2, raw = T) + poly(inconsistent,
##     2, raw = T) + mEDU + poly(mAge, 2, raw = T) + parity + gender,
##     data = my_data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -15.14  -4.44  -1.47   2.70  44.57
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       11.4775     1.7455    6.58  5.0e-11 ***
## poly(positive, 2, raw = T)1       -0.5930     0.0621   -9.55  < 2e-16 ***
## poly(positive, 2, raw = T)2       -0.1757     0.0602   -2.92   0.0035 **
## poly(inconsistent, 2, raw = T)1    0.9771     0.0521   18.76  < 2e-16 ***
## poly(inconsistent, 2, raw = T)2    0.1128     0.0403    2.80   0.0052 **
## mEDU.L                            -2.6358     0.3553   -7.42  1.2e-13 ***
## mEDU.Q                             0.3451     0.3147    1.10   0.2728
## mEDU.C                             0.0972     0.2782    0.35   0.7269
## mEDU^4                             0.4561     0.2363    1.93   0.0536 .
## mEDU^5                             0.2551     0.1836    1.39   0.1646
## poly(mAge, 2, raw = T)1           -0.1103     0.1129   -0.98   0.3289
## poly(mAge, 2, raw = T)2            0.0019     0.0018    1.06   0.2911
## parity.L                          -1.6607     0.3784   -4.39  1.1e-05 ***
## parity.Q                           0.9420     0.3228    2.92   0.0035 **
## parity.C                           0.0728     0.2755    0.26   0.7917
## parity^4                          -0.0388     0.1993   -0.19   0.8456
## gendergirl                        -2.3956     0.1009  -23.75  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.7 on 17699 degrees of freedom
##   (486 observations deleted due to missingness)
## Multiple R-squared:  0.0788, Adjusted R-squared:  0.0779
## F-statistic: 94.6 on 16 and 17699 DF,  p-value: <2e-16
```

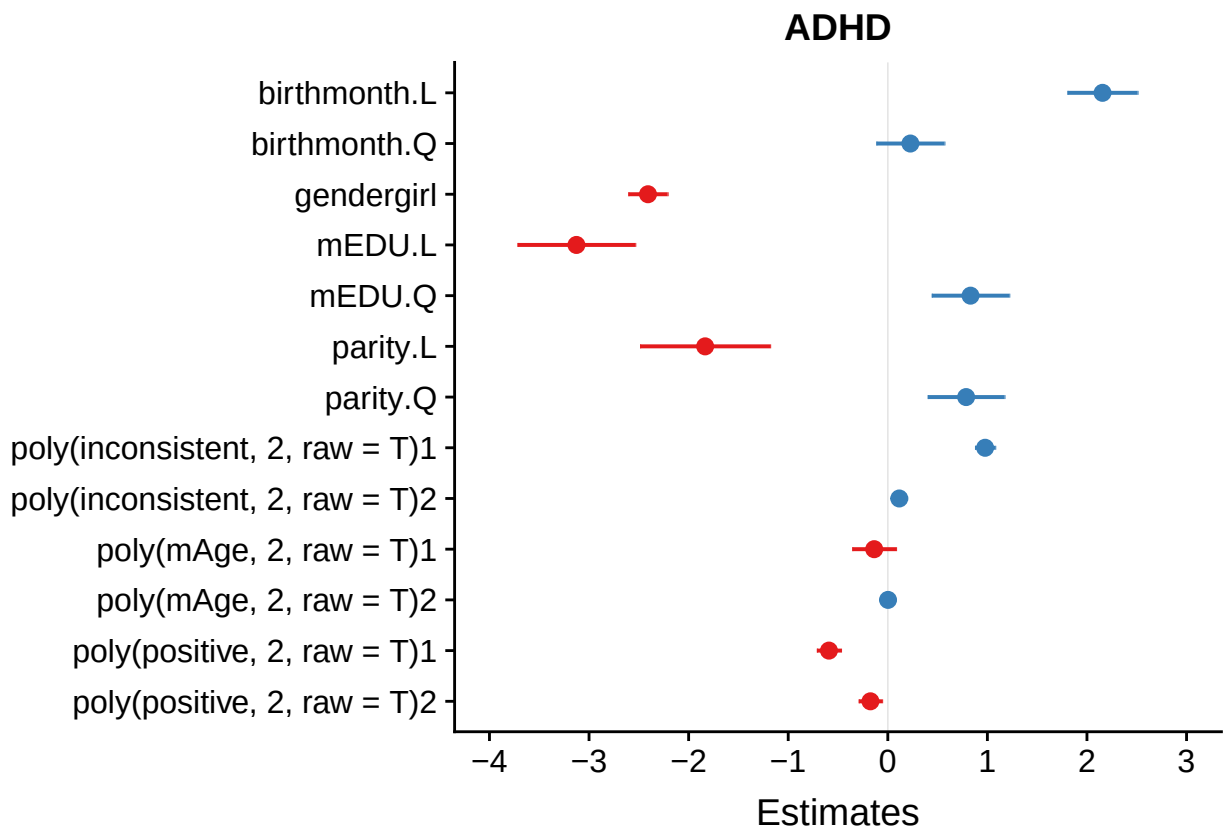
This looks a bit wild. The reason is that if we include ordinal variables like education and parity into the

model, R will by default implement polynomials to a degree equal to `number_levels - 1`. However, we can correct this by setting the contrast to for example a polynomial of order 2.

```
n_levels_Edu = length(levels(my_data$mEDU))
n_levels_parity = length(levels(my_data$parity))
n_levels_bm = length(levels(my_data$birthmonth))
contrasts(my_data$mEDU, 2) <- contr.poly(n_levels_Edu)
contrasts(my_data$parity, 2) <- contr.poly(n_levels_parity)
contrasts(my_data$birthmonth, 2) <- contr.poly(n_levels_bm)

ADHD_model4 = lm(ADHD ~
                   poly(positive,2, raw = T) +
                   poly(inconsistent,2, raw = T) +
                   mEDU +
                   poly(mAge,2, raw = T) +
                   parity +
                   gender +
                   birthmonth,
                 data = my_data)
summary(ADHD_model4)
```

```
##
## Call:
## lm(formula = ADHD ~ poly(positive, 2, raw = T) + poly(inconsistent,
##     2, raw = T) + mEDU + poly(mAge, 2, raw = T) + parity + gender +
##     birthmonth, data = my_data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -14.40  -4.43  -1.45   2.70  44.21
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     12.02315    1.73142    6.94  3.9e-12 ***
## poly(positive, 2, raw = T)1     -0.59085    0.06185   -9.55  < 2e-16 ***
## poly(positive, 2, raw = T)2     -0.17429    0.05996   -2.91   0.0037 **
## poly(inconsistent, 2, raw = T)1  0.97731    0.05190   18.83  < 2e-16 ***
## poly(inconsistent, 2, raw = T)2  0.11521    0.04019    2.87   0.0041 **
## mEDU.L                          -3.12674    0.30151  -10.37  < 2e-16 ***
## mEDU.Q                           0.83087    0.19859    4.18  2.9e-05 ***
## poly(mAge, 2, raw = T)1         -0.13655    0.11211   -1.22   0.2232
## poly(mAge, 2, raw = T)2          0.00227    0.00178    1.28   0.2021
## parity.L                        -1.83421    0.33310   -5.51  3.7e-08 ***
## parity.Q                         0.78658    0.19662    4.00  6.3e-05 ***
## gendergirl                      -2.40801    0.10052  -23.96  < 2e-16 ***
## birthmonth.L                     2.15627    0.17953   12.01  < 2e-16 ***
## birthmonth.Q                     0.22700    0.17441    1.30   0.1931
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.7 on 17702 degrees of freedom
##   (486 observations deleted due to missingness)
## Multiple R-squared:  0.0849, Adjusted R-squared:  0.0843
## F-statistic:  126 on 13 and 17702 DF,  p-value: <2e-16
```

```
plot_model(ADHD_model4)
```




This table suggests that parenting style still has an effect, after we controlled for a number of potential confounders. It is, however, difficult to understand the importance (effect sizes) for the different predictors, because they are measured on different scales (have different means and standard deviations). To get a better idea of effect size we can again plot marginal effects.

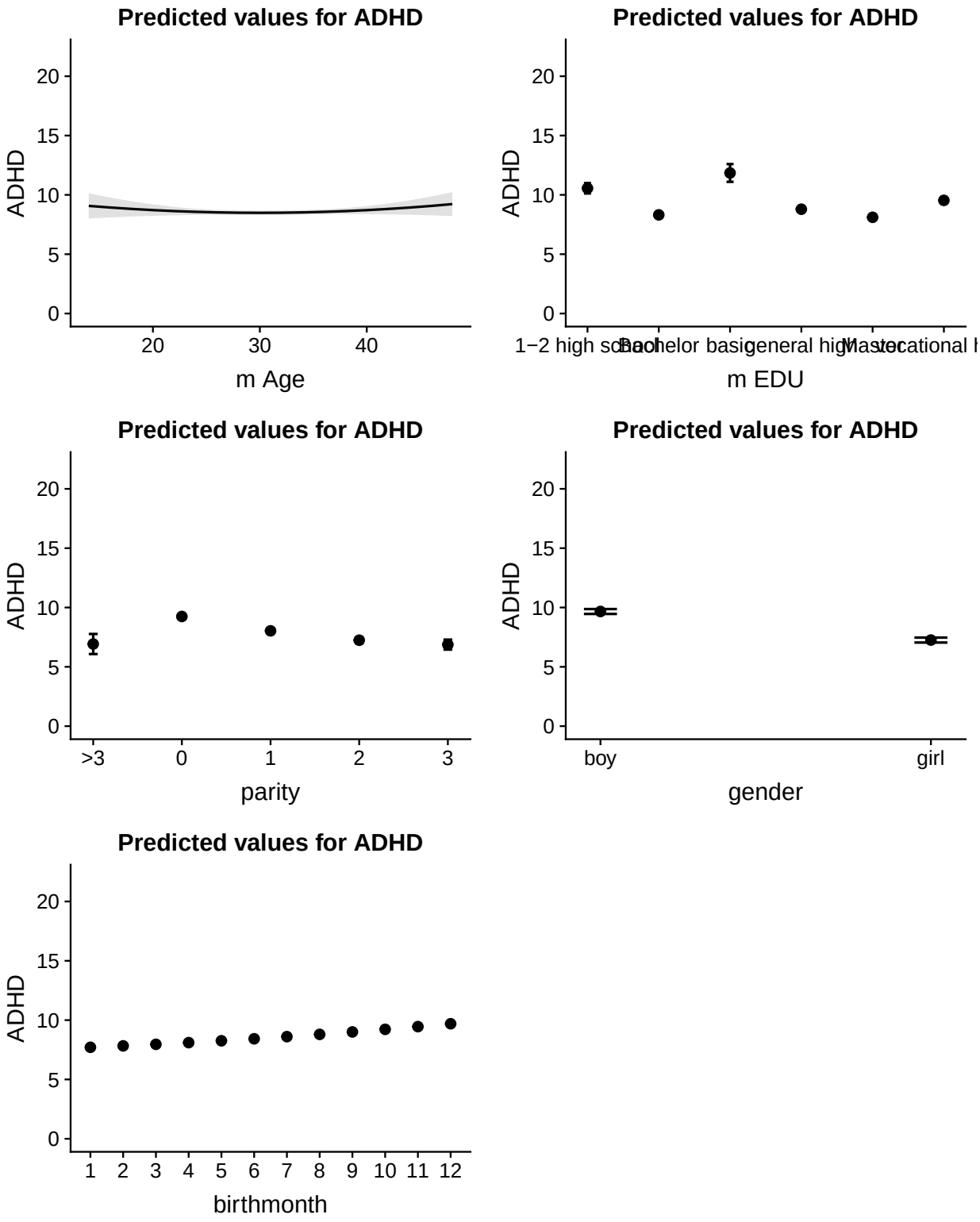
We use a small loop, so that we don't have to write the same lines of code repeatedly for the different predictors. (We also quickly re-fit the model, because the ggeffects package as problems if one uses the `raw = TRUE` options when defining ploynomials.)

```
predictors = c("positive","inconsistent","mAge","mEDU","parity","gender", "birthmonth")
my_data = my_data[complete.cases(my_data[,predictors]),]
ADHD_model4x = lm(ADHD ~
                    poly(positive,2) +
                    poly(inconsistent,2) +
                    mEDU +
                    poly(mAge,2) +
                    parity +
                    gender +
                    birthmonth,
                data = my_data)

ylim = c(0,quantile(my_data$ADHD,c(.95)))

effect_plots = vector(mode = "list", length = length(predictors))
names(effect_plots) = predictors
```

```
for ( p in 1:length(predictors)) {
  effect_plots[[p]] = plot_model(ADHD_model4x, type = "eff", terms = predictors[p]) +
                      coord_cartesian(ylim = ylim)
}
```

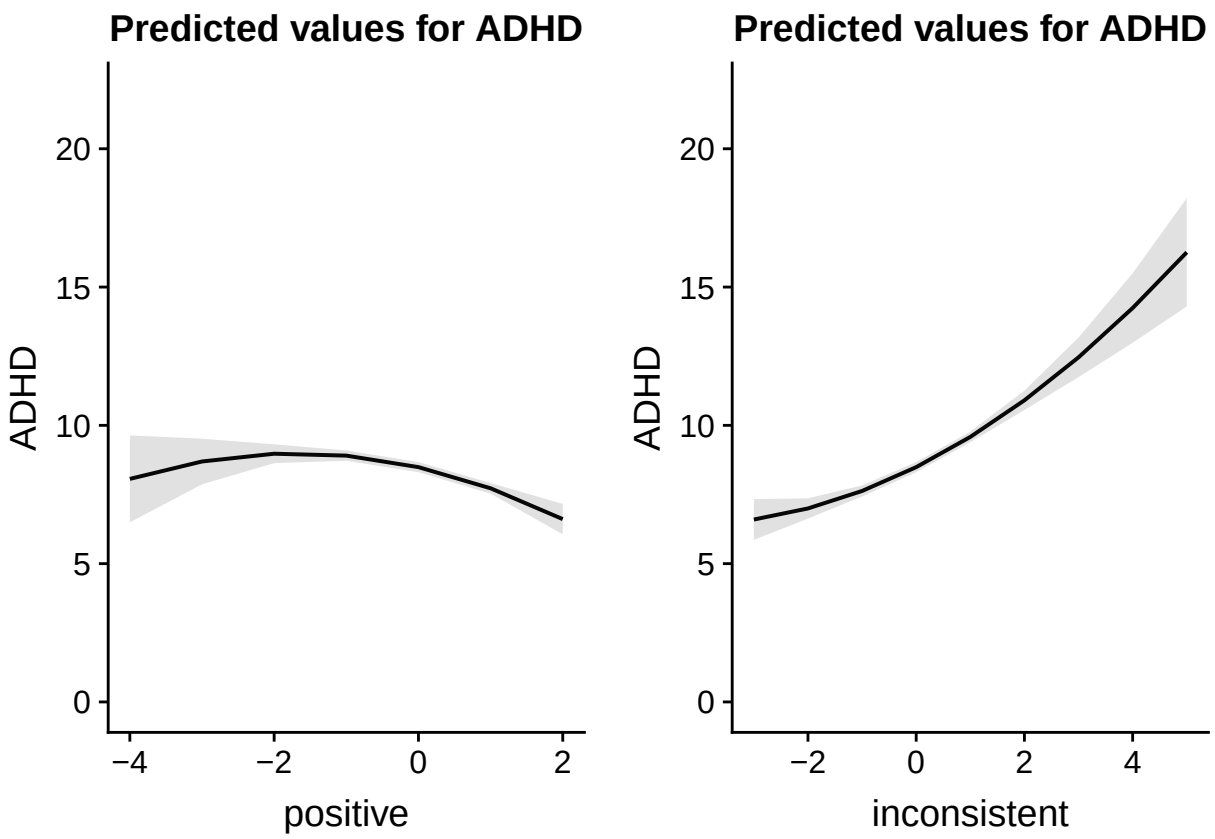
First we look at the effect of confounders:

```
plot_grid(plotlist = effect_plots[3:length(predictors)], nrow = 3)
```




And finally on the effect of our exposures:

```
plot_grid(plotlist = effect_plots[1:2])
```




Interestingly, this suggests that the association between inconsistent parenting style at child age 5 and ADHD sum score at child age 8 increases after we control for some potential confounders.