Real Data Analysis

Anonymous

This Rmd is devoted to an empirical analysis of the same data set as that in Houtepen et al. (2016) and van Kesteren & Oberski (2019), for studying how DNA methylation plays a role in the regulation of human stress reactivity. This Rmd aims to reproduce the results in Section 3 of the paper "High-dimensional mediation analysis for selecting DNA methylation Loci mediating childhood trauma and cortisol stress reactivity".

1. Preprocess raw data

Before running this Rmd, one should first download the data and specify the path of the data.

1.1 Download data and set up correct paths

- Step 1. Download data `E-GEOD-77445.processed.1.zip` (https://www.ebi.ac.uk/arrayexpress/files/E-GEOD-77445/E-GEOD-77445.processed.1.zip) as well as `E-GEOD-77445.processed.2.zip` (https://www.ebi.ac.uk/arrayexpress/files/E-GEOD-77445/E-GEOD-77445.processed.2.zip). Unzip these two zip files in the `data` folder.
- Step 2. Run High-dimensional-mediation-analysis-JASA.Rproj by double clicking its filename to set up working directory
- Step 3. Specify the `data` folder's absolute path as a string to the `PATH` variable below. Caution: the PATH should end with `/` or `\\`. For example, we store the unzipped files in following PATH:

```
PATH = "E:/JASAACS20210115Code/data/"

# Load source codes
source("utils_mediation.R")
```

Note that there is no need to download `E-GEOD-77445.sdrf.txt`, which has different row length and cannot be read into R directly. We have downloaded and manually cleaned it. We saved it as `X.tsv` in the `data` folder because this file contains information about exposure variable.

- Step 3. After finishing Step 1 & 2, you can click `Knit` (in the upper left corner) to run the entire markdown.

1.2 Load data

We loaded these original .txt files and merged them into a clean data set (containing two vectors, exposure variable X as well as response variable y, and two dataframes potential mediators M as well as confounding variables S). To make it convenient for further analysis, we stored the data set in a .RData file.

The following code implements merging the raw data:

```
RawDataPath = LoadRawData(InputPath = PATH)
```

```
## [1] "Success of loading and merging data from .txt files"
## [1] "Raw data is stored in: ./results/raw_data.Rdata"
```

1.3 Marginal screening for potential mediators

We carry out a screening step to retain the top 1000 potential mediators by ranking the absolute value of the product of two correlations - the correlation between $\mathbf{x}$ and each element of $\mathbf{m}$, and between y and each element of $\mathbf{m}$. This indeed is a marginal screening procedure based on Pearson correlation proposed by Fan & Lv (2008).

The following codes are to show marginal screening step

```
topPotential = 1000 #retain the top 1000 potential mediators
cleanedData = MarginalScreening(RawDataPath, topPotential = topPotential)
```

```
## [1] "Preprocessed data is stored in ./results/preproc_JASA.Rdata"
```

Now we are ready to load the cleaned data:

```
load(cleanedData)
```

2. Our proposed new method

Procedure of identifying active mediators in the mediation models (2.1) and (2.2), and estimating the direct effect α_1 and indirect effect $\boldsymbol{\beta}$ that can get around high dimensional matrix estimation.

2.1 Identifying active mediators

we apply the partial penalized least squared method to fit model (2.1) by only penalizing $\boldsymbol{\alpha}_0$, which corresponding to Eq. (2.5).

2.1.1 Implement the algorithm for solving Eq. (2.5)

We apply the local linear approximation algorithm (LLA) in (Zou & Li, 2008) with the SCAD penalty (Fan & Li, 2001), and set $a = 3.7$.

$$p'_\lambda(t) = \lambda\{I(t \leq \lambda) + \frac{(a\lambda - t)_+}{(a-1)\lambda} I(t > \lambda)\},$$

Below is the function of the first order derivative of SCAD penalty

```
deSCAD <- function(z,lamb,a=3.7){
  # First order derivative of SCAD penalty
  # tuning parameter "a" (or "gamma") use the default value 3.7
  return(1*(z<=lamb)+pmax((a*lamb-z),0)/((a-1)*lamb)*(lamb<z))
}
```

The numerical algorithm to solve Eq. (2.5), the partial penalized least squares problem, is given in Section S.4 in the supplementary material of this paper.

Below is the implementation of the local linear approximation Zou and Li (2008). Note that in R package `glmnet`, by passing the weight w to argument `penalty.factor`, we can assign different shrinkage to each coefficient (see details in `help(glmnet)`).

```
## ONE-STEP SPARSE ESTIMATES IN NONCONCAVE PENALIZED LIKELIHOOD MODELS
LLA_h1<- function(X,Y,M,lamb,n,p,q, n_imp = 0,S = NULL){
  #' Local linear approximation algorithm
  #' @param X matrix or dataframe, n by q exposure matrix. q can be 1, and requires q < n
  #' @param Y vector, n-dimensional outcome vector.
  #' @param M matrix or dataframe, n by p mediator matrix and q < n.
  #' @param lamb float, tuning parameter lambda for the penalty
  #' @param n int, the sample size
```

```
  #' @param p int, X matrix dimension (number of columns)
  #' @param q int, M matrix dimension (number of columns)
  #' @param n_imp int, for important mediators that will not be penalized

  if(length(S) == 0){
    s = 0
    V = X
  }else{
    s = ncol(S)
    V = cbind(X,S)
  }
  # Step 1 using Lasso
  w = matrix(0,nrow = (p + q + s),ncol=1)
  w[1:(p-n_imp)] = 1
  alpha_int = coef(glmnet(cbind(M,V),Y,family = 'gaussian', alpha=1,
                          lambda = lamb, penalty.factor=w,intercept = FALSE))[-1]
  # Step 2 using local linear approximation of SCAD
  w = matrix(0,nrow = (p+q + s),ncol=1)
  for(j in 1:(p- n_imp)){
    w[j] = deSCAD(alpha_int[j],lamb)
  }
  alpha = coef(glmnet(cbind(M,V),Y,family = 'gaussian', alpha=1,
                      lambda = lamb, penalty.factor=w, intercept = FALSE))
  return(alpha[-1])
}
```

The tuning parameter λ for our method is chosen based on the high-dimensional BIC (HBIC) method in Wang et al. (2013). The following function is to calculate HBIC for a given tuning parameter λ

```
HBIC_calc <- function(lamb, xx,yy,mm,S = NULL, n_imp =8){
  #‘ Calculate HBIC for a specific tuning parameter lambda
  #' @param lamb float, tuning parameter lambda for penality function
  #' @param xx n by q exposure matrix. q can be 1, and q < n is required
  #' @param yy n-dimensional outcome vector.
  #' @param mm n by p mediator matrix. p can be larger than n.
  #' @param S n by s confounding variable matrix. s can be 1, and s < n is required.
  #' @param n_imp int, number of important mediators that will not be penalized

  #' @return
  #'        A list of class `HBIC_calc'
  #'        - BIC: HBIC score
  #'        - alpha0: estimated alpha0,
  #'        - alpha1: estimated alpha1,
  #'        - alpha2: estimated alpha2,
  #'        - sigma1_hat: estimated sigma1
  #'
  n = nrow(xx)
  p = ncol(mm)
  q = ncol(xx)
  if(is.null(S)){
    s = 0
    result <- LLA_h1(xx,yy,mm,lamb,n,p,q, n_imp = n_imp)
    alpha0 = result[1:p]
    alpha1 = result[(p+1):(p+q)]
```

```
    alpha2 = NULL
    tmp = yy - mm%*%alpha0 - xx%*% alpha1
  }else{
    s = ncol(S)
    result <- LLA_h1(xx,yy,mm,lamb,n,p,q,n_imp = n_imp, S = S)
    alpha0 = result[1:p]
    alpha1 = result[(p+1):(p+q)]
    alpha2 = result[(p+q+1):(p+q+s)]

    tmp = yy - mm%*%alpha0 - xx%*% alpha1 - S %*% alpha2
  }

  df = length(which(alpha0!= 0))+q + s
  sigma_hat = t(tmp)%*%tmp/n
  BIC = log(sigma_hat) + df*log(log(n))*log(p+q + s)/n
  #obj = objective(xx,yy,M,alpha0,alpha1,lamb)
  return(list(BIC=BIC,alpha0=alpha0,alpha1 = alpha1,
              alpha2 = alpha2, sigma1_hat = sigma_hat))
}
```

The following function integrates the LLA algorithm with HBIC parameter tuning. It will choose the λ from given grid points `lam_list` that yields the smallest HBIC value and return the selected mediators.

```
HBIC_PPLS <- function(X, y, M, S = NULL, lam_list =NULL, n_imp =8, topPotential =1000){
  #' This function implements solving the partially penalized least squares with the
  #' SCAD penalty. The tuning parameter lambda for the penalty function is chosen
  #' based on the high-dimensional BIC (HBIC) method.

  #' @param X The n by q exposure matrix. q can be 1, and q < n is required
  #' @param y The n-dimensional outcome vector.
  #' @param M The n by p mediator matrix. p can be larger than n.
  #' @param S The n by s confounding variables matrix. s can be 1, and s < n is required.
  #' @param lam_list a list of tuning parameter for HBIC
  #' @param n_imp int, for important mediators that will not be penalized


  #' @return
  #'
  #'    A dataframe of selected important mediators (denoted as M_{\hat{A}} in the paper)
  hbic= c()

  result =lapply(lam_list, HBIC_calc, xx=X,yy=y, mm=M, S = S, n_imp = n_imp)

  for( ii in 1: length(lam_list)){
    hbic[ii] = result[[ii]]$BIC
  }
  # Find minimum HBIC score's corresponding lambda
  id = which(hbic==min(hbic))
  id = tail(id,1)
  lamb = lam_list[id]
  print(paste("choose lambda:", lamb))
  result = result[[id]]
  alpha0_hat = result$alpha0
  alpha1_hat = result$alpha1
```

```
  A = which(alpha0_hat!=0)
  # Selected mediators
  M_A = M[,c(A[A>topPotential],A[A<=topPotential])]
  return(M_A)
}
```

The candidate set of λ is 50 equally spaced grid from 20 to 70. Below are the codes to specify grid points for λ and then use the `HBIC_PPLS` function to select important mediators:

```
# specify the candidate set of lambda
lamb_min = 20
lamb_max = 70
ngrid = 50
lam_list = seq(lamb_min,lamb_max,length.out = ngrid)

M_A = HBIC_PPLS(X, y, M, S = S, lam_list =lam_list, n_imp =8,
                topPotential = topPotential)
```

```
## [1] "choose lambda: 60.8163265306122"
```

2.1.2 Estimate α_1 and α_0

Estimated Coefficients, SE, t-values and p-values (Table 1)

```
colnames(S) = paste0("Z",0:(ncol(S)-1))
df = data.frame(cbind(M_A, X, S))
round(summary(lm(y~ 0+., data = df))$coefficients,digits = 3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## cg27512205  -237.547    199.506  -1.191    0.238
## cg05608730  -301.168    151.038  -1.994    0.050
## cg26179948  -474.486    160.042  -2.965    0.004
## cg02309301   259.730    108.633   2.391    0.020
## cg12500973    30.029    116.354   0.258    0.797
## cg16657538    84.330     53.236   1.584    0.118
## cg25626453   369.183     97.988   3.768    0.000
## cg13136721   260.990     65.585   3.979    0.000
## cg19230917   321.196    149.918   2.142    0.036
## cg06422529   418.173    107.252   3.899    0.000
## cg03199124   471.865    143.943   3.278    0.002
## X              1.365      4.553   0.300    0.765
## Z0         -3110.834   3805.517  -0.817    0.417
## Z1            -1.864      2.056  -0.906    0.368
## Z2           349.037     82.811   4.215    0.000
## Z3          1843.451   3702.100   0.498    0.620
## Z4           406.642   3533.801   0.115    0.909
## Z5           781.938   3368.749   0.232    0.817
## Z6           967.962   3745.283   0.258    0.797
## Z7           123.714   3544.597   0.035    0.972
## Z8           341.974   3401.318   0.101    0.920
```

2.1.3 Estimate Γ_1 and Γ_2

Estimated Coefficients of Γ_1 and Γ_2 and their p-values (Table 2)

```
cpg = colnames(M_A)
colnames(M_A) = paste0("m",1:ncol(M_A))
Gamma = MediatorExposure(cbind(X, S), M_A)
round(Gamma$Gamma_hat,digits = 3)
```

```
##           X     Z0     Z1     Z2     Z3     Z4     Z5     Z6     Z7     Z8
## m1    0.005 -2.999  0.000  0.016  0.870  0.197 -0.311  0.784  0.406  0.010
## m2    0.007 -1.723 -0.002  0.013  1.479  0.032  0.693  0.602  0.935  1.448
## m3    0.006 -3.566 -0.001  0.004  0.985  0.704 -0.809  0.840  0.567 -0.373
## m4   -0.012 -9.222  0.007 -0.115  5.179  3.716  4.666  5.653  4.371  4.901
## m5   -0.011 -3.974  0.004 -0.009  2.051 -2.022 -0.485  0.437 -0.860 -0.594
## m6    0.023  0.322  0.003  0.403 -6.719 -5.542  0.115 -1.608 -5.701 -4.943
## m7   -0.012  5.701  0.002  0.000  0.562 -0.290  0.552 -0.510 -0.285 -0.516
## m8    0.012  1.093  0.001  0.083 -2.086  3.799  1.748  3.922  2.673  1.312
## m9   -0.006 -2.009  0.001  0.018  1.257 -1.623 -1.448 -0.942 -1.181 -1.276
## m10 -0.008  8.459  0.003  0.042 -7.930 -4.387 -2.036 -3.341 -4.408 -4.387
## m11 -0.006 -5.736  0.003 -0.080  4.745  2.151  1.013  1.080  3.009  2.688
```

```
round(Gamma$Gamma_pvalue,digits = 3)
```

```
##         X    Z0    Z1    Z2    Z3    Z4    Z5    Z6    Z7    Z8
## m1  0.044 0.225 0.711 0.766 0.726 0.936 0.895 0.765 0.868 0.997
## m2  0.016 0.568 0.145 0.848 0.627 0.992 0.811 0.851 0.755 0.613
## m3  0.020 0.201 0.448 0.946 0.724 0.798 0.761 0.776 0.837 0.887
## m4  0.004 0.025 0.001 0.202 0.205 0.356 0.230 0.191 0.277 0.203
## m5  0.002 0.286 0.054 0.917 0.584 0.584 0.892 0.912 0.816 0.866
## m6  0.004 0.968 0.490 0.026 0.406 0.487 0.988 0.850 0.474 0.515
## m7  0.006 0.205 0.444 0.998 0.901 0.948 0.898 0.915 0.949 0.903
## m8  0.056 0.865 0.861 0.563 0.748 0.554 0.777 0.568 0.676 0.830
## m9  0.041 0.528 0.551 0.795 0.695 0.608 0.635 0.781 0.709 0.672
## m10 0.048 0.044 0.223 0.646 0.060 0.288 0.608 0.449 0.286 0.266
## m11 0.045 0.068 0.108 0.253 0.133 0.488 0.734 0.744 0.332 0.364
```

2.2 Test of direct effect and indirect effect

The estimated coefficients, standard errors, test statistics values and p-values (Table 3)

```
direct_indirect_eff <- mediationInference(X, y, M_A, S)
knitr::kable(direct_indirect_eff$summary_result)
```

Coeffcient	Estimated_Coeffcient	SE	Test_statistics	p_value
α_1	1.365	4.553	0.0899	0.7643
β	-17.372	5.495	9.9971	0.0016

2.3 Wrapper function

The above are the break down of our proposed method so that users can follow the analysis step by step. We also provide a wrapper function `hdMediation` that consolidate the estimation and inference steps (section 2.1 ~ 2.2). Calling this function will return selected important mediators' name, estimated direct as well as indirect effect, test statistics and p-value. Note that users should put the unpenalized mediators in the last part of M matrix and use `n_imp` to specify the number of them.

```
fit<-hdMediation(X,y,M, lam_list =lam_list, S=S, n_imp =8)
```

```
## [1] "choose lambda: 60.8163265306122"
```

```
# Important mediators:
print(fit$Mediators_imp)
```

```
##  [1] "cg27512205" "cg05608730" "cg26179948" "cg02309301" "cg12500973"
##  [6] "cg16657538" "cg25626453" "cg13136721" "cg19230917" "cg06422529"
## [11] "cg03199124"
```

```
knitr::kable(fit$summary_result)
```

Coeffcient	Estimated_Coeffcient	SE	Test_statistics	p_value
α_1	1.365	4.553	0.0899	0.7643
β	-17.372	5.495	9.9971	0.0016

2.4 Annotation

Annotation of the included mediators (Table 4)

```
hm450 <- get450k()
probes <- hm450[cpg]
annot <- getNearestTSS(probes)
annot[,'nearestGeneSymbol', drop=FALSE]
```

```
##            nearestGeneSymbol
## cg27512205             KITLG
## cg05608730           C1QTNF2
## cg26179948             JAZF1
## cg02309301            ARGLU1
## cg12500973            HNRNPF
## cg16657538           ZSCAN30
## cg25626453            PRRC2A
## cg13136721             RPTOR
## cg19230917            RAB5IF
## cg06422529               CPQ
## cg03199124            AGPAT1
```

3. Some comparisons

Compare our results with those in Houtepen et al. (2016) and van Kesteren & Oberski (2019) from statistical point of view.

Below are the codes to produce the estimated α_j's and their SE and p-values in model $\mathbf{m}_{(1)}$, $\mathbf{m}_{(2)}$, and $\mathbf{m}_{(3)}$. (Table 5)

```
## m_{(1)}  Houtepen et al (2016)
Houtepen<-analysis_helper(X,y,M_A[,1:3],
                          S, mod_name = '$\\mathbf{m}_{(1)}$')

##m_{(2)} van Kesteren & Oberski (2019)
Kesteren<-analysis_helper(X,y,M_A[,4:8], S, mod_name = '$\\mathbf{m}_{(2)}$')

## m_{(3)} Mediators merge from Houtepen et al (2016) or van Kesteren & Oberski (2019)
combine8<-analysis_helper(X,y,M_A[,1:8], S,  mod_name = '$\\mathbf{m}_{(3)}$')

Table5 <- merge(Houtepen$mod_summary,
```

```
                Kesteren$mod_summary,by = 'row.names',
                all = TRUE, check.names=F)
Table5 <- merge(Table5, combine8$mod_summary,
                by.x = 'Row.names', by.y ='row.names' ,
                all = TRUE, check.names=F)
Table5[is.na(Table5)] <- ""
colnames(Table5)[1] <- " "
knitr::kable(Table5)
```

	$\mathbf{m}_{(1)}$ Estimate	$\mathbf{m}_{(1)}$ SE	$\mathbf{m}_{(1)}$ p-value	$\mathbf{m}_{(2)}$ Estimate	$\mathbf{m}_{(2)}$ SE	$\mathbf{m}_{(2)}$ p-value	$\mathbf{m}_{(3)}$ Estimate	$\mathbf{m}_{(3)}$ SE	$\mathbf{m}_{(3)}$ p-value
m1	-590.499	251.384	0.022				-296.445	245.930	0.232
m2	-576.293	193.816	0.004				-473.583	187.541	0.014
m3	-560.333	217.63	0.012				-535.606	202.237	0.010
m4				283.187	152.175	0.067	223.424	135.335	0.103
m5				248.362	162.892	0.132	155.986	144.278	0.284
m6				98.662	75.037	0.193	44.432	67.583	0.513
m7				395.764	132.243	0.004	321.461	121.227	0.010
m8				279.961	92.063	0.003	198.778	83.166	0.020
X	-5.487	4.916	0.268	-10.713	6.31	0.094	-2.933	5.791	0.614
Z0	-4861.474	4852.906	0.32	905.028	5260.918	0.864	-3098.506	4697.950	0.512
Z1	3.281	2.553	0.203	0.814	2.88	0.778	0.388	2.592	0.881
Z2	370.391	106.637	0.001	322.659	118.831	0.008	356.776	104.691	0.001
Z3	2471.497	4825.767	0.61	-397.456	5185.967	0.939	1095.246	4563.136	0.811
Z4	526.262	4755.095	0.912	-955.42	5090.263	0.852	-483.148	4469.498	0.914
Z5	1822.249	4584.471	0.692	139.567	4879.678	0.977	365.449	4292.333	0.932
Z6	2473.115	5090.954	0.629	-1254.295	5436.242	0.818	284.806	4784.582	0.953
Z7	991.865	4751.88	0.835	-1201.853	5081.203	0.814	-266.494	4465.057	0.953
Z8	726.239	4544.062	0.873	-820.998	4859.867	0.866	-293.975	4277.640	0.945

The estimated coefficients, standard errors, test statistics values and p-values for model $\mathbf{m}_{(1)}$, $\mathbf{m}_{(2)}$, $\mathbf{m}_{(3)}$ (Table 6)

```
Table6 <- rbind(Houtepen$summary_result, Kesteren$summary_result, combine8$summary_result)
row.names(Table6) <- c('$\\mathbf{m}_{(1)}$', '', '$\\mathbf{m}_{(2)}$' ,
                       ' ', '$\\mathbf{m}_{(3)}$', '  ')
knitr::kable(Table6)
```

	Coeffcient	Estimated_Coeffcient	SE	Test_statistics	p_value
$\mathbf{m}_{(1)}$	α_1	-5.487	4.916	1.2456	0.2644
	β	-10.521	3.620	8.4469	0.0037
$\mathbf{m}_{(2)}$	α_1	-10.713	6.310	2.8824	0.0896
	β	-5.295	4.913	1.1614	0.2812
$\mathbf{m}_{(3)}$	α_1	-2.933	5.791	0.2565	0.6125
	β	-13.074	5.356	5.9593	0.0146

4. Relationship among the mediators

4.1 Sample pearson correlation

The following code shows the sample Pearson correlation $\hat{\rho}(m_j, m_k)$ and its p-values for $H_0 : \rho(m_j, m_k) = 0$ (Table S.2) The lower triangle of the table lists their pairwise Pearson correlations, and the upper triangle provides the corresponding p-values for testing the pairwise correlations.

```
knitr::kable(Fisher.helper(M_A))
```

	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10	m11
m1		0.0023	0.0000	0.1236	0.1245	0.3121	0.0067	0.5546	0.0705	0.4381	0.9664
m2	0.3251		0.0310	0.0477	0.0975	0.4100	0.7073	0.0831	0.3186	0.2841	0.0388
m3	0.4327	0.2339		0.2500	0.1163	0.9144	0.0662	0.9424	0.5203	0.0332	0.9829
m4	-0.1684	-0.2152	-0.1263		0.0029	0.3184	0.7900	0.8896	0.0600	0.7308	0.3932
m5	-0.1680	-0.1810	-0.1717	0.3171		0.9710	0.4336	0.6590	0.1132	0.3724	0.0054
m6	-0.1112	-0.0907	-0.0119	0.1097	-0.0040		0.6043	0.6849	0.2166	0.7042	0.1541
m7	-0.2906	-0.0414	-0.2001	-0.0294	0.0862	-0.0572		0.6226	0.4898	0.1591	0.6431
m8	-0.0652	-0.1891	-0.0080	-0.0153	0.0487	0.0448	-0.0543		0.3715	0.8036	0.3788
m9	-0.1971	-0.1097	-0.0709	0.2047	0.1732	-0.1356	-0.0761	-0.0984		0.0132	0.0045
m10	-0.0854	-0.1178	-0.2310	0.0380	0.0982	-0.0419	0.1543	-0.0275	0.2671		0.9098
m11	-0.0046	-0.2243	-0.0024	0.0940	0.2978	-0.1561	-0.0511	-0.0969	0.3036	-0.0125	

4.2 Sample partial correlation

The following codes present the sample partial correlation $\hat{\rho}(m_j, m_k|X, \mathbf{z})$ and its p-values for $H_0 : \rho(m_j, m_k|X, \mathbf{z}) = 0$ (Table S.3)

```
df = data.frame(cbind(M_A, X, S))
par.corr.x = partial.helper(df, 1:11, 12:21 )
knitr::kable(par.corr.x)
```

	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10	m11
m1		0.0082	0.0007	0.1833	0.1861	0.0687	0.0359	0.2039	0.0942	0.8358	0.9873
m2	0.3030		0.0670	0.1639	0.1423	0.0994	0.7098	0.0485	0.4042	0.6679	0.0557
m3	0.3835	0.2126		0.9051	0.3576	0.6441	0.2453	0.4152	0.6706	0.1536	0.8175
m4	-0.1553	-0.1624	-0.0140		0.0919	0.0729	0.1953	0.7749	0.1292	0.5912	0.7082
m5	-0.1544	-0.1711	-0.1077	0.1960		0.3847	0.8624	0.1433	0.6604	0.6346	0.0580
m6	-0.2114	-0.1917	-0.0542	0.2083	0.1018		0.7353	0.7002	0.5919	0.8190	0.8066
m7	-0.2427	0.0437	-0.1358	-0.1512	-0.0204	0.0397		0.8699	0.1523	0.3512	0.2335
m8	-0.1484	-0.2286	-0.0955	0.0336	0.1706	-0.0452	0.0192		0.8430	0.7792	0.6490
m9	-0.1947	-0.0977	-0.0499	0.1768	0.0516	-0.0629	-0.1670	-0.0233		0.0045	0.0280
m10	-0.0243	-0.0503	-0.1664	-0.0630	0.0558	-0.0269	0.1092	-0.0329	0.3244		0.9810
m11	0.0019	-0.2219	0.0271	-0.0439	0.2199	-0.0288	-0.1392	-0.0534	0.2539	0.0028	

4.3 Multi-collinearity between mediators

The R^2, F statistics values and p-values of regression models between mediators to investigate multi-collinearity between mediators. (Table S.4)

```
fit1 <- summary(lm( m1~ m4+m5+m6+m7+m8, data = df))

fit2 <- summary(lm( m1~ m9+m10+m11, data = df))
```

```
fit3 <- summary(lm( m4~ m5+m6+m7+m8, data = df))

fit4 <- summary(lm( m5~ m4+m6+m7+m8, data = df))

fit5 <- summary(lm( m6~ m4+m5+m7+m8, data = df))

S4 <- generateTableS4(list(fit1 = fit1, fit2 = fit2,
                           fit3 = fit3,fit4 = fit4,fit5 = fit5))
tableS4<- data.frame(Dependent_variable = c('m1', 'm1', 'm4','m5','m6'),
                     Independent_variable = c('m4, m5, m6, m7, m8',
                                              'm9, m10, m11',
                                              'm5, m6, m7, m8',
                                              'm4, m6, m7, m8',
                                              'm4, m5, m7, m8'),
                     R2 = S4$R2,
                     F_stat=S4$F_stat,
                     p_value = S4$p_val)
knitr::kable(tableS4, digits  = 4)
```

Dependent_variable	Independent_variable	R2	F_stat	p_value
m1	m4, m5, m6, m7, m8	0.1423	2.6205	0.0303
m1	m9, m10, m11	0.0430	1.2132	0.3103
m4	m5, m6, m7, m8	0.1170	2.6496	0.0392
m5	m4, m6, m7, m8	0.1145	2.5868	0.0430
m6	m4, m5, m7, m8	0.0183	0.3735	0.8269

References

Fan, J. & Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of American Statistical Association*, **96**, 1348-1360.

Fan, J. & Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **70**(5), 849–911.

Houtepen, L.C., Vinkers, C.H., Carrillo-Roa, T., Hiemstra, M., Van Lier, P.A., Meeus, W., Branje, S., Heim, C.M., Nemeroff, C.B., Mill, J. & Schalkwyk, L.C. (2016). Genome-wide DNA methylation levels and altered cortisol stress reactivity following childhood trauma in humans. *Nature Communications*, **7**(1), 10967

van Kesteren, E. J. & Oberski, D. L. (2019). Exploratory Mediation Analysis with Many Potential Mediators *Structural Equation Modeling: A Multidisciplinary Journal*, **26**(5), 710-723.

Wang, L., Kim, Y. & Li, R.(2013). Calibrating non-convex penalized regression in ultra-high dimension. *Annals of Statistics* **41**, 2505-2536

ZOU, H., & LI, R. (2008). One-step sparse estimates in nonconcave penalized likelihood models. *Annals of Statistics* **36**(4), 1509-1533. 35