**Background**

This is a brief post about making [my first Shiny App](https://cillianmacaodh.shinyapps.io/JS_mediation/) ([see also](https://www.cillianmchugh.com/other/js-mediation-shinyapp/)). I made this app following a meeting of the [Advancing Social Cognition lab (ASC-Lab)](https://asclab.wordpress.com/) where we discussed [this paper](https://perso.uclouvain.be/vincent.yzerbyt/Yzerbyt%20et%20al.%20JPSP%202018.pdf) by Yzerbyt et al. ([2018](http://feedproxy.google.com/~r/Cillianmacaodh/~3/wz4vomFAPYQ/#ref-yzerbyt_new_2018)) proposing a new method for mediation analysis. Any attempt to detail the differences in methods is well beyond the scope of a blog post. The take home message is that the method proposed by Yzerbyt et al. ([2018](http://feedproxy.google.com/~r/Cillianmacaodh/~3/wz4vomFAPYQ/#ref-yzerbyt_new_2018)) is less prone to Type I errors (or false positives) than the most commonly used methods (e.g., Hayes [2017](http://feedproxy.google.com/~r/Cillianmacaodh/~3/wz4vomFAPYQ/#ref-hayes_introduction_2017)). In addition to identifying a problem and proposing a solution, the authors also provide the tools to implement their solution with an R package (Batailler et al. [2019](http://feedproxy.google.com/~r/Cillianmacaodh/~3/wz4vomFAPYQ/#ref-R-JSmediation)). Unfortunately, not everyone uses R, and this is why I set about developing a simple way for SPSS users to access this new method.

**Regression and JS Mediation**

Before I describe the [Shiny App](https://cillianmacaodh.shinyapps.io/JS_mediation/), I’ll briefly demonstrate the 2 functions that are included in the [Shiny App](https://cillianmacaodh.shinyapps.io/JS_mediation/). I’ll use the built in dataset mtcars and investigate the relationship between 1/4 mile time (qsec), gross horsepower (hp) and weight (wt), specifically[1](http://feedproxy.google.com/~r/Cillianmacaodh/~3/wz4vomFAPYQ/#fn1):

* does horsepower predict 1/4 mile time?
* and is this relationship mediated by weight?

**Set up the dataframe**

For ease of reusing code (particularly later on) I’ll save mtcars as a dataframe df and rename the variables of interest as iv (predictor variable), dv (outcome variable), and mediator.

df <- mtcars # create df from mtcars

# create new variables with generic names

df$dv <- df$qsec # save 1/4 mile time as dv

df$iv <- df$hp # save horsepower as iv

df$mediator <- df$wt # save weight as mediator

**Simple Regression**

Before running the mediation I’ll run a quick regression to assess the nature of the relationship between the variables.

fit <- lm(dv ~ iv + mediator, data=df) # save the regression in an object 'fit'

summary(fit) # show the results

##

## Call:

## lm(formula = dv ~ iv + mediator, data = df)

##

## Residuals:

## Min 1Q Median 3Q Max

## -1.8283 -0.4055 -0.1464 0.3519 3.7030

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 18.825585 0.671867 28.020 < 2e-16 \*\*\*

## iv -0.027310 0.003795 -7.197 6.36e-08 \*\*\*

## mediator 0.941532 0.265897 3.541 0.00137 \*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 1.09 on 29 degrees of freedom

## Multiple R-squared: 0.652, Adjusted R-squared: 0.628

## F-statistic: 27.17 on 2 and 29 DF, p-value: 2.251e-07

As you can see from the output, 1/4 mile time is predicted by both horsepower and by weight.

**Simple Mediation**

Now that we have a picture of the relationships between the variables we can run the mediation analysis. The code for this is detailed below.

JS\_model <- mdt\_simple(data = df, # create an object 'JS\_model'

DV = dv,

IV = iv,

M = mediator)

add\_index(JS\_model) # display the results of the mediation

## Test of mediation (simple mediation)

## ==============================================

##

## Variables:

##

## - IV: iv

## - DV: dv

## - M: mediator

##

## Paths:

##

## ==== ============== ===== ======================

## Path Point estimate SE APA

## ==== ============== ===== ======================

## a 0.009 0.002 t(30) = 4.80, p < .001

## b 0.942 0.266 t(29) = 3.54, p = .001

## c -0.018 0.003 t(30) = 5.49, p < .001

## c' -0.027 0.004 t(29) = 7.20, p < .001

## ==== ============== ===== ======================

##

## Indirect effect index:

##

## - type: Indirect effect

## - point estimate: 0.00885

## - confidence interval:

## - method: Monte Carlo (5000 iterations)

## - level: 0.05

## - CI: [0.00337; 0.0156]

##

## Fitted models:

##

## - X -> Y

## - X -> M

## - X + M -> Y

* Here we can see that horsepower predicts both 1/4 mile time and weight.
* There is also an indirect effect of horsepower on 1/4 mile time through weight.

**Building a Shiny App**

The full code for the app is below, for the next sections I’ll go through some of the key pieces of code.[2](http://feedproxy.google.com/~r/Cillianmacaodh/~3/wz4vomFAPYQ/#fn2)

**The Geography of the Shiny App**

The [Shiny App](https://cillianmacaodh.shinyapps.io/JS_mediation/) has two panels.

* On the left we have:
  + The data upload option
  + A dropdown menu for selecting the data you wish to use (the uploaded file, the mtcars data set, or the iris data set)
  + Dropdown menus for defining each of your variables,
  + Text describing the App
* On the right we have:
  + The output of the regression
  + The output from the mediation analysis

The code for generating these panels is below (comments above relevant lines describe the purpose of the various sections):

# UI for app

ui<-(pageWithSidebar(

# We use headerPanel() to give a title to our app

headerPanel("JS Mediation"),

# use sidebarPanel() to create the content of the side panel (panel on the left)

sidebarPanel

(

# use fileInput() to create a dialogue for inputting a file

fileInput("file1", "Upload SPSS File",

multiple = TRUE,

accept = c(".sav")),

# create a horizontal line break

tags$hr(),

# create a dropdown menu for selecting the dataset to be used

selectInput("dataset","Data:",

choices =list(iris = "iris",

mtcars = "mtcars",

uploaded\_file = "inFile"), selected=NULL),

# create a dropdown menu for selecting the dependent variable to be used

selectInput("dv","Dependent Variable:", choices = NULL),

# create a dropdown menu for selecting the Independent variable to be used

selectInput("iv","Independent Variable:", choices = NULL),

# create a dropdown menu for selecting the mediator to be used

selectInput("mediator","Mediator:", choices = NULL) #,

# use HTML() to input formatted text describing the App

,HTML('In response to

[this](https://perso.uclouvain.be/vincent.yzerbyt/Yzerbyt%20et%20al.%20JPSP%202018.pdf)

paper by Yzerbyt, Batailler and Judd (2018) which outined a new method of conducting mediation analyses

(with less susceptability to false positives than Hayes’ PROCESS) I created a ShinyApp so that their

R-package could be used by SPSS users. Upload your SPSS file above and select the variables you wish

to compare.')

,br(),br(),br()

,HTML('

Yzerbyt, V., Muller, D., Batailler, C., & Judd, C. M. (2018). New Recommendations for

Testing Indirect Effects in Mediational Models: The Need to Report and Test Component Paths.

*Journal of Personality and Social Psychology: Attitudes and Social Cognition*, 115(6),

929–943. <http://dx.doi.org/10.1037/pspa0000132>

')

),

# use mainPanel() to create the panel on the right where the output of our tests will be

mainPanel(

# give a title to the the first output

h3("Summary of Regression Model"),

# report the result of the regression, saved in the object 'fit'

verbatimTextOutput("fit"),

# give a title for the second output

h3("Mediation Results"),

# report the result of the mediation, saved in the object 'mediation'

verbatimTextOutput("mediation")

)

))

**The Backend of the Shiny App**

Above we have the code for setting up and modifying the look and feel of our app. Below we go through the code for making the app do what it is supposed to do. The code in full is at the bottom of this post, however I have isolated specific sections of code to describe their function.

**Inputting data from file**

The code below runs read.spss() on whatever file you have uploaded using the dialogue box in the side panel and creates a dataframe called inFile.

upload\_data<-reactive({

inFile <- input$file1

if (is.null(inFile))

return(NULL)

read.spss(input$file1$datapath, to.data.frame = TRUE)

})

observeEvent(input$file1,{

inFile<<-upload\_data()

})

**Selecting data and variables**

The code below retrieves information about the dataset that is selected, and displays the variables associated with the selected dataset in the dropdown menus for each of your variables (IV, DV, & mediator).

# update variables based on the data

observe({

# make sure upload exists

if(!exists(input$dataset)) return()

# retrieve names of columns (variable names) and save as 'var.opts'

var.opts<-colnames(get(input$dataset))

# set var.opts as the options for the drop down menus

updateSelectInput(session, "dv", choices = var.opts)

updateSelectInput(session, "iv", choices = var.opts)

updateSelectInput(session, "mediator", choices = var.opts)

})

**Setting up data for analysis**

Below we extract the data and variables selected in the dropdown menus and save them as objects that we can use in functions. Specifically we create a list obj which contains the vectors dv, iv, and mediator.

# get data object

get\_data<-reactive({

if(!exists(input$dataset)) return() # if no upload

check<-function(x){is.null(x) || x==""}

if(check(input$dataset)) return()

# retrieve the selected data and create objects and

obj<-list(data=get(input$dataset),

dv=input$dv,

iv=input$iv,

mediator=input$mediator

)

# require all to be set to proceed

if(any(sapply(obj,check))) return()

# make sure choices had a chance to update

check<-function(obj){

!all(c(obj$dv,obj$iv,obj$mediator) %in% colnames(obj$data))

}

if(check(obj)) return()

# return 'obj' on completion

obj

})

**Running the analyses**

Now that we can retrieve the selected data and variables, we can turn them into a dataframe and run our analyses on them.

**Regression**

The code below creates an object output$fit which contains the output of the regression.

output$fit <- renderPrint({

# create an object 'data\_list', which is a list that contains the selected data and variables

dataset\_list <- get\_data()

# isloate the elements in the list as separate objects

a <- dataset\_list$dv

b <- dataset\_list$iv

m <- dataset\_list$mediator

c <- dataset\_list$data

# create a dataframe 'df' from the object 'c' the selected dataset

df <- `colnames<-`(

cbind.data.frame(

# we extract and use the variables from 'c' that have the same names as those selected

c[which(colnames(c)==a)],

c[which(colnames(c)==b)],

c[which(colnames(c)==m)]

), c("dv","iv","mediator"))

# now we have a dataframe df with 3 variables named 'dv', 'iv', and 'mediator'

# we need to ensure data is numeric

df$dv <- suppressWarnings(as.numeric(df$dv))

df$iv <- suppressWarnings(as.numeric(df$iv))

df$mediator <- suppressWarnings(as.numeric(df$mediator))

# using the same code previously discussed, we run the regression

fit <- lm(dv ~ iv + mediator, data=df)

summary(fit) # show results

})

**Mediation**

Below we follow mostly the same steps to create our dataframe, and this time we run the mediation instead of the regression.

output$mediation <- renderPrint({

# create an object 'data\_list', which is a list that contains the selected data and variables

dataset\_list <- get\_data()

# isloate the elements in the list as separate objects

a <- dataset\_list$dv

b <- dataset\_list$iv

m <- dataset\_list$mediator

c <- dataset\_list$data

# create a dataframe 'df' from the object 'c' the selected dataset

df <- `colnames<-`(

cbind.data.frame(

# we extract and use the variables from 'c' that have the same names as those selected

c[which(colnames(c)==a)],

c[which(colnames(c)==b)],

c[which(colnames(c)==m)]

), c("dv","iv","mediator"))

# now we have a dataframe df with 3 variables named 'dv', 'iv', and 'mediator'

# we need to ensure data is numeric

df$dv <- suppressWarnings(as.numeric(df$dv))

df$iv <- suppressWarnings(as.numeric(df$iv))

df$mediator <- suppressWarnings(as.numeric(df$mediator))

# and we run the mediation using the same code as at the beginning of this post

JS\_model <- mdt\_simple(data = df,

DV = dv,

IV = iv,

M = mediator)

add\_index(JS\_model)

})

**Conclusion**

Above I have described how I went about making my first [Shiny App](https://cillianmacaodh.shinyapps.io/JS_mediation/) which makes a new method of mediation analysis accessible to SPSS users. Feel free to try it out (although I have not paid for a premium account with Shiny, so it might time out).

Both the mtcars dataset and the iris dataset are preloaded in the app if you want to try it but you don’t have any SPSS files to upload. If you are an R user hopefully this post might help you to make your own Shiny Apps to make R functionality available to your SPSS using colleagues. Many thanks to the examples online that helped me, particularly [this example for uploading files and working with them](https://gist.github.com/dgrapov/dfcf6ab2339b1cf19b090cfb8dadc855).

(also if you have any suggestions for improving the app, or if I have left anything out, let me know)

library(shiny)

library(foreign)

library(purrr)

library(dplyr)

library("devtools")

#install.packages("JSmediation")

library(JSmediation)

# UI for app

ui<-(pageWithSidebar(

# We use headerPanel() to give a title to our app

headerPanel("JS Mediation"),

# use sidebarPanel() to create the content of the side panel (panel on the left)

sidebarPanel

(

# use fileInput() to create a dialogue for inputting a file

fileInput("file1", "Upload SPSS File",

multiple = TRUE,

accept = c(".sav")),

# create a horizontal line break

tags$hr(),

# create a dropdown menu for selecting the dataset to be used

selectInput("dataset","Data:",

choices =list(iris = "iris",

mtcars = "mtcars",

uploaded\_file = "inFile"), selected=NULL),

# create a dropdown menu for selecting the dependent variable to be used

selectInput("dv","Dependent Variable:", choices = NULL),

# create a dropdown menu for selecting the Independent variable to be used

selectInput("iv","Independent Variable:", choices = NULL),

# create a dropdown menu for selecting the mediator to be used

selectInput("mediator","Mediator:", choices = NULL) #,

# use HTML() to input formatted text describing the App

,HTML('In response to

[this](https://perso.uclouvain.be/vincent.yzerbyt/Yzerbyt%20et%20al.%20JPSP%202018.pdf)

paper by Yzerbyt, Batailler and Judd (2018) which outined a new method of conducting mediation analyses

(with less susceptability to false positives than Hayes’ PROCESS) I created a ShinyApp so that their

R-package could be used by SPSS users. Upload your SPSS file above and select the variables you wish

to compare.')

,br(),br(),br()

,HTML('

Yzerbyt, V., Muller, D., Batailler, C., & Judd, C. M. (2018). New Recommendations for

Testing Indirect Effects in Mediational Models: The Need to Report and Test Component Paths.

*Journal of Personality and Social Psychology: Attitudes and Social Cognition*, 115(6),

929–943. <http://dx.doi.org/10.1037/pspa0000132>

')

),

# use mainPanel() to create the panel on the right where the output of our tests will be

mainPanel(

# give a title to the the first output

h3("Summary of Regression Model"),

# report the result of the regression, saved in the object 'fit'

verbatimTextOutput("fit"),

# give a title for the second output

h3("Mediation Results"),

# report the result of the mediation, saved in the object 'mediation'

verbatimTextOutput("mediation")

)

))

# shiny server side code for each call

server<-(function(input, output, session){

# update variables based on the data

observe({

#browser()

if(!exists(input$dataset)) return() #make sure upload exists

var.opts<-colnames(get(input$dataset))

updateSelectInput(session, "dv", choices = var.opts)

updateSelectInput(session, "iv", choices = var.opts)

updateSelectInput(session, "mediator", choices = var.opts)

})

# get data object

get\_data<-reactive({

if(!exists(input$dataset)) return() # if no upload

check<-function(x){is.null(x) || x==""}

if(check(input$dataset)) return()

obj<-list(data=get(input$dataset),

dv=input$dv,

iv=input$iv,

mediator=input$mediator

)

# require all to be set to proceed

if(any(sapply(obj,check))) return()

#make sure choices had a chance to update

check<-function(obj){

!all(c(obj$dv,obj$iv,obj$mediator) %in% colnames(obj$data))

}

if(check(obj)) return()

obj

})

upload\_data<-reactive({

inFile <- input$file1

if (is.null(inFile))

return(NULL)

# could also store in a reactiveValues

read.spss(input$file1$datapath, to.data.frame = TRUE)

})

observeEvent(input$file1,{

inFile<<-upload\_data()

})

# create regression output

output$fit <- renderPrint({

dataset\_list <- get\_data()

a <- dataset\_list$dv

b <- dataset\_list$iv

m <- dataset\_list$mediator

c <- dataset\_list$data

df <- `colnames<-`(

cbind.data.frame(

c[which(colnames(c)==a)],

c[which(colnames(c)==b)],

c[which(colnames(c)==m)]

), c("dv","iv","mediator"))

df$dv <- suppressWarnings(as.numeric(df$dv))

df$iv <- suppressWarnings(as.numeric(df$iv))

df$mediator <- suppressWarnings(as.numeric(df$mediator))

fit <- lm(dv ~ iv + mediator, data=df)

summary(fit) # show results

})

# create mediation output

output$mediation <- renderPrint({

dataset\_list <- get\_data()

a <- dataset\_list$dv

b <- dataset\_list$iv

m <- dataset\_list$mediator

c <- dataset\_list$data

df <- `colnames<-`(

cbind.data.frame(

c[which(colnames(c)==a)],

c[which(colnames(c)==b)],

c[which(colnames(c)==m)]

), c("dv","iv","mediator"))

df$dv <- suppressWarnings(as.numeric(df$dv))

df$iv <- suppressWarnings(as.numeric(df$iv))

df$mediator <- suppressWarnings(as.numeric(df$mediator))

JS\_model <- mdt\_simple(data = df,

DV = dv,

IV = iv,

M = mediator)

add\_index(JS\_model)

})

# #JS\_model

})

# Create Shiny app ----

shinyApp(ui, server)

**References**

Batailler, Cédric, Dominique Muller, Vincent Yzerbyt, and Charles Judd. 2019. *JSmediation: Mediation Analysis Using Joint Significance*. [https://CRAN.R-project.org/package=JSmediation](https://cran.r-project.org/package=JSmediation).

Hayes, Andrew F. 2017. *Introduction to Mediation, Moderation, and Conditional Process Analysis, Second Edition: A Regression-Based Approach*. Guilford Publications.

Yzerbyt, Vincent, Dominique Muller, Cédric Batailler, and Charles M Judd. 2018. “New Recommendations for Testing Indirect Effects in Mediational Models: The Need to Report and Test Component Paths.” *Journal of Personality and Social Psychology: Attitudes and Social Cognition* 115 (6): 929–43. [https://doi.org/http://dx.doi.org/10.1037/pspa0000132](https://doi.org/http:/dx.doi.org/10.1037/pspa0000132).

1. The purpose of this post is to demonstrate the code for these analyses, as such there may be issues with the analyses reported – I haven’t checked any assumptions or anything.[](http://feedproxy.google.com/~r/Cillianmacaodh/~3/wz4vomFAPYQ/#fnref1)
2. In order to enable people to use the app for their own analysis I needed a way for them to upload their data into the app. After a bit of googling I found [this example](https://gist.github.com/dgrapov/dfcf6ab2339b1cf19b090cfb8dadc855), for uploading .csv files. I copied the code and modified it to include read.spss() from the package foreign instead of read.csv()[](http://feedproxy.google.com/~r/Cillianmacaodh/~3/wz4vomFAPYQ/#fnref2)