Disability IAT: Walk-Through of Cleaning and Analysis

Jenna Harder, MA

December 23, 2018

## Introduction

**Before You Click “Knit”:** Be aware that you’ll need to install the car package if you don’t have it already. I’ve commented out the installation command here.

In addition, please note that this will take some time to knit, because the data file is quite large.

#install.packages("car")

## The Data

My collaborators and I downloaded this data from the Open Science Framework page managed by the Project Implicit team at Harvard: <https://osf.io/y9hiq/>

Project Implicit is a website (<https://www.projectimplicit.net/index.html>) that collects data on the implicit and explicit (read: conscious and unconscious) prejudices of visitors to the site.

We looked at data from the Project Implicit tests of prejudice against people with disabilities (hereafter “disability prejudice”). The data include approximately 700,000 participants who visited the page between April of 2004 and December of 2017.

Note: I worked on this project with my colleagues Victor Keller and William Chopik, both of whom contributed invaluably to all stages of the project. However, as project leader I handled the bulk of the analysis, and the subset presented here is my own work.

Let’s start by grabbing the raw data and loading some packages. (Be patient: this dataset may take a few minutes to load!)

setwd("~/Disability IAT")  
diat<-read.csv("disabilityIATwithExclusions.csv", header=T, stringsAsFactors = F)  
library(car)  
dim(diat) #how big is this dataframe?

## [1] 319026 185

## First Glance

Let’s look at the first five lines of the first five variables:

diat[1:5,1:5]

## ï..exclude date D\_biep.Abled\_Good\_all Order Side\_Abled\_34  
## 1 0 4/30/2004 20:15:40 0.8233208 1 1  
## 2 0 5/1/2004 2:42:04 0.8918659 2 2  
## 3 0 5/1/2004 9:19:46 0.8508314 1 1  
## 4 0 5/1/2004 12:02:23 1.0045926 2 2  
## 5 0 5/1/2004 22:53:18 0.5184749 2 2

The first variable allows us to exclude participants based on certain criteria–but its name is messed up because we converted this from an SPSS file to a csv file. It’s also unneccessary for this walk-through because I’ve already excluded most of the cases I want to exclude. We’ll replace it with a unique identifier for the participants, since the dataset doesn’t already have one:

#I want to name the new variable "id", but first let's make sure that doesn't duplicate any variable names:  
"id" %in% names(diat)

## [1] FALSE

#It doesn't, so we can go ahead and rename the variable:  
names(diat)[1] <- "id" #rename variable  
  
#Create an id number for each person--we can just use the row numbers:  
diat$id <- 1:nrow(diat)

This is a very rich dataset, and my colleagues and I addressed quite a few questions with it. Here, though, we’ll focus on just one of the questions we answered: What predicts a participant’s level of implicit disability prejudice?

## Cleaning

Implicit disability prejudice is represented by the participant’s D score, derived from reaction times on an Implicit Association Test.

(Side note: The D score is actually a questionable measure and has received some criticism: it’s computed as the difference between two numbers, and computing difference scores causes some methodological issues such as lowered reliability. If we had the raw data, we could analyze participants’ behavior more appropriately; but D scores are still widely accepted in the field of implicit prejudice research, and it’s the only thing the Project Implicit website included, so we’re stuck with it.)

In this dataset, the D score variable is the one we saw above with the name “D\_biep.Abled\_Good\_all.” Let’s rename it to something a little easier to work with:

"Dscore" %in% names(diat) #make sure we're not duplicating a pre-e0000xisting variable name

## [1] FALSE

names(diat)[3] <- "Dscore"

The variables I’d like to use to predict “Dscore” are the following:

Date (when during the 13-year study period did the individual participate?)

Age

Gender

Educational Attainment

Disability Status (does the individual have a disability herself?)

Contact with People with Disabilities (does the individual know someone with a disability?)

Many of those variables are pretty messy right now.

# Date

Let’s start by cleaning the date variable, which is currently a text variable, and includes a timestamp as well:

head(diat$date)

## [1] "4/30/2004 20:15:40" "5/1/2004 2:42:04" "5/1/2004 9:19:46"   
## [4] "5/1/2004 12:02:23" "5/1/2004 22:53:18" "5/2/2004 0:06:34"

We want a numerical value representing what day a person participated during these 13 years, and for now we’re not interested in looking at time of day as a predictor.

So we’ll start by getting rid of the timestamp, which we can easily do by splitting these strings on the space (" “) character:

"datedate" %in% names(diat) #make sure this variable name not already in use

## [1] FALSE

diat$datedate <- sapply(diat$date, function(x) (strsplit(x, split = " ", fixed=T)[[1]][1]))

Next, we need to figure out whether there are any days during the study period when no one participated. We can do this by creating a vector of all the dates between April 30, 2004 and December 31, 2017 and finding the mismatches between that vector and the unique recorded dates.

We can just look at the day-of-the-month (1 through 31, etc., for each month):

monthLengths <-c(1:31,1:28,1:31,1:30,1:31,1:30,1:31,1:31,1:30,1:31,1:30,1:31)  
leapYear <-c(1:31,1:29,1:31,1:30,1:31,1:30,1:31,1:31,1:30,1:31,1:30,1:31)  
  
#month lengths for our study period, 2004 to 2017:  
studyPeriod <- c(monthLengths, monthLengths, monthLengths, monthLengths, leapYear, monthLengths, monthLengths, monthLengths, leapYear, monthLengths, monthLengths, monthLengths, leapYear, monthLengths)  
  
#Technically, the first item in that vector should have said "leapYear," but it doesn't matter because our data start part way through 2004: they don't include February. We'll eliminate the first 119 days:  
studyPeriod <- studyPeriod[120:length(studyPeriod)]  
  
#Now, let's see how that number compares to the number of unique days on which people participated:  
uniqueDates <- unique(diat$datedate)  
c(length(studyPeriod), length(uniqueDates)) #print the respective lengths of studyPeriod and uniqueDates

## [1] 4994 4976

We now have two helpful vectors: uniqueDates lists the dates on which someone participated, and studyPeriod lists the dates on which someone *could have* participated. We can see that uniqueDates is shorter by 18 items, so we know there were 18 days when no one participated.

We’ll grab the day of the month from uniqueDates (e.g., “5/27/06” will become “27”) so that it matches the content of studyPeriod. Since uniqueDates is in chronological order, and so is studyPeriod, the two vectors will be identical until–and only until–the first day that no one participated. We can use this fact to identify all 18 of these “missing” days.

#grab day  
dayOfMonth <- sapply(uniqueDates, function(x) (strsplit(x, split = "/", fixed=T)[[1]][2]))  
dayOfMonth2<-as.integer(dayOfMonth) #we'll modify this integer version in our loop below  
  
nobodyDays <- vector() #initiate vector that we'll use to keep track of the missing days  
studyPeriod2<-studyPeriod #copy studyPeriod to a new vector that we can modify  
a<-0 #we'll start from day zero  
while(length(studyPeriod2)!=length(dayOfMonth2)){ #we'll be knocking sections off of these vectors each time we identify a missing day. When we've found all 18, they'll be the same length.  
 b <-sum(dayOfMonth2==studyPeriod2) + 1 #identify index number of next mismatch between current iterations of dayOfMonth2 and studyPeriod: the next missing day  
 a<-a + sum(dayOfMonth2==studyPeriod2) #identify index number of date immediately before that missing date in the study period  
 nobodyDays <- c(nobodyDays,a) #append a to nobodyDays, growing our vector of [index numbers of days immediately before] missing days  
 dayOfMonth2 <- as.integer(dayOfMonth2[b:length(dayOfMonth2)]) #shrink dayOfMonth2 to the days after the identified missing day  
 studyPeriod2 <- studyPeriod2[(b+1):length(studyPeriod2)] #shrink studyPeriod2 to the days after the identified missing day  
}  
cbind(uniqueDates[nobodyDays], uniqueDates[nobodyDays+1]) #now, using nobodyDays, we can see where our 18 missing days are:

## [,1] [,2]   
## [1,] "6/4/2004" "6/6/2004"   
## [2,] "8/7/2004" "8/9/2004"   
## [3,] "8/13/2004" "8/15/2004"   
## [4,] "8/21/2004" "8/22/2004"   
## [5,] "8/27/2004" "8/29/2004"   
## [6,] "10/30/2004" "11/1/2004"   
## [7,] "12/23/2004" "12/25/2004"  
## [8,] "12/25/2004" "12/27/2004"  
## [9,] "9/13/2005" "9/15/2005"   
## [10,] "9/20/2005" "9/23/2005"   
## [11,] "9/20/2005" "9/23/2005"   
## [12,] "9/26/2005" "10/4/2005"   
## [13,] "9/26/2005" "10/4/2005"   
## [14,] "9/26/2005" "10/4/2005"   
## [15,] "9/26/2005" "10/4/2005"   
## [16,] "9/26/2005" "10/4/2005"   
## [17,] "9/26/2005" "10/4/2005"   
## [18,] "9/26/2005" "10/4/2005"

This table shows us the gaps in participation date (some of them duplicated on multiple rows when there was a string of days with no participation). We have zero participants for June 5th 2004, August 8th 2004, August 14th 2004, etc.

Knowing this, we can now create a new variable, numdate, and populate it with numbers indicating the number of days since April 29th, 2004. (This takes a minute to run:)

"numdate" %in% names(diat) #make sure this name not already in use

## [1] FALSE

diat$numdate <- NA  
a<-1 #we'll use this to tell the for loop what row to start at  
  
#nobodyDays has one item for each "missing" day (days when no one participated)  
#the values represent the index number of the uniqueDates value corresponding to the last day on which someone participated before the missing day  
#we'll create nobodyDays2, which gets rid of duplicates (when multiple days were missing in a row) and appends the length of dayOfMonth (i.e., our number of non-missing days--although it doesn't really matter what number we put here)  
nobodyDays2 <- c(unique(nobodyDays),length(dayOfMonth))  
  
#now we'll create breakdays2, which will have the diat row numbers for the last day before each missing day (not counting duplicates)  
breakdays2<-sapply(unique(uniqueDates[nobodyDays]), FUN = function(x){  
 max(which(diat$datedate == x))  
})  
#and we'll add the last row number to the end of breakdays2  
breakdays3 <- c(breakdays2,nrow(diat))   
  
#this for loop uses breakdays3 to loop through each section of continuously non-missing days  
for(i in(breakdays3)){  
 b <- nobodyDays2[which(nobodyDays2==which(uniqueDates==diat$datedate[i]))] #b is the value of nobodyDays2 corresponding to the end of the current section   
 for(j in(a:i)){ #for each row in this section  
 diat$numdate[j] <- which(uniqueDates == diat$datedate[j]) + #the index number of the corresponding uniqueDates date is sort of the number of days since April 29 2004...  
 #...but not quite, because uniqueDates doesn't include the missing days.  
 length(nobodyDays[which(nobodyDays<b)]) #so we add the number of nobodyDays values prior to this section: this will correspond to the number of missing days so far  
 }  
 a<-i+1 #and adjust a so we know the row of diat where the next section starts.  
}

We now have the variable numdate, which has a number for each row indicating the number of days since Apr 29 2004. When no one participated on a given day (e.g. June 5th, 2004), the numbering accounts for that:

## Date numdate  
## 1 6/4/2004 36  
## 2 6/4/2004 36  
## 3 6/6/2004 38  
## 4 6/6/2004 38

# Age

This study started out by asking participants their age, and later switched to asking them for their month and year of birth. We want one variable that has everyone’s age in years. Let’s look at this data:

diat$datedate[which(!is.na(diat$age))[length(which(!is.na(diat$age)))]] #the last day on which we have numerical data for age:

## [1] "8/16/2016"

summary(diat$age)

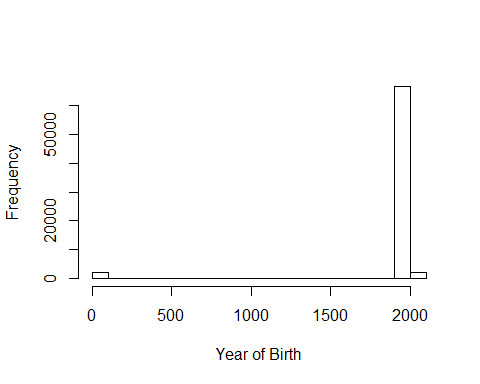
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 10.0 19.0 23.0 28.2 34.0 89.0 73646

summary(diat$birthyear)

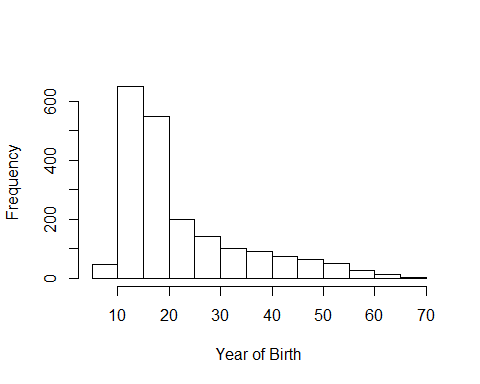
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 6 1978 1992 1932 1996 2009 248123

That birthyear data looks a little weird. It’s unlikely that we had any participants who were born in the year 6 A.D.

A histogram suggests that only a few people put very low numbers:



A second histogram of all “birthyear” values below 1900 shows that all of these are lower than 100.



Perhaps some people entered their age here; perhaps others were putting the month or day of their birth date, or answering at random. It’s impossible to know, and this is just a small fraction of our sample, so we’ll consider these individuals to have missing data for age.

diat$birthyear[which(diat$birthyear < 1900)] <- NA

Since we have birth month/year and current date for all participants after August 2016, we could calculate relatively precise ages for these participants. But to be consistent with the earlier participants, who listed their ages in years, we will estimate integer ages for these later participants. Those who participated after the 15th of their birthmonth in a given year will be assumed to have had their birthday that year.

agerows <- which(!is.na(diat$birthyear))  
for(dude in(agerows)){  
 if(is.na(diat$birthmonth[dude])){  
 if(diat$month[dude]<=6){  
 diat$age[dude] <- diat$year[dude] - diat$birthyear[dude] - 1  
 }  
 else if(diat$month[dude]>6){  
 diat$age[dude] <- diat$year[dude] - diat$birthyear[dude]  
 }  
 }  
 else if(diat$birthmonth[dude] > diat$month[dude]){ #if it's before their birthmonth  
 diat$age[dude] <- diat$year[dude] - diat$birthyear[dude] - 1   
 }  
 else if(diat$birthmonth[dude] < diat$month[dude]){ #if it's after their birthmonth  
 diat$age[dude] <- diat$year[dude] - diat$birthyear[dude]  
 }  
 else if(diat$birthmonth[dude] == diat$month[dude]){ #if it's their birthday month  
 if(diat$day[dude] <= 15){ #if it's before the 15th, assume they haven't had their birthday yet  
 diat$age[dude] <- diat$year[dude] - diat$birthyear[dude]-1  
 }  
 else{  
 diat$age[dude] <- diat$year[dude] - diat$birthyear[dude]  
 }  
 }  
}

Our new age variable looks like this:

summary(diat$age)

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 6.00 20.00 24.00 28.44 35.00 107.00 4747

But we’ll eliminate all participants below the age of 10 from our dataframe, because young children may have misunderstood questions and/or answered unreliably.

save<-diat  
diat<-diat[which(diat$age>=10 | is.na(diat$age)),]  
dim(save)

## [1] 319026 187

dim(diat[which(diat$age>=10 | is.na(diat$age)),])

## [1] 319008 187

We only lost 18 participants from this exclusion.

# Gender

The codebook for this survey tells us that gender was coded as a binary variable (male/female) for the majority of the study, with a change near the end of the study period to allow people to indicate a minority gender identity. It would be nice to use the more precise version from the end of the study period. We might be able to predict implicit prejudice more precisely if we had gender as (perhaps) a three-level variable: cisgender male vs. cisgender female vs. gender minority identities.

However, we want to include both gender and time as covariates when predicting D score, and we would have multicollinearity problems if the third level of gender only appeared during the last 16 months of the study. So unfortunately, in this analysis, my colleagues and I decided we would have to examine gender data only for people who identified as either male or female. (We did look at gender as a three-level variable in the supplemental materials to our publication, but we won’t get into that here.)

To do this, we need to transform responses to the second version of the question so that they’re compatible with responses to the first version of the question. This takes a little extra legwork because in the second version, participants were able to check multiple boxes.

First, we’ll create two vectors of potential responses to the second gender question. The responses in the “men” vector are those where the participant checked the “man” box and/or the “trans man” box, and did not check either the “woman” box or the “trans woman” box. The responses in the “women” vector are those where the participant checked the “woman” box and/or the “trans woman” box, and did not check either the “man” box or the “trans man” box.

men <- c("[1]", "[3]", "[1,3]", "[1,5]", "[3,5]", "[1,6]", "[3,6]", "[1,3,5]", "[1,3,6]")  
women <- c("[2]", "[4]", "[2,4]", "[2,5]", "[4,5]", "[2,6]", "[4,6]", "[2,4,5]", "[2,4,6]")

Next, we’ll use these vectors to add to the variable “Gen,” representing responses to the first gender question.

First, let’s identify the first row in diat that used the second version of the question, which is recorded in the column “genderidentity”.

min(which(!is.na(diat$genderidentity)))

## [1] 1

That’s odd. The first row of diat should have NA for genderidentity. Let’s look at the first few rows of genderidentity:

head(diat$genderidentity)

## [1] " " " " " " " " " " " "

That’s the problem: The missing data are saved as spaces instead of NAs.

In fact, when we look at a frequency table with all the values of genderidentity:

table(diat$genderidentity)

##   
## -999 . [1,2,3,4,5,6] [1,2,3,4,5]   
## 79 247567 1 6 1   
## [1,2,4,5] [1,2,5] [1,2] [1,3,5,6] [1,3,5]   
## 1 3 12 2 8   
## [1,3] [1,4] [1,5,6] [1,5] [1,6]   
## 50 2 3 33 13   
## [1] [2,3,5] [2,3] [2,4,5] [2,4]   
## 17892 1 1 2 25   
## [2,5,6] [2,5] [2,6] [2] [3,5,6]   
## 10 116 24 52256 3   
## [3,5] [3] [4,5] [4,6] [4]   
## 16 91 4 1 35   
## [5,6] [5] [6]   
## 20 643 87

There are multiple missing-data indicators, including -999, " “, and”."

We’ll recode these as NA.

diat$genderidentity <- car::recode(diat$genderidentity, "'-999' = NA; ' ' = NA; '.' = NA")

Now we should be able to identify the first row in diat that has a value for “genderidentity”, and use genderidentity and our “men” and “women” vectors to fill in data after that point.

min(which(!is.na(diat$genderidentity)))

## [1] 247567

diat$Gen[247567:nrow(diat)] <- sapply(diat$genderidentity[247567:nrow(diat)],   
 FUN = function(x){  
 if(x %in% men){-1}  
 else if(x %in% women){1}  
 else {NA}  
})  
  
table(diat$Gen) #-1 indicates men, 1 indicates women

##   
## -1 1   
## 82823 233320

# Educational Attainment

The education question did not change over the study period. It asked participants to identify their highest level of educational attainment, and offered 14 response options.

table(diat$edu)

##   
## 1 2 3 4 5 6 7 8 9 10 11 12   
## 518 3429 31994 22190 98386 27371 47490 35357 30223 3859 3423 6841   
## 13 14   
## 2366 3319

The first seven ranged from “some elementary school” to “4-year college degree,” and represent categories that are unambiguously listed in order of increasing education. However, options 8 through 14 represent various graduate and professional school experiences, and the relative “status” of the options is less obvious: for example, it’s not clear whether someone with a medical degree is more or less educated than someone with a PhD.

That doesn’t matter if we treat education as a categorical (non-ordinal) variable, which is what we’d want to do if the goal was to predict implicit prejudice as precisely as possible. However, our research question was whether increased level of education was related to level of implicit disability prejudice (i.e., do more educated people tend to be more or less prejudiced?). Anyone who’s been to graduate school is already at the top of the education distribution, and it is unlikely that additional education among graduate/professional school students will be strongly related to prejudice. So it may make the most sense to treat all post-undergraduate education as a single level of education:

diat$edu[which(diat$edu==-999)] <-NA #clean up places where missing values are coded as -999  
diat$edu2 <-car::recode(diat$edu, "1=1; 2=2; 3=3; 4=4; 5=5; 6=6; 7=7; 8=8; 9=8; 10=8; 11=8; 12=8; 13=8; 14=8")  
table(diat$edu2)

##   
## 1 2 3 4 5 6 7 8   
## 518 3429 31994 22190 98386 27371 47490 85388

# Disability Status

Our next predictor of interest is disability status. Participants were asked to indicate whether they themselves had a disability (however the participant chose to define that) and answered “yes” or “no.” Fortuneately, one of my collaborators has already put this in the format we want:

table(diat$Disab) #-1 means they don't have a disability; 1 means they do have a disability

##   
## -1 1   
## 260269 45606

So let’s move on to our final predictor.

# Contact with People with Disabilities

Contact was measured with a yes/no question along the lines of “Does a friend, family member, or close acquaintance have a disability?” However, the exact phrasing changed three times over the years, so responses are stored in four different variables:

table(diat$disabledfamilymem)

##   
## . no null yes   
## 81534 20744 129829 38 86863

table(diat$disabledfamilymem\_001) #1=yes, 2=no

##   
## 1 2   
## 41380 26825

table(diat$disabledfriend)

##   
## . no null yes   
## 81534 21335 117927 120 98092

table(diat$disabledsigother)

##   
## . no null yes   
## 291043 6504 9348 146 11967

We want a varible–we’ll call it “contactPWD”–that takes the value 1 if the participant responded yes to any of these questions or -1 if the participant responded no.

"contactPWD" %in% names(diat) #make sure we're not overwriting a preexisting variable

## [1] FALSE

diat$contactPWD <- NA #create variable  
  
diat$contactPWD[which(diat$disabledfamilymem=="no")] <- -1  
diat$contactPWD[which(diat$disabledfamilymem\_001==2)] <- -1  
diat$contactPWD[which(diat$disabledfriend=="no")] <- -1  
diat$contactPWD[which(diat$disabledsigother=="no")] <- -1  
  
diat$contactPWD[which(diat$disabledfamilymem=="yes")] <- 1  
diat$contactPWD[which(diat$disabledfamilymem\_001==1)] <- 1  
diat$contactPWD[which(diat$disabledfriend=="yes")] <- 1  
diat$contactPWD[which(diat$disabledsigother=="yes")] <- 1  
  
table(diat$contactPWD)

##   
## -1 1   
## 121846 184854

This concludes all the cleaning necessary to run our model of implicit prejudice. When I have time (eventually), I’ll continue adding to this github folder to include the analysis and visualization stages of this project.