Online Supplement:
Code for data preprocessing and statistical analyses

This document contains the complete R script used to download, preprocess, and analyze data for the associated paper by Peter L. Phalen in *Psychiatric Services*.

This R script is complete and can be run in order as is to reproduce all reported results, though the user must specify certain variables such as working directory. The script also outputs some figures not reported in the paper, such as demographics associated with different levels of psychological distress. Output is a single .txt file holding all results with annotations.

```
#download packages you don't have. First four are only needed for getting
#the data from NHIS in the first place
wants <- c("SAScii", "RCurl", "downloader", "digest", "survey", "mitools")
has   <- wants %in% rownames(installed.packages())
if(any(!has)) install.packages(wants[!has])

#load packages#
sapply(wants, require, character.only = TRUE)


# # # # # # # # # # # # # # # # # # # # #
# # block of code to get data from NHIS... warning: this can literally take all night. uncomment to run
# # Many thanks to Anthony Damico [http://www.asdfree.com/], some of whose code is used by
permission here and in
# # sections of the preprocessing components of this script
# # # # # # # # # # # # # # # # # # # # #
 options( encoding = "windows-1252" )    # only macintosh and *nix users need this line
   library(downloader)
   setwd( "/Users/Data analysis" )
   nhis.years.to.download <- c(2013,2014)  # choose the years that you want to get
   source_url( "https://raw.github.com/ajdamico/asdfree/master/
National%20Health%20Interview%20Survey/download%20all%20microdata.R" ,
            prompt = FALSE ,
            echo = TRUE )
# # # # # # # # # # # # # # #
# # end of auto-run block # #
# # # # # # # # # # # # # # #
```

```
#######################################################################################
####################
# Analyze the National Health Interview Survey personsx, samadult, and imputed income files with R #
#######################################################################################
####################




########################################
###       Begin user settings.      ##
###  (These variables are set by you. ##
###  The rest of the code should      ##
###   depend on these settings.)      ##
########################################


# set working directory
setwd( "/Users/Data analysis" )


# choose what year of data to analyze
# note: this can be changed to any year that has already been downloaded locally
# fair warning: I wrote this with reference to NHIS years 2009-2014. If you
# want to run this with earlier years of data you may need to adjust the preprocessing
# function below to account for relevant changes in variable names and formats

latestYear <- 2014        #2014 is currently the latest available dataset
comparedToYear <- 2013


## Minimum and maximum ages to include in dataset.
## Your choice here subsets down your models/stats.
## If you want to use the complete data, just set
## minimumAge to 0 and maximumAge to 999
##
## Note: minimumAge <= x < maximumAge

minimumAge <- 26
maximumAge <- 65


# use this as a 'greater than or equal to' cutoff for the K6
# to determine who counts as SMI. Most common cut-off is 13 for SMI,
# see Kessler et al 2010
# http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3659799/pdf/nihms447801.pdf

SMIthreshold <- 13
```

```
# Set this to the quarters you'd like to keep. Leave as c(1:4) or c(1,2,3,4) if you
# want the complete year's worth of data. Otherwise, choose subsets like c(2:4)
# This may be a good idea if you're analyzing the effect of the January 1, 2014
# healthcare expansion, as these may only be visible in later quarters.

quarters.to.keep <- c(2:4)


# Set updatePreprocessingFunction to TRUE only if you have just manually changed
# the preprocessNHISdata function and need to force it to update (e.g., if you've
# just gone into the function code below and changed how variables are calculated
# or formatted).
# Note: If this is a new R session, don't worry about this as the script
# will update the preprocessing function automatically whenever it's not presently
# declared in your Global Environment.

updatePreprocessingFunction <- TRUE



## Select the analysis variables that you're interested in
## You need to specify these or your analyses will take
## forever and/or your computer may crash

variables.to.keep <-
  c(
    ## survey variables: do not alter these
    "psu_p",   # (cluster)
    "strat_p", # (stratum)
    "wtfa_sa", # (weight)
    "intv_qrt", # (interview quarter [1-4])

    # merge variables: do not alter these
    "hhx" ,  # (household unique identifier)
    "fmx" ,  # (family unique identifier)
    "fpx" ,  # (person unique identifier)



    # analysis variables: you can change these
    "K6",          # K6 Serious Psychological Distress score, continuous, 0-24
    "SMI" ,        # K6 >= SMIthreshold versus everyone else. see http://www.ncbi.nlm.nih.gov/pmc/
articles/PMC3659799/pdf/nihms447801.pdf
    "SMIvNon" ,      # 1 = (K6 >= 13) 0 = (K6 < 5) excludes mod distress
    "distressLevel", # 0 = K6 >= 5 but less than 13, indicating moderate clinical distress. see http://
www.ncbi.nlm.nih.gov/pmc/articles/PMC3370145/
    "anyDistress", # K6 greater than or equal to 5
    "modDistress", # K6 >= 5 < 13
    "modDistressvNon", # 1 = (K6 >= 5 < 13) 0 = (K6 < 5) exclude SMI
    "YEAR",        # Survey year, 0 = 2013, 1 = 2014, recoded later
    "coverage",    # Health insurance coverage, 0 = Not covered at time of survey
    #                          1 = covered
    "povrati3",    # Earnings / poverty line, continous, native to dataset,
    # e.g., 1 = at poverty line, 1.33 = 133% poverty line
    "fine.povcat", # Poverty in four categories
    "sex",         # 1 = Male, 2 = Female (native to dataset)
    "receivedMHcare",     # Saw any mental health professional within last 12 months
    "WHITE",       # 0 = non-White, 1 = White
    "PublicInsurance",  # 1 = Covered by medicaid or medicare, 0 = Anyone else
    "Unemployment",   # level of unemployment, ordinal, we transform into binary chronic unemployment
variable with cutoff  >=2
    # 0 = Had job last week,
    # 1 = No job last week, had job past 12 months
    # 2 = No job last week, no job past 12 months,
    # 3 = Never worked
    "couldNotAffordMH",     #needed mental health care but couldn't afford it  1=Yes 2=No
    "diff2findplan", #how difficult has it been to find a plan you could afford 1 = Very difficult 2 =
somewhat difficult 3= not at all difficult
   "race", # 01 Hispanic
    # 02 Non-hispanic white
    # 03 Non-hispanic black
    # 04 Non-hispanic asian
    # 05 Non-hispanic all other
    "educCat" , # recode to
    #1 = No high school diploma;
    #2 = GED or High School;
    #3 = some college
    #4 = AA degree
    #5 = 4-year college
    #6 = Postgrad
    "ageCat" # recode to 1 = 24-34 2= 35-44 3 = 45-54, 4 = 55-64
  )

if (comparedToYear >= 2013){ #these variables only exist in the the post-2012 datasets

  variables.to.keep <- c(variables.to.keep,
"decentHCsatisfaction",     # 1 = Better 2 = Worse 3 = About the same, we binarize to 1 = Better, 0 =
anything else
"satisfiedWhealthcare", #how satisfied are you with healthcare you received in past 12 mo 1 = Very
satisfied 2 = somewhat satisfied 3 = somwhat dissatisfied 44 = very dissatisfied
"worryAbtCostHealthCatastrophe") # How worried are you right now about not being able to pay medical
costs of a serious illness or accident? Are you...
#1 = Very worried
#2 = moderate worried
#3 = not too worried
#4 = not worried at all
```

```
##recoded to >= 2 for moderately or very worried
}


########################################
###          End user settings.        ##
###     (The rest of the script can    ##
###           run on its own)          ##
########################################










# If you've already got the preprocessing function in your system and
# don't need to force it to update within a single session, this if-statement
# saves time by skipping the function declaration
if (!("preprocessNHISdata" %in% ls()) | (updatePreprocessingFunction == TRUE)){

  #########################################################
  #### Create function to perform data preprocessing. #####
  #########################################################
  #### This function will remain in your environment, #####
  #### so unless you need to change the recoding      #####
  #### scripts you can get away with running it just  #####
  #### once per session                               #####
  #########################################################

  preprocessNHISdata <- function(year, mostRecentData = TRUE){

    #if the user asks to process pre-2013 data,
    #don't stop them but throw a warning
    if (year < 2013){
      warning("This script is written primarily for years 2013 and 2014 of the NHIS (though 2011-2012
work for almost all variables). Earlier years typically use different variables or variable formats.")
    }


    if (mostRecentData == TRUE) {

      # For Pre-Post analyses, label the more recent dataset
      # with 'post'
      chronology <- "post"
      #dummy code for year
      post <- 1

    }else{
      if (mostRecentData == FALSE){

        # and the earlier dataset with 'pre'
        chronology <- "pre"

        #dummy code for year
        post <- 0

      }else{

        #If the user doesn't set mostRecentData to either T or F, throw an error
        stop("The mostRecentData argument must be set to either TRUE or FALSE.
             For Pre-Post analyses, set mostRecentData to FALSE if this is your Pre dataset,
             or TRUE if this is your Post dataset")
      }
      }

    # Throw an error and stop script if the user asked for a quarter greater than 4 or less than 1
    if (TRUE %in% c(quarters.to.keep > 4, quarters.to.keep < 1)){
      stop("You can only choose quarters between 1 and 4")
    }


    ########################################################
    ##########          Load data           ################
    ########################################################

    # set R to produce conservative standard errors instead of crashing
    # http://r-survey.r-forge.r-project.org/survey/exmample-lonely.html
    options( survey.lonely.psu = "adjust" )
    options(stringsAsFactors=FALSE)

    # construct the filepath (within the current working directory) to the three rda files
    path.to.personsx.file <- paste( getwd() , year , "personsx.rda" , sep = "/" )
    path.to.samadult.file <- paste( getwd() , year , "samadult.rda" , sep = "/" )
    path.to.incmimp.file <- paste( getwd() , year , "incmimp.rda" , sep = "/" )

    # Tell us which filepaths we're pulling from
    cat("Pulling from the following files:
        ",path.to.personsx.file,"
        ",path.to.samadult.file,"
        ",path.to.incmimp.file)

    # now the "NHIS.[year].personsx.df" data frame can be loaded directly
    # from your local hard drive.  this is much faster.
```

```
    load( path.to.personsx.file )


# this loads a data frame called NHIS.[year].personsx.df
    load( path.to.samadult.file )


# this loads a data frame called NHIS.[year].samadult.df
    # the five imputed income files will be loaded later


    # construct a string containing the data frame name of the personsx data table
    # stored within the R data file (.rda)
    # note: for 2014, this data frame will be named "NHIS.14.personsx.df"
    # but constructing it dynamically will allow analyses of other years
    # by simply changing the 'year' variable above
    df.name <- paste( "NHIS" , substr( year , 3 , 4 ) , "personsx" , "df" , sep = "." )

    # repeat this for the sample adult data frame,
    # but not for the five imputed income data frames (which will be dealt with later)
    samadult.name <- paste( "NHIS" , substr( year , 3 , 4 ) , "samadult" , "df" , sep = "." )

    # copy the personsx data frame to the variable x for easier analyses
    # (because NHIS.11.personsx.df is unwieldy to keep typing)
    x <- get( df.name )

    # copy the samadult data frame to the variable sa for easier typing
    # (because NHIS.11.samadult.df is unwieldy to keep typing)
    sa <- get( samadult.name )

    # remove the original copy of the two data frames from memory
    rm( list = c( df.name , samadult.name ) )

    # clear up RAM
    gc()


    #######################################################
    ##########       BEGIN DATA PREP       ################
    #######################################################

    # Status update
    cat("
       Merging files...")

    ####################################
    # merge personsx and samadult files #
    ####################################
```

```
# the personsx and samadult files are both at the individual or person-level
# (as opposed to family-level or household-level)
# so merging them together will require three variables:


# store the names of these three columns in a character vector
merge.vars <- c( "hhx" , "fmx" , "fpx" )

# these two files have multiple overlapping (redundant) columns,
# so determine which columns are included in both data frames
# at the same time, enclose this statement in () thereby printing the vector to the screen
( columns.in.both.dfs <- intersect( names( sa ) , names( x ) ) )


# since the merge.vars will be used to merge the two data frames,
# those three variables should be excluded from the list of redundant columns
# keep all column names that don't match the merge variables
# at the same time, enclose this statement in () thereby printing the vector to the screen
( redundant.columns <- columns.in.both.dfs[ !( columns.in.both.dfs %in% merge.vars ) ] )

# notice that the three merge.vars have disappeared


# most analyses start with the personsx file,
# so shave the redundant columns off of the samadult file
# keep all columns in the samadult file that are not in the redundant.columns vector
sa <- sa[ , !( names( sa ) %in% redundant.columns ) ]

# at this point, the only overlap between the personsx and samadult files
# should be the three merge.vars
# throw an error if that's not true
stopifnot( merge.vars == intersect( names( sa ) , names( x ) ) )


# remember that the samadult file contains a subset of the individuals in personsx
# therefore, an inner join of the two should have the same number of records as samadult

# perform the actual merge
x.sa <- merge( x , sa )
# note that the merge() function merges using all intersecting columns -

# uncomment this line to see intersecting columns
# intersect( names( sa ) , names( x ) )

# - by default, the 'by' parameter does not need to be specified
# for more detail about the merge function, type ?merge in the console

# throw an error if the number of records in the merged file
# does not match the number of records in the samadult file
```

```
  stopifnot( nrow( x.sa ) == nrow( sa ) )

  # if the user chose to restrict the number of quarters to include
  # in the analysis, cut those quarters out of the data set
  if ( length(quarters.to.keep)<4 ){
    x.sa <- x.sa[which(x.sa$intv_qrt %in% quarters.to.keep),]
  }


  #subset down to specified age ranges
  x.sa <- x.sa[which((x.sa$age_p >= minimumAge) & (x.sa$age_p < maximumAge) ),]

  # now the x.sa data frame contains all of the rows in the samadult file and
  # all columns from both the samadult and personsx files
  # therefore, there's no more need for the samadult file on its own
  # so delete the samadult file
  rm( sa )

  # and clear up RAM
  gc()


  # since the samadult file is a subset of the personsx file,
  # the personsx.design should always be used if it contains
  # all variables necessary for the analysis.  the sample size is larger.



  ########################################
  # prepare data frames for an analysis #
  # involving multiply-imputed income   #
  ########################################

  rm( x )

  # and immediately clear up RAM
  gc()


  # now load the imputed income data frames
  load( path.to.incmimp.file )


# this loads five data frames called ii1, ii2, ii3, ii4, and ii5


  # loop through all five imputed income files
  for ( i in 1:5 ){
```

```
# create a temporary current.i data frame
# containing the current iteration's (1 through 5) imputed income file
current.i <- get( paste0( "ii" , i ) )

# the 2014 imputed income merge fields are currently stored as character variables
# and should immediately be converted over to numeric types
merge.vars <- intersect( names( x.sa ) , names( current.i ) )

# loop through all variables used in the merge
# overwrite each column with itself, only converted to a numeric field
for ( j in merge.vars ) x.sa[ , j ] <- as.numeric( x.sa[ , j ] )


# a handy trick to view the class of all columns within a data frame at once:
# sapply( x.sa , class )

# merge the merged file with each of the five imputed income files
y <-
  merge(
    x.sa , # the samadult-personsx merged data frame
    current.i # ii1 - ii5, depending on the current iteration of this loop
  )

# and confirm the new data frame (merged + the current iteration of the multiply-imputed data)
# contains the same number of records as the original merged file
stopifnot( nrow( x.sa ) == nrow( y ) )


##############################
# START OF VARIABLE RECODING #
# any new variables that the user would like to create should be constructed here #

# Status update
loopNumber <- paste0("imputation (",i,"/5)")
cat("
    Recoding variables...",loopNumber)

##############################################################
# IMPORTANT NOTE: Some variable names for the National Health
# Interview Survey have changed over time. Others have changed
# slightly in terms of format. So, our calculations have to
# change depending upon the year of the survey.
##################################


# start with variables that haven't changed over the years
# recode them here
y <-
  transform(
        y ,
        #### Most variables have codes for things like "Refused to answer"
        #### This block of recodes is intended to set those kinds of codes
        #### to NA
        notcov = ifelse(notcov < 7, notcov, NA),
        receivedMHcare = ifelse(ahcsyr1 < 7, ahcsyr1, NA) ,
        educ1 = ifelse(educ1>22, NA, as.numeric(educ1)),
        educCat = ifelse(educ1 <= 12, 1, #no high school diploma
                  ifelse(educ1 == 13, 2, # GED
                     ifelse(educ1 == 14, 2, # HS lump with GED
                        ifelse(educ1 == 15, 3, # some college
                           ifelse(educ1 %in% c(16:17), 4, # AA
                              ifelse (educ1 == 18, 5, #BA
                                 ifelse(educ1 %in% c(19:21), 6, NA))))))), #graduate degree
        race = factor(hiscodi3,labels=c("Hispanic", "Non-H white", "Non-H black", "Non-H asian", "Non-
H other" ) )
      )

    y <-
      transform(
        y ,
        #Recode variables of interest as needed. Here we're converting variables with multiple levels into
binary variables
        coverage = factor( as.numeric(notcov==2), labels = c("Not covered now", "Covered now"),
levels=c(0,1),exclude=NA),
        receivedMHcare = factor(as.numeric(ahcsyr1==1), labels = c("No mh prof", "Saw mh prof"),
levels=c(0,1), exclude = NA),

        #these variables either don't have missing values, or the missing values are irrelevant to our
analyses
        YEAR = factor(post),
        WHITE = factor(as.numeric(hiscodi3 == 2), labels = c("NonWhite", "White"), exclude = NA),
        sex = factor(sex, labels=c("Male", "Female")),
        PublicInsurance = ifelse(medicaid <= 2, 1, ifelse( medicare <=2 ,1, 0)),
        couldNotAffordMH = ifelse(ahcafyr2 == 1, 1, ifelse( ahcafyr2 ==2 ,0, NA))
      )



    if (year >= 2013){
      y <-
        transform(
          y ,
          #### set all the "I don't know" / "refused to answer" codes to NA
          # these are the K6 components, which were renamed in 2013 for some
          # reason
          asisad = ifelse(asisad < 7, asisad, NA),
          asinerv = ifelse(asinerv < 7, asinerv, NA),
          asihopls = ifelse(asihopls < 7, asihopls, NA),
          asirstls = ifelse(asirstls < 7, asirstls, NA),
          asieffrt = ifelse(asieffrt < 7, asieffrt, NA),
          asiwthls = ifelse(asiwthls < 7, asiwthls, NA),
          decentHCsatisfaction = ifelse(ahicomp > 3, NA, ifelse(ahicomp == 1 , 1, 0))) #note this variable
does not exist for earlier years
    y <-
      transform(
        y ,
        K6 =
          # calculate K6 score. Variables must be reverse coded and
          # scaled from 0-24 to match the standard scoring.
          (5-asisad +
             5-asinerv +
             5-asihopls +
             5-asirstls +
             5-asieffrt +
             5-asiwthls),
        satisfiedWhealthcare = ifelse(asisathc == 1 | asisathc ==2, 1, ifelse(asisathc == 3 | asisathc == 4,
0,NA)), #how satisfied are you with healthcare you received in past 12 mo 1 = Very satisfied 2 =
somewhat satisfied 3 = somewhat dissatisfied 4 = very dissatisfied
        worryAbtCostHealthCatastrophe = ifelse(asimedc == 1 | asimedc ==2, 1, ifelse(asimedc == 3 |
asimedc == 4, 0,NA)) # How worried are you right now about not being able to pay medical costs of a
serious illness or accident? Are you... 1 = Very worried 2 = moderate worried 3 = not too worried 4 = not
worried at all

        )



    }
    if (year < 2013){

      y <-
        transform(
          y ,
          #### set all the "I don't know" / "refused to answer" codes to NA
          # these are the K6 components, which were renamed in 2013 for some
          # reason
          sad = ifelse(sad < 7, sad, NA),
          nervous = ifelse(nervous < 7, nervous, NA),
          restless = ifelse(restless < 7, restless, NA),
          hopeless = ifelse(hopeless < 7, hopeless, NA),
          effort = ifelse(effort < 7, effort, NA),
          worthls = ifelse(worthls < 7, worthls, NA)

        )

     y <- transform(
        y ,
        K6 = (5 - sad +
          5 - nervous +
          5 - restless +
          5 - hopeless +
          5 - effort +
          5 - worthls)
          # calculate K6 score. Variables must be reverse coded and
          # scaled from 0-24 to match the standard scoring.
          )


    }


    if (year >= 2013){
      y <-
        transform(
          y ,
    factor (satisfiedWhealthcare, labels = c("Unsatisfied","Satisfied"), levels = c(0,1)),
    factor (worryAbtCostHealthCatastrophe, labels = c("Unworried","Worried"), levels = c(0,1))

        )

    }


    y <-
      transform(
        y ,

        # code variable for "Serious Psychological Distress (SMI)" based on K6 score
        SMI = factor(as.numeric(K6 >= SMIthreshold), labels=c("No Serious Distress", "Serious Psych
Distress (SMI)"), levels=c(0,1),exclude=NA),
        distressLevel = ifelse(K6 >= SMIthreshold, 2, ifelse( (K6 >= 5 ) & (K6 < 13) ,1,
ifelse(K6<5,0,NA))), # 1 = SMI, 0 = moderate clinical distress, else NA http://www.ncbi.nlm.nih.gov/
pmc/articles/PMC3370145/x
        anyDistress = ifelse(K6 >= 5, 1, 0), # 1 = SMI, 0 = moderate clinical distress, else NA http://
www.ncbi.nlm.nih.gov/pmc/articles/PMC3370145/x
        modDistress = ifelse(K6 >=5 & K6 < 13,1,0),
        SMIvNon = ifelse(K6 >= 13, 1, ifelse(K6 < 5, 0, NA)),
        Unemployment = ifelse(wrklyr4 >= 7, NA, ifelse(wrklyr4 >= 2,1,0)),
        # 999999 is NA code. Also need to add appropriate decimal place
        povrati3 = ifelse(povrati3 == 999999, NA, povrati3)
      )



    ### now we can further recode some of the year-dependent variables that we just generated
    y <-
      transform(
        y ,
        # create a four-category poverty variable
        fine.povcat =
          cut(
            povrati3 ,
            c( -Inf , 1.38 , 2 , 4 , Inf ) ,
            labels = c( "<138%" , "138-200%" , "200-399%" , "400%+" )
          ),
        ageCat =
          cut(
            age_p ,
            c( -Inf , 35 , 45 , 55 , 65 ) ,
            labels = c( "26-34" , "35-44" , "45-54" , "55-65" )
          ),
        #label important variables
        modDistress = factor(modDistress, labels=c("No distress","Moderate distress"), levels=c(0,1)),
        modDistressvNon = ifelse( modDistress == 1 , 1, ifelse( (anyDistress == 0) , 0, NA)),
        SMIvNon = factor(SMIvNon, labels=c("No distress","SMI"), levels=c(0,1)),
        diff2findplan = ifelse(ainddif2 %in% c(7:9), NA, ifelse ( ainddif2 == 1 , 1 , 0))


      )

    if ( ! FALSE %in% (quarters.to.keep == c(2:4) ) ) {
      y <-
        transform(
          y ,

          wtfa_sa = wtfa_sa + (wtfa_sa * (1/3)) # reweight given quarter restriction. This only matters for
calculation of total counts

        )

    }


    # END OF VARIABLE RECODING #
    ############################

    # save the data frames as objects x1post - x5post (or x1pre -x5pre), depending
    # on the iteration in the loop.
    # at the same time restrict the resulting dataset to the variables of interest
    # specified above. If we don't do this the script takes much, much longer.
    assign( paste0(  'x' , i , chronology) , y[,variables.to.keep] )

    # delete the y and ii# data frames
    y <- NULL
    assign( paste0( "ii" , i ) , NULL )
```

```
    # garbage collection - free up RAM from recently-deleted data tables
    gc()
  }

  #clean up intermediary data
  rm(current.i)

  # To minimize processing time for analyses that *don't* require multiply imputed income
  # we want to have a dataset available that has all the recoded variables
  unimputedDataName <- paste0("unimputed",chronology)
  assign(unimputedDataName, get(paste0("x1",chronology)))

  #create a list holding multiple imputations
  output <- list(
    get(paste0(  'x1' , chronology)),
    get(paste0(  'x2' , chronology)),
    get(paste0(  'x3' , chronology)),
    get(paste0(  'x4' , chronology)),
    get(paste0(  'x5' , chronology))
  )

  #tack on the unimputed list
  output <- list(output, get(unimputedDataName) )

  # ouptut[[1]] holds the 5 imputed datasets
  # output[[2]] hold the unimputed dataset
  return(output)

    }
}

#####################################################################################
  ######### end of prepocessing function ##############################################

#####################################################################################








# use the function we just defined to pull lists of processed data.
# these datasets have been restricted to the variables.to.keep that
# you specified earlier
```

```
preOutput <- preprocessNHISdata(comparedToYear, mostRecentData=FALSE) # 'Pre' year
postOutput <- preprocessNHISdata(latestYear, mostRecentData=TRUE)     # 'Post' year


# get the list of imputed files for Pre and Post years
for (i in 1:5){
  assign(paste0("x",i,"pre"), preOutput[[1]][[i]])
  assign(paste0("x",i,"post"), postOutput[[1]][[i]])
}

#get a set for non-imputed analyses of the PRE year
preUnimputed <- preOutput[[2]]
postUnimputed <- postOutput[[2]]

#delete the stripped outputs
rm(preOutput)
rm(postOutput)

#clear up RAM
gc()



#create survey design object for all the Post analyses that don't require info on income
psa.Post <-
  svydesign(
    id = ~psu_p ,
    strata = ~strat_p ,
    nest = TRUE ,
    weights = ~wtfa_sa,
    data = postUnimputed
  )

#create multiply imputed survey design object for Post analyses that require income info
psa.impPost <-
  svydesign(
    id = ~psu_p ,
    strata = ~strat_p ,
    nest = TRUE ,
    weights = ~wtfa_sa,
    data = imputationList(list(x1post,
                          x2post,
                          x3post,
                          x4post,
                          x5post)
    )
  )
```

```
#create survey design object for unimputed 'Pre' analyses
psa.Pre <-
  svydesign(
    id = ~psu_p ,
    strata = ~strat_p ,
    nest = TRUE ,
    weights = ~wtfa_sa,
    data = preUnimputed
  )

#create multiply imputed survey design object for income-related 'Pre' analyses
psa.impPre <-
  svydesign(
    id = ~psu_p ,
    strata = ~strat_p ,
    nest = TRUE ,
    weights = ~wtfa_sa,
    data = imputationList(list(x1pre,
                               x2pre,
                               x3pre,
                               x4pre,
                               x5pre) )
  )


#create Pre-Post survey design object for non-income 'Pre' analyses
psa.noImp <-
  svydesign(
    id = ~psu_p ,
    strata = ~strat_p ,
    nest = TRUE ,
    weights = ~wtfa_sa,
    data = rbind(preUnimputed,postUnimputed) #combine unimputed pre and post datasets together
  )

rm(preUnimputed,postUnimputed) # we don't need these datasets anymore

for ( i in 1:5 ){
  # For each of the five imputed dataset, collapse Pre-Post for multi-year analyses
  assign(paste0("x",i), rbind(
    get( paste0( "x" , i, "pre" ) ),
    get( paste0( "x" , i, "post" ) )
  )
  )
  # delete the component datasets after they've been collapsed
  assign(paste0( "x" , i, "pre" ), NULL)
  assign(paste0( "x" , i, "post" ), NULL)
}
```

```
# clear up RAM
gc()


## Create multiply imputed survey design object for Pre-Post analyses
psa.imp <-
  svydesign(
    id = ~psu_p , # cluster ids
    strata = ~strat_p , # stratification levels
    nest = TRUE , # stratification is nested
    weights = ~wtfa_sa,

# weights
    data = imputationList( list( x1 , # income imputations
                             x2 ,
                             x3 ,
                             x4 ,
                             x5 ) )
  )

#erase the 5 component datasets which won't be needed anymore
for (i in 1:5) { assign(paste0( "x" , i ), NULL)}
gc() #clear up ram




###########################
##### END OF DATA PREP #####
###########################


## Create a "sink" txt file that we can open at the end to show us all our results in a more readable way
#  Skip this code if you don't want to run all of the analyses in this script at once
sinkFile <- paste(getwd(),"/TempOutput", substr( comparedToYear , 3 , 4 ),"-", substr( latestYear , 3 ,
4 ),".txt", sep="")
sink(sinkFile, type="output")


## We use the cat() function to annotate the sink output
cat("
    ################################################################################
    ######## This is the auto-generated output of R code written to display apparent
    ######## health disparities between people with/without psychological distress
    ######## in the United States.
    ########
    ######## Dataset: National Health Interview Survey (NHIS; http://www.cdc.gov/nchs/nhis.htm)
    ########        Representative household survey; complex sample design.
    ###########################
    ",latestYear,"compared with", comparedToYear,"
    Minimum age for inclusion: ", minimumAge,"
    Maximum age for inclusion: ", maximumAge,"

    ")
if (length(quarters.to.keep) < 4 ){
  cat("
     You chose to restrict your dataset to these survey quarters: ", quarters.to.keep,"

     ")
}

cat("

    ################################################################################

    Percent (%) of the USA population with and without psychological distress

    The \"mean\" columns hold percentages. E.g., 0.30 = 30% of the corresponding population.

    ###### in",latestYear,"

    ")

cat("

    no distress

    ")

# note: because the "anyDistress" variable gives you a 1 if you have moderate or serious
#       psychological distress and a 0 otherwise, "1-anyDistress" is equal to no or low distress
1 - svyby(~anyDistress, ~YEAR, svymean, vartype="ci", design = psa.Post, na.rm=T)
psa.Post.noDistress <- subset(psa.Post, !is.na(anyDistress) & anyDistress== 0 )
svyby(~anyDistress, ~YEAR, svymean, design = psa.Post.noDistress, na.rm=T)
rm(psa.Post.noDistress)



cat("

    moderate distress

    ")


svyby(~modDistress, ~YEAR, svymean, vartype="ci", design = psa.Post, na.rm=T)
psa.Post.modDistress <- subset(psa.Post, modDistress == "Moderate distress" )
```

```
svyby(~modDistress, ~YEAR, unwtd.count, design = psa.Post.modDistress, na.rm=T)
rm(psa.Post.modDistress)

cat("

  serious distress

  ")


svyby(~SMI, ~YEAR, svymean, vartype="ci", design = psa.Post, na.rm=T)
psa.Post.SMI <- subset(psa.Post, SMI == "Serious Psych Distress (SMI)" )
svyby(~SMI, ~YEAR, unwtd.count, design = psa.Post.SMI, na.rm=T)
rm(psa.Post.SMI)

blankSpace <- function(){
 cat("

    ")
}


#this function tells us about rates of psychological distress in chosen demographic groups
calculateDistressDemographics <- function(outcomes = NULL, currentYearDesign = psa.Post){

 if (is.null(outcomes)){
   stop("outcomes argument is null. Please enter outcome variables as strings.")
 }else{
   for (i in outcomes) {



    cat("

      #############
      ######Population-level percentages for the following outcome variable:",i,"

      no distress (", i,")

      ")

    print(svyby(~as.character(anyDistress),~as.character(get(i)),svymean, vartype="ci",design =
currentYearDesign, na.rm=T))

    cat("

      moderate distress (", i,")

      ")
      print(svyby(~as.character(modDistress),~as.character(get(i)),svymean, vartype="ci",design =
currentYearDesign, na.rm=T))


      cat("

          serious distress (", i,")

          ")


      print(svyby(~as.character(SMI),~as.character(get(i)),svymean, vartype="ci",design =
currentYearDesign, na.rm=T))


    }
  }
}



calculateDistressDemographics(outcomes= c("race",
                                          "sex",
                                          "Unemployment",
                                          "fine.povcat",
                                          "educCat",
                                          "ageCat"))




calculateDistressRatesAndChanges <- function(outcomes = NULL, currentYearDesign = psa.Post,
multiYearDesign=psa.imp){

  if (is.null(outcomes) | FALSE %in% is.character(sapply(t,class))){
    stop("'outcomes' argument is invalid. Please enter outcome variables as strings.")
  }else{
    for (i in outcomes) {



      cat("

          #############
          ######Population-level percentages for the following outcome variable:",i,"

          All
```

```
        ")

    print(svyby(~as.character(get(i)),~YEAR,svymean, vartype="ci",design = currentYearDesign,
na.rm=T))

    blankSpace()

    glmFunctionNoInt <- paste("as.numeric(",i,")~YEAR")
    summary(MIcombine( with( psa.imp ,
                    svyglm(glmFunctionNoInt)
    )
    )
    )



    cat("

        no distress (", i,")

        ")

    print(svyby(~as.character(get(i)),~as.character(anyDistress),svymean, vartype="ci",design =
currentYearDesign, na.rm=T))

    blankSpace()

    glmFunction <- paste("as.numeric(",i,")~as.character(anyDistress) + YEAR +
as.character(anyDistress):YEAR")
    summary(MIcombine( with( multiYearDesign ,
                    svyglm(glmFunction)
    )
    )
    )

    cat("

        moderate distress (", i,")

        ")


    print(svyby(~as.character(get(i)),~as.character(modDistress),svymean, vartype="ci",design =
currentYearDesign, na.rm=T))

    blankSpace()

    psa.imp.modDistress <- subset(multiYearDesign, !is.na(modDistress) & modDistress == "Moderate
distress" )

    summary(MIcombine( with( psa.imp.modDistress ,
                    svyglm(glmFunctionNoInt)
    )
    )
    )


    cat("

        serious distress (", i,")

        ")


    print(svyby(~as.character(get(i)),~as.character(SMI),svymean, vartype="ci",design =
currentYearDesign, na.rm=T))

    blankSpace()

    psa.imp.SMI <- subset(multiYearDesign, !is.na(SMI) & SMI == "Serious Psych Distress (SMI)" )
    summary(MIcombine( with( psa.imp.SMI ,
                    svyglm( glmFunctionNoInt)
    )
    )
    )

  }
  rm(psa.imp.modDistress)
  rm(psa.imp.SMI)
}
}



cat("

############################################
###########    FOR YEAR",latestYear,"    ###########
############################################

  ")
## calculate distress rates and changes for the indicated variables
calculateDistressRatesAndChanges(outcomes = c("coverage",
                                   "PublicInsurance",
                                   "diff2findplan",
                                   "receivedMHcare",
                                   "couldNotAffordMH",
                                    "satisfiedWhealthcare",

                                    "worryAbtCostHealthCatastrophe",
                                    "decentHCsatisfaction"
                                    )
                          )

cat("

############################################
###########    FOR YEAR",comparedToYear,"    ###########
############################################

  ")

calculateDistressRatesAndChanges(outcomes = c("coverage",
                                    "PublicInsurance",
                                    "diff2findplan",
                                    "receivedMHcare",
                                    "couldNotAffordMH",
                                    "satisfiedWhealthcare",
                                    "worryAbtCostHealthCatastrophe",
                                    "decentHCsatisfaction"),
                                    currentYearDesign=psa.Pre)


rm(psa.Pre)
rm(psa.Post)
rm(psa.impPre)


#this function calculates differences in differences for chosen variables
calculateDIDs <- function(outcomes = NULL, design=psa.imp){

  if (is.null(outcomes)){
    stop("outcomes argument is null. Please enter outcome variables as strings.")
  }else{
    for (i in outcomes) {

      glmBaseFormula <- paste("as.numeric(",i,") ~",
                        "as.numeric(fine.povcat)",
                        "+ as.character(educCat)",
                        "+ as.character(WHITE)",
                        "+ as.character(sex)",
                        "+ as.character(ageCat)",
                        "+ Unemployment",sep=" ")

      MODglmFormula <- paste(glmBaseFormula,
                       "+ as.character(modDistressvNon)",
                       "+ as.character(YEAR)",
                       "+ as.character(modDistressvNon):as.character(YEAR)", sep=" ")

      SMIglmFormula <- paste(glmBaseFormula,
                       "+ as.character(SMIvNon)",
                       "+ as.character(YEAR)",
                       "+ as.character(SMIvNon):as.character(YEAR)", sep=" ")

      cat( "
          ####### DIDs for ",i,"
          Controlling for income,
          education,
          race (white v. non-white),
          gender,
          age,
          unemployment (>=12 months v. not)

          Differences in differences for moderately distressed people on:",i,"

          ")

      summary(MIcombine( with( design ,
                       svyglm(MODglmFormula))))

      cat( "

          Differences in differences for severely distressed people on:",i,"

          ")

      summary(MIcombine( with( design ,
                       svyglm(SMIglmFormula))))

     }
  }
}


cat("
     ##############################################################################


     ####################
     #### Differences in differences (DIDs) for health insurance coverage (distress v. no-distress).
     #### The interaction effect (anyDistress:YEAR) tells us how much the disparity between distressed
     #### and non-distressed people improved or worsened during the intervening time. In this case,
     #### a positive interaction between anyDistress and YEAR would indicate that the health
     #### insurance disparity is decreasing.
     ####
     #### Note: only the interaction effect is interpretable in these DIDs because the
     #### other variables aren't mean-centered.
     #######


     ")


#calculate differences-in-differences between 2013 and 2014 on the following variables
calculateDIDs(outcomes = c("couldNotAffordMH",
                  "receivedMHcare",
                  "coverage",
                  "worryAbtCostHealthCatastrophe",
                  "satisfiedWhealthcare",
                  "decentHCsatisfaction",
                  "PublicInsurance",
                  "diff2findplan"))



cat("

     ########################################################################
     #### adjusted odds ratios for having serious psychological distress ####
     ########################################################################

     ## controlling for race, chronic unemployment, income, education, age, and gender


     ")
summary(MIcombine( with( psa.impPost ,
                svyglm(SMIvNon~
                       C(race, base = 2 ) # change reference category to non-hispanic white
                       + as.character(Unemployment)
                       + as.character(fine.povcat)
                       + as.character(educCat)
                       + as.character(ageCat)
                       + as.character(sex),
                       family=quasibinomial(link="logit"))
)
)
)
cat("

     ########################################################################
     #### adjusted odds ratios for having moderate psychological distress ###
     ########################################################################

     ")
```

```
summary(MIcombine( with( psa.impPost ,
                svyglm(modDistressvNon~
                      C(race, base = 2 ) # change reference category to non-hispanic white
                     + as.character(fine.povcat)
                     + as.character(Unemployment)
                     + as.character(educCat)
                     + as.character(ageCat)
                     + as.character(sex),
                     family=quasibinomial(link="logit"))
)
)
)

cat("


  ##########################################################################
  #### adjusted odds ratios for having no or low psychological distress ##
  ##########################################################################

  ")

summary(MIcombine( with( psa.impPost ,
                svyglm((1-anyDistress)~
                      C(race, base = 2 ) # change reference category to non-hispanic white
                     + as.character(fine.povcat)
                     + as.character(Unemployment)
                     + as.character(educCat)
                     + as.character(ageCat)
                     + as.character(sex),
                     family=quasibinomial(link="logit"))
)
)
)

# remove the sink
sink()
# automatically open the sink file to see the output of your analyses
file.show(sinkFile)
#tell us the file location for the outputs
paste("Analysis results saved to:",sinkFile)
#clear RAM
gc()



# Erase all the objects we created earlier except the preprocessing function, which can be reused
rm(list = ls()[!(ls() %in% "preprocessNHISdata")])
```