```
#P.O. 841
#10/15/14
#Lab 6: Working with Existing Data in R

#Last week we (very briefly) introduced features of R that are available
 through additional packages. Packages have to be downloaded and installed
 once on your computer, and then loaded at each new session (using the
 "library" command). Let's install and load the packages 'foreign' and
 'TeachingDemos'.

#You only have to do the following once. You need to select a server
 ('mirror') to download from; choose the one closest to you (e.g., USA PA
 1).

install.packages('foreign')
install.packages('TeachingDemos')

#You do the following every time you use the package:

library(foreign)
library(TeachingDemos)

#To read more about what's available in a package, run help.start(), and
 click on 'Packages':

help.start()

#External data files may be available in a few formats or only one format,
 depending. More recent data will probably be available in either Stata,
 SPSS, Excel, or various standard text formats, while older data may only
 be available in standard text formats (e.g., tab delimited, fixed-width
 format). I'm going to show you the "read" commands for only the most
 common types of formats, but keep in mind that reading other formats into
 R might require a few additional lines of code -- you may need to specify
 things like how missing data is represented, etc. You can learn more about
 it here (or via a Google search):

?read.table
?read.fwf

#Now, let's read in a file in Stata format (.dta), the 2002 National
 Election Study. You can read in files by specifying the full file path.
 Use forward slashes, '/', not Windows-style backslashes! The
 convert.factor=F tells R not to load the data dictionary from the original
```

```
file.

#For Mac users, if you have Dropbox in the default directory:

NES<-read.dta('/Users/claireleavitt/Dropbox/po841/03740-0001-Data.dta',
 convert.factor=F)

#For PC users, if you have Dropbox in the default directory:

NES<-read.dta('~/.../Dropbox/po841/03740-0001-Data.dta', convert.factor=F)


#Alternatively, you can just browse for the file yourself using:

NES<-read.dta(file.choose(),convert.factor=F)

#And to simplify your life, you can set the working directory and refer
 directly to filenames in the future. Just specify (after /Users) your user
 name and folders.

setwd("/Users/claireleavitt/Dropbox/po841") #Mac
setwd("C:/Users/claireleavitt/Dropbox/po841") #Windows

#Now we could do:

NES<-read.dta('03740-0001-Data.dta',convert.factor=F)

#To load Excel files, the easiest thing to do is to save the file as CSV in
 Excel and then use read.csv:

mozaffar<-read.csv('/Users/claireleavitt/Dropbox/po841/mozaffar.csv')

#OR:

mozaffar<-read.csv(file.choose()) #Leave out the convert.factor=F command
 this time (not necessary for CSV files).


#Now let's inspect our data:

dim(NES) #731 columns (variables), 1511 rows (responses)
dim(mozaffar) #20 columns (variables), 62 rows

names(NES)
```

```
names(mozaffar)

head(NES)
head(mozaffar)

#NOTE: All of the "read" commands create data.frames, as you can see:

class(NES)
class(mozaffar)


#The 2002 NES interviewed respondents both before and after the 2002
 election, but some dropped out. Let's see how many completed both
 interviews.

#Look in the codebook to find the variable you want (either scan through or
 do a search to see that the variable we want -- the one that tells us
 whether the respondent participated in just the pre-election interview or
 both the pre-and-post-election interviews -- is named V021002). The
 codebook will also tell you the meaning of the data: 1 means pre-interview
 only (participant dropped out), while 2 means both interviews.

#We can change the names of the variables in the data set if we want to.
 Let's change the name of this variable:

which(colnames(NES)=="V021002") #Which column number in the NES data set is
 V021002?
colnames(NES)[7]<-'PreorPost' #Change the name of column 7

#OR

colnames(NES)[colnames(NES)=='V021002']<-'PreorPost'

summary(NES$PreorPost)

#Let's arrange the data in table format for better exposition:

table(NES$PreorPost)

#Let's say we ONLY want to analyze the responses from participants who
 completed both interviews. Let's create a new dataset, a subset of the
 original, containing only those respondents who did both interviews:

NESboth<-NES[NES$PreorPost==2,] #Tell R that for data set NES, we want to
```

```
isolate only the values of variable NES$PreorPost that = 2, as well as all
corresponding columns

dim(NESboth) #1,346 rows, 731 columns

#Compare with:
dim(NES) #1,511 rows, 731 columns

#Respondents are asked both before and after the election how they felt
about George W. Bush, on a 0 to 100 scale. We can examine these variables
using the summary() command:

#From the codebook, we can see that variable V023010 is the George W. Bush
feeling thermometer (asked BEFORE the election). Let's change the name of
the variable:

colnames(NESboth)[colnames(NESboth)=='V023010']<-'PreElectionBushFT'
summary(NESboth$PreElectionBushFT)

#From the codebook, we can see that variable V025043 is the George W. Bush
feeling thermometer (asked after the election). Let's change the name and
look at the variable:

colnames(NESboth)[colnames(NESboth)=='V025043']<-'PostElectionBushFT'
summary(NESboth$PostElectionBushFT)

#The codebook also tells us that missing data (respondents who refused to
answer, didn't know who GWB was, etc.) are coded 887, 888, 889, and 999.
Let's recode all those non-responses to NA. We can use either the ifelse
function that we learned last week, or the recode function in the "car"
package:

bush1<-ifelse(NESboth$PreElectionBushFT %in% c(887,888,889,999), NA,
NESboth$PreElectionBushFT)
#Tells R: If the values for NESboth$PreElectionBushFT are within the
specified vector, change to NA; if not, leave the values as they are

bush2<-ifelse(NESboth$PostElectionBushFT %in% c(887,888,889,999), NA,
NESboth$PostElectionBushFT)

#OR:

library(car)
gwb1<-recode(NESboth$PreElectionBushFT, '887:889=NA; 999=NA')
```

```
gwb2<-recode(NESboth$PostElectionBushFT, '887:889=NA; 999=NA')

summary(bush1)
summary(bush2)

summary(gwb1)
summary(gwb2)

#Note: once you have NAs in a vector, matrix, etc., some R functions
 require you to tell it what to do with the NAs:

mean(bush1) #You'll get an NA response
mean(bush1,na.rm=T) #Tell R to remove the NAs before computing the
 function. (Notice that the summary() function also gives you the mean,
 rounded to the nearest hundredth.)


#Now let's say we want to figure out how many people changed their opinion
 of Bush from before to after the election. Let's subtract the pre-election
 FT scores for all Xi from the post-election FT scores for all Xi

change<-bush2-bush1
table(change)
#Here, we see how many people's opinions dropped by Y points (- values),
 how many people's opinions stayed the same (difference=0) and how many
 people's opinions rose by Y points (+ values).

#Now let's test the null hypothesis that there was NO CHANGE in opinion of
 GWB from pre to post election, versus the alternative hypothesis that
 there WAS a change (maybe because the campaign had an effect). To do that,
 we might use the z.test() function, which I'm not going to demonstrate
 since that's your HW assignment. To use z.test or read the help file, you
 have to load the TeachingDemos package, which we already did.


#Now let's say we want to rearrange our data so that instead of giving us
 the number of points by which opinions of GWB fell and rose, we just want
 to look at our data categorically. Let's create three categories for
 analysis: those whose opinions became more positive, those whose opinions
 became less positive, and those for whom there was no change. We'll leave
 the zeros as zero, and make all the positive numbers 1 and all the
 negative numbers -1 (by dividing change by its absolute value).

change2<-ifelse(change==0,0,change/abs(change)) #If a value for "change" =
```

```
0, leave it at 0; all else, change to "the number divided by the absolute
value of the number", which will give us either 1 or -1.

tab1<-table(change2)
tab1
#See that 386 people's opinions of GWB fell post-election, 493 people's
opinions rose, and 460 people's opinions stayed the same.


#We can also use the recode function to do the exact same thing:

change3<-recode(change, '0="0"; -100:-1="-1"; 1:100="1"')
table(change3)

#Let's assign row names and column names to our table for clearer
exposition:

names(tab1)<-c('Opinion Fell','No Change','Opinion Rose')
tab1

#Let's examine the results in a pie chart:

pie((tab1),main='Change in Approval of GWB')

#You could also insert the labels "manually" if you hadn't renamed your
table:
pie((tab1), main='Change in Approval of GWB', labels=names(tab1))

#*********************************************************************

#Maybe a respondent's party ID has something to do with whether or not they
changed their opinions of GWB. Let's inspect the party-ID variable (check
the codebook for the variable name):

table(NESboth$V023036)

#We can see that a response of 1=Democrat, 2=Republican, 3=Independent,
4=Other Party; and 5="No preference". Every other response either means
NA, don't know, or refused to answer, so let's code all those useless
responses as NA under a new variable name that we'll call "PartyID":

#Recode 0, 8, and 9 as NA and create a new variable:

partyID<-ifelse(NESboth$V023036 %in% c(0,8,9),NA,NESboth$V023036)
```

```
table(partyID)

#OR:

partyID2<-recode(NESboth$V023036, '0=NA; 8=NA; 9=NA')
table(partyID2)

#We can also make a cross-tabulation using the table command, which will
 tell us how many of the people whose opinion of GWB dropped post-election
 were Dems, how many were Repubs, etc.:

tab2<-table(partyID,change2)
tab2

#Let's assign row names and column names for clearer exposition:

rownames(tab2)<-c('Dem','Rep','Indep','Other','None')
colnames(tab2)<-c('Opinion Fell','No Change','Opinion Rose')
tab2

#But it's more interesting to look at PROPORTIONS, i.e., what PROPORTION of
 Democrats like GWB less, what proportion like him more and what proportion
 didn't change their opinion:

tab3<-prop.table(tab2,1) #margin=1 for row, margin=2 for column
tab3

#Remember, we use the margin=1 command to tell R to organize the table by
 row proportion. That is, if we want to know the proportion of each row
 (party ID) that falls into each column (opinion change), we should specify
 margin=1. If we wanted to know the proportion of each column that falls
 into each row, i.e., the proportion of people who like GWB less that fall
 into each of the party ID categories, then we'll specify margin=2.

#Compare:
tab4<-prop.table(tab2,2)
tab4

#Now let's make a barplot from tab4, where we display the breakdown by
 party affiliation of the three categories: "Opinion Fell," "No Change" and
 "Opinion Rose"

bp<-barplot(tab4, main="Change in GWB Opinion by Party ID", xlab="Change in
 Opinion of GWB", col=c("darkblue","red","gray",
```

```
 "green","black"),legend=rownames(tab4))
bp

#If you want to make a horizontal barplot:
bphoriz<-barplot(tab4, main="Change in GWB Opinion by Party ID",
 ylab="Change in Opinion of GWB", col=c("darkblue","red","gray",
 "green","black"),legend=rownames(tab4), horiz=TRUE)
bphoriz

#There are a zillion different commands for dressing up bar plots, making
 them more comprehensive, etc. It can be fun to play around with these;
 see:
?barplot

# ------------------------------
# SAVING AND EXPORTING DATA FILES
# ------------------------------

#NOTE: I'm going to run these commands, but please don't run them on your
 own laptops, because if everyone runs them right, there will be 20
 identical copies of the files in the shared DropBox folder. To run them
 later yourself, just reset your working directory to wherever you want the
 data to be saved on your hard drive and then run the commands.

#To export data from R, we use commands that start with 'write'.  To export
 our revised NES dataset as a Stata file, we would do:

write.dta(NESboth,'NESboth.dta')

#To export tables as a file that can be opened in Excel, use write.csv:

write.csv(tab4,'table1.csv')

#To save the revised NSA dataset as an R file, we would do:

save(NESboth, file='NESboth.RData')

#To save everything that we've done in this session, we would do:

save.image(file='lab6.RData')

#To open up any of these R files in a new session, you can either click on
 them, or use the load() command:
```

```
load('NESboth.RData')
load('lab6.RData')

#If you do this, remember you first have to set the working directory using
 setwd(), or else you have to specify the file location.
```