Data Management in R (Handout 1)

Before beginning ***Exploratory Data Analysis*** and ***Statistical Inference***, data must be cleaned up and subsets created to address specific research questions and associations.

The following conventions are used in this document:

- Bold font indicates R code and functions that should be typed exactly
- Non-bold font shows code or text that should be replaced with user-supplied values
- This document calls functions in the statistics package descr which can be loaded with the **library()** function.

1. The Basics

A. Setting the working directory and loading data objects

```
# It can be convenient to set a working directory for the project; this is a folder in which all R objects
# for the project will be stored (.RDATA, .R, .RMD, etc..).
# For students using Ashesi computers, the working directory for the project should be on your
# One Drive so that files can be accessed from any Ashesi computer

setwd("C:/filepath")                    #substitute the file path for your project folder

load("datafile.RData")                  #if datafiles are saved in the project folder, a file path
                                        #may not be needed

# If you have additional data you will use for your project in a tab delimeted or CSV file, then it can be
loaded with the read.delim() function. It's best to use the import icon on the environment tab and
then copy and past the read.delim() code into your RMarkdown or script file.
```

B. Selecting the variables you want to examine

```
vars <- c("VAR1", "VAR2", "VAR3", "VAR4", "VAR5", "VAR6", "VAR7", "VAR8")
my_data <- dataframe[vars]

# VAR1, VAR2 etc… are the variable labels specified in the codebook
```

C. Saving your data subsets and variables

```
# save one or more R objects to a .RDATA file in the working directory
save(my_data, file="my_data.RDATA")

# alternatively you can save everything in your workspace to the working directory
save.image("myProject.RDATA")

# save data as a tab delimited text file if you want to open it in Excel:
write.table(my_data, file = "filename.txt", sep = "\t", row.names = FALSE)
```

2. Data Management Basics

A. Logical operators

equal to	greater than or equal to	less than or equal to	greater than	less than	not equal to	x or y	x and y
==	>=	<=	>	<	!=	x | y	x & y

B. Identify and recode error codes and missing data

R uses “NA” to identify missing data. You will have to reassign error codes to missing data, e.g. “NA” , so that R doesn’t treat them as real/meaningful values and include them in analysis and plots.

```
# First run the freq() function to identify the error codes, though you should know what
# to expect from the codebook. After assigning NA to the error codes, run the freq() function
# again to verify that the process worked as expected.

freq(my_data$VAR1)
my_data$VAR1[my_data$VAR1 == 9 ] <- NA
freq(my_data$VAR1)

# In the above example, the codebook indicated that non-response was coded with the number 9
# and this was verified when running freq().
# Note: you should have already run library(descr) to load the descriptive statistics package.
```

C. Sub-setting Data

When using large data sets, it is often necessary to subset the data so that you are including only those ***observations*** that can assist in answering your particular research question. For example, if you are interested in identifying demographic predictors of mosquito net use among rural households, you would first subset the data to include only respondents from rural households.

Sub-setting can be accomplished in two ways, through indexing with square brackets [ ] or by using the subset() function.

```
# Note: indexing is by [row,col].
# When using [row, ] with a blank after the comma, ALL columns associated with row criteria are
# included
rural_subset <- my_data[my_data$URBRUR == 2, ]

# using the subset() function
rural_subset <- subset(my_data, COUNTRY == 11 | COUNTRY == 22)


# Note that logical operators may be combined. In the example below, rural respondents older
# than 15 are selected.
rural_subset <- subset(my_data, URBRUR == 2 & Q1 > 15)
```

D. Recode responses to “no” based on skip patterns

There are a number of legitimate skips in some sections of the codebooks. For example, if we ask someone whether or not they have ever used marijuana, and they say "no", it would not make sense to ask them more detailed questions about their marijuana use (e.g. quantity, frequency, onset, impairment, etc.). When analyzing more detailed questions regarding marijuana (e.g. have you ever smoked marijuana daily for a month or more?), those individuals that never used the substance may be coded as “legitimate skip” or show up as missing data. Since they have never used marijuana, we can assume that their answer is "no", they have never smoked marijuana daily. This would need to be explicitly recoded.

```
# in this example, 7 is given in the codebook as a legitimate skip and is recoded to 0 which is given in
# the codebook as Never
my_data$VAR1[my_data$VAR1 == 7] <- 0

# In the example below, NA’s are assumed to be legitimate skips, so are recoded to 0 which is Never.
my_data$VAR1[is.na(my_data$VAR1)] <- 0
```

3. Creating Secondary Variables

A. Collapse response categories

If a variable has many response categories, it can be difficult to interpret the statistical analyses in which it is used. Alternatively, there may be too few subjects or observations identified by one or more response categories to allow for a successful analysis. In these cases, it is necessary to collapse across categories. For example, if a religion variable has separate response categories for the different denominations of the major religions, then you may want to collapse all of the denominations into “No Religion”, “Christian”, “Islam”, “Traditional”, “Other”.

```
# Remove error codes before collapsing categories
# Next, name and initialize the new variable by filling it with NAs.
my_data$religion<-rep(NA,nrow(my_data))

my_r6$religion[my_r6$Q98A == 0] <- 0
my_r6$religion[my_r6$Q98A > 0 & my_r6$Q98A < 18] <- 1
my_r6$religion[my_r6$Q98A > 17 & my_r6$Q98A < 25] <- 2
my_r6$religion[my_r6$Q98A == 25] <- 3
my_r6$religion[my_r6$Q98A > 25] <- 4

freq(my_r6$religion)

# For the new religion variable, 0 = No religion, 1 = Christian, 2 = Muslim, 3 = Traditional religion,
# and 4 = Other. If the counts for “No Religion” are small, then it would
#  make sense just to include those respondents as “Other”
```

B. Create aggregate variables

In many cases, you will want to combine multiple variables into one. For example, while the codebook may assess several individual anxiety disorders, you may be interested in anxiety more generally. In this case you would create a general anxiety variable in which those individuals who received a diagnosis of social phobia, generalized anxiety disorder, specific phobia, panic disorder, agoraphobia, or obsessive-compulsive disorder would be coded "yes" and those who were free from all of these diagnoses would be coded "no".

```
# First, name and initialize the new variable by filling it with 0 (in this example 0 means “no
# anxiety disorder”)
my_data$anxiety <- rep(0, nrow(my_data))

# Then, if the person responded positively to any of the five anxiety conditions, assign 1 to the new
#anxiety variable.
my_data$anxiety[my_data$AnxVar1 == 1 | my_data$AnxVar2==1 | my_data$AnxVar3 == 1 |
                    my_data$AnxVar4==1 | my_data$AnxVar5 == 1] <- 1

# Finally, if there was missing data in the original anxiety variables, then recode the new variable with
# NA
my_data$anxiety[is.na(my_data$AnxVar1) & is.na(AnxVar2) &
                    is.na(my_data$AnxVar3) & is.na(my_data$AnxVar4) &
                    is.na(my_data$AnxVar5)] <- NA
```

C. Create composite variables, such as counters, composite scores, or averages

If you are working with a number of variables that are related to a single construct, it may be useful to create a composite variable or score. For example, if the codebook has a list of nicotine dependence symptoms meant to address the presence or absence of nicotine dependence (e.g. tolerance, withdrawal, craving, etc.), rather than using a dichotomous variable (i.e. nicotine dependence present/absent), you may want to examine the construct as a dimensional scale (i.e. the number of

nicotine dependence symptoms the respondent has). In this case, recode each symptom variable so that yes=1 and no=0 and then sum the items so that they represent one composite score.

An example of a continuous secondary composite variable is a variable that represents the average of a respondent's math, science, English and social studies grade.

```
# First, name and initialize the new variable by filling it with NA
my_data$nd_sum <- rep(NA, nrow(my_data))

# The example below assumes each symptom variable is coded 1 if the person has the nicotine
# dependent symptom and 0 if they do not.
my_data$nd_sum <- my_data$nd_symptom1 + my_data$nd_symptom2 +
                  my_data$nd_symptom3 + my_data$nd_symptom4

# When calculating an average grade, NA in any one of the subjects would make the average
# incomplete and un-comparable, so the record should be removed from the statistical analysis.
my_data$Avg_grade[is.na(my_data$H1ED11) | is.na(my_data$H1ED12) |
                  is.na(my_data$H1ED13) | is.na(my_data$H1ED14)] <- NA

# The example below calculates an average grade, where an A is coded 1, B is coded 2, C is coded 3,
# D and below is coded 4. All non-response categories have already been coded NA.
my_data$Avg_grade <- rep(NA, nrow(my_data))
my_data$Avg_grade <- (as.numeric(my_data$H1ED11) + as.numeric(my_data$H1ED12) +
                      as.numeric(my_data$H1ED13) + as.numeric(my_data$H1ED14)) / 4
```

4. Miscellaneous Data Management Tasks

A. Labeling variable response categories

When nominal and ordinal variables have numeric response values (i.e. dummy codes), it can be useful to label those values so that the labels are displayed in your output.

```
levels(my_data$VAR1)
levels(my_data$VAR1) <- c("value0label", "value1label", "value2label", "value3label")

# make sure you reassign the levels in the correct order!
```

B. Create groups that will be compared to one another

Often, you will need to create groups or sub-samples from the data set for the purpose of making comparisons (for example by gender or by the existence of a condition). It is important to be certain that the groups that you would like to compare are of adequate size and number. For example, if you were interested in comparing complications of depression in parents who had lost a child through miscarriage vs. parents who had lost a child in the first year of life, it would be important to have large enough groups of each. It would not be appropriate to attempt to compare 5000 observations in the miscarriage group to only 9 observations in the first-year group.

C. Specifying categorical data type for a numerically coded variable

Sometimes R confuses dummy codes for numerical data. If code is not working as expected try forcing R to recognize the dummy codes as factors (categories).

```
as.factor(Var_name)
```