USING R FOR EPIDEMIOLOGICAL RESEARCH

Kiffer G. Card, PhD
Kirk J. Hepburn, MPP

Using R for Epidemiological Research:
An Introduction

By: Kiffer G. Card, PhD & Kirk J. Hepburn, MPP

Table of Contents

Before you Begin

This manual will provide a rapid overview of the content you will need to conduct basic epidemiologic analyses in R. I strongly encourage you to also explore a general introduction to R, such as the one found at Lynda.com titled “R Statistics Essential Training” by Barton Poulson. He provides 6 hours of very useful training material which will bring you up to speed and provide much more than this introduction can. It will take you an afternoon or two to complete, but it will be well worth the time spent. You should be able to access the Lynda learning database through your university library.

1. Getting Started with R

In this section, we will introduce R and explore how it can be used to summarize and analyze data with a focus on the NHANES dataset, which you will use for completing each skill check. In this guide, we often provide the simplest methods for meeting each objective, however, in R there are numerous ways to complete given tasks and you are welcome to explore and identify strategies that work better for you.

1.1 R Basics

1.1.1 About R

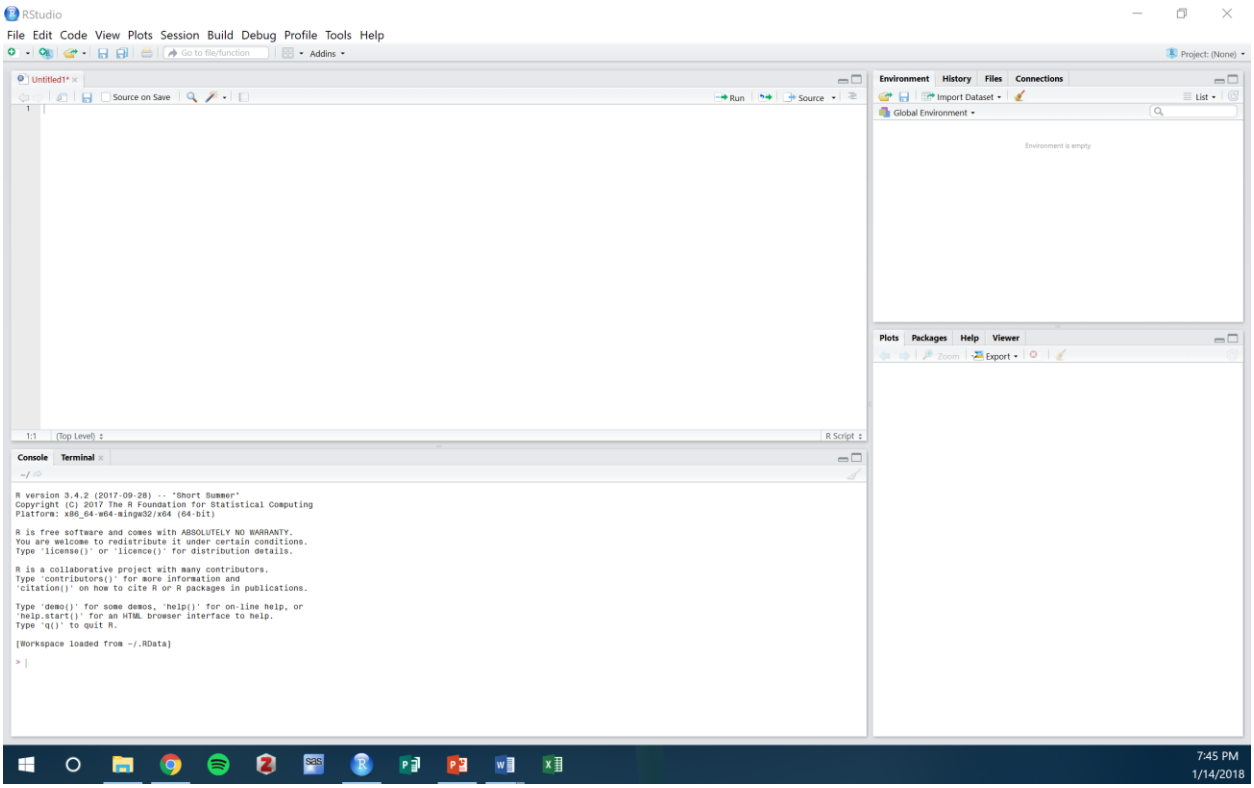
R is a statistical language commonly accessed through the integrated development environment called RStudio. Base R performs a vast number of useful statistical operations, but it can be enhanced with packages. These packages are open source and created by the community of R users, making these resources open for public use free of charge.

1.1.2 Install R and RStudio

Follow the instructions at this link to install both R and RStudio on your computer. Install R first.

1.1.3 A quick tour of RStudio

There are several important windows to become familiar with when working in RStudio:



• The top left window is a script editor. Here you can edit and execute R code (by highlighting and hitting ctrl+Enter).

• The bottom left is the R console, where the R engine does its work. You're your code is executed, with results and error notifications. You can also enter ad hoc commands in this window, but these commands will not be saved.

• In the top right, R objects are displayed and summarized. Objects are named sets of data, whether a set of data from a survey, a list of names, a set of randomly-generated numbers, or even functions you have defined (created) yourself.

• The bottom right contains several useful tools: a file explorer, a graphical device for displaying any plots you create (histograms, bar charts, and the like), a list of the packages you have installed, a viewer for function help documentation, and a viewer for web content.

1.1.4 Getting help

Specific functions are described fully in R. Getting help on the function `mean`, for instance, is as simple as entering `?mean` into the console. The documentation for the `mean` function will appear in the"Help" window in Rstudio.

Furthermore, if you have questions about a specific function or error, the R community is extremely active. Your question has likely already been asked on StackExchange or similar websites (e.g., https://www.r-bloggers.com/, https://www.statmethods.net/).

1.1.5 R Packages

As mentioned, R comes ready with a large set of useful functions, but the R community has created many new functions which perform more complex or specialized tasks or which simplify the language around basic R tasks. For the latter purpose, I highly recommend installing and familiarizing yourself with the Tidyverse family of packages, particularly `plyr`, `dplyr`, `ggplot2`, and `readr`. These simplify and extend data reading, cleaning, and plotting.

You will need these packages to perform the tasks I demonstrate below. Install them from RStudio under Tools > Install Packages… Input the list of desired packages: plyr tidyverse. Hit `OK` and RStudio will download and install the packages from CRAN. You can also download the packages from the R console:

```
install.packages(pkgs = c("plyr", "tidyverse"))
```

Finally, to activate all of the functions in each package, enter the following in the script editor or R console:

```
library(plyr) library(tidyverse)
```

This will load all of the functions contained in these packages into your R session; without doing this, R will not have access to many of the functions used below.

1.1.6 R Syntax

An R function almost universally follows this form: `function(arguments)`. For instance, the `mean` function can take the following arguments, each of which is described in the functions help file:

x	An R object. Currently there are methods for numeric/logical vectors and date, date-time and time interval objects. Complex vectors are allowed for trim = 0, only.
trim	the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
na.rm	a logical value indicating whether NAvalues should be stripped before the computation proceeds.
...	further arguments passed to or from other methods.

R takes the arguments in order, meaning that you can provide it with the arguments without naming them and R will simply assume what each argument represents. I find it less confusing if I go ahead and name each argument. For instance:

```
mean(x = women$height, trim = 0, na.rm = TRUE)

## [1] 65
```

Here I have found the average height of a women from a dataset included in R. I have not trimmed any observations, nor have I removed missing values. Because the trim and na.rm arguments have defaults, I do not need to provide those arguments. I can achieve the same result by inputting

```
mean(x = women$height)

## [1] 65
```

1.1.7 Annotating Your R Code

R has a comment flag, the hashtag (#). Any text on a line of code beginning with a # will not be read by the R engine. This allows you to explain what your code is trying to accomplish for your future self or for other readers of your code. It is vital that you comment your code, and it is best that you comment as you go. You will thank yourself time and again for developing this habit.

```
#Calculate Mean for Height of Women
mean(x = women$height, trim = 0, na.rm = TRUE)

## [1] 65
```

1.1.8 Entering Data into R

Before we begin using R to explore data, we must read that data into the R environment. Anything stored in the R environment's memory is called an object. To assign data to an object, we use the <- syntax.

The data we will be using is from the National Health and Nutrition Examination Survey (NHANES) from the Centers for Disease Control (CDC). We will begin by looking at the demographic data, which is located here.

Much of the data that you encounter in your careers will be stored in comma-separated values or other text-based formats (.tab, for instance), and R has commands specifically for these formats: read.csv, read.delim, etc. NHANES distributes their data in SAS transport files, which can be viewed in a SAS universal reader but which require SAS to manipulate. Fortunately, Hadley Wickham has created – among the myriad packages he has otherwise created – a package for easily accessing a variety of formats: `readr`. If you have already installed the `tidyverse` package as instructed in 2.1.3, you have already installed `readr`. A simple call to `library(tidyverse)` will load `readr` along with the other useful packages in the Tidyverse. `readr` also has a handier version of `read.csv`, `read_csv`, which will read in csv files as follows:

```
read.csv("https://github.com/tidyverse/readr/raw/master/inst/extdata/mtcars.csv")
```

```
##     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## 1  21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## 2  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## 3  22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## 4  21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## 5  18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## 6  18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## 7  14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## 8  24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## 9  22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## 10 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## 11 17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## 12 16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## 13 17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## 14 15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## 15 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## 16 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## 17 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## 18 32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## 19 30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## 20 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## 21 21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## 22 15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## 23 15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## 24 13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
```

```
## 25 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## 26 27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## 27 26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## 28 30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## 29 15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## 30 19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## 31 15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## 32 21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

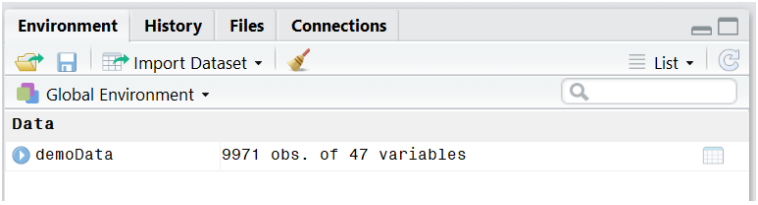
If your data is available as a SAS transport file (as the NHANES data is) the `haven` package can be loaded via `library` to access to the `read_xpt` function, which can read SAS transport files into an R object. `read_xpt` takes only one argument:"file", which the help documentation tells us should be"Either a path to a file, a connection, or literal data (either a single string or a raw vector)." The data we are interested in is stored on the Internet at the URL below:

"https://wwwn.cdc.gov/Nchs/Nhanes/2015-2016/DEMO_I.XPT"

You could download the data and then direct `read_xpt` to it, but since the help documentation says that"file" can be a connection, let's just use the URL and assign the data to an object called `demoData` as follows:

```
#Load needed package
library(haven)

#Download the SAS transport file (.xpt) from NHANES.
demoData <- read_xpt(file = "https://wwwn.cdc.gov/Nchs/Nhanes/2015-2016/DEMO_
I.XPT")
```

"demoData" is the name we have given to our new object; the"<-" symbol is like an arrow which tells R"Assign the results of this function to this object". Therefore, whatever the `read_xpt` function manages to spit out (but only if it manages to spit something out) will be stored in an object named"demoData". After you run this line of code, the following should appear in your object explorer.



You now have the demographic data for the 2015-2016 administration of NHANES!

1.2 Exploring and Summarizing Data

1.2.1 Understanding Your Data

Before exploring the data for our own purposes, it will help to understand what R thinks of the data. Run this line of code:

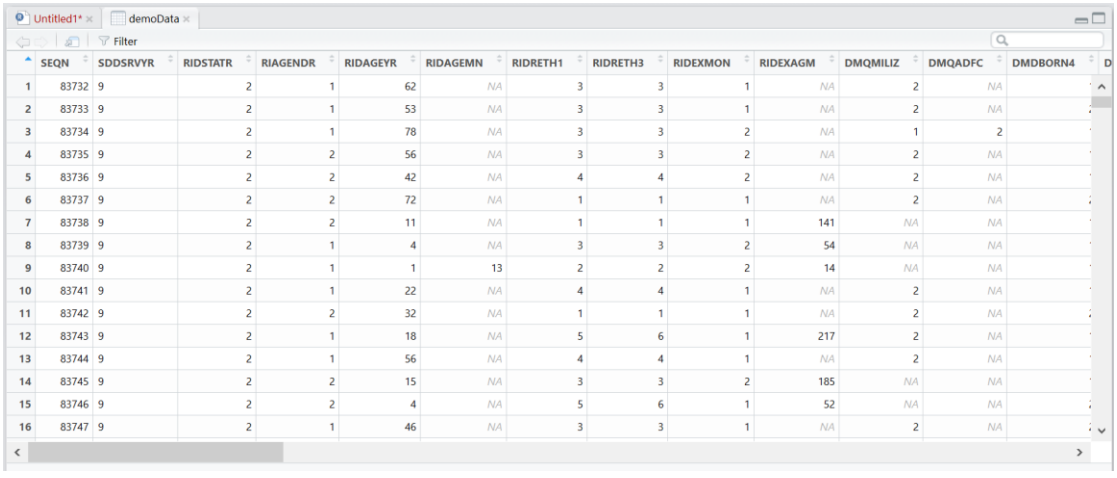
```
class(demoData)

## [1] "tbl_df"     "tbl"        "data.frame"
```

The code will give you a result with a length of 3. The first two,“tbl_df” and“tbl” are not important at this point, but the third is vital. It has told us that the object `demoData` is stored as a data frame.

In R, there are some basic object classes: logical, numeric, factor, character, list, and data frame. Logical data can take either TRUE, FALSE, or NA (the latter being the value for missing data in R). Numeric data can take any real number. Character data can take characters or strings of characters or even multiple strings of characters. Factor data behave best for categorical variables, as each factor datum fits into a category to which a number is also assigned. (Much more on this later.)

Lists are objects which can hold other objects, i.e.“lists” of objects. These objects can be of any class (including other lists) and be any length. I like to think of these as“folders” similar to your computer’s folder system (though this is not entirely accurate).

Data frames are similar to lists in that they can hold objects of different classes, but all of the objects in a data frame must be the same length. If this sounds at all familiar, it is because data frames a way to look at data like a spreadsheet. Click on the“demoData” name in your object explorer. This calls RStudio’s `View` function on your data, which opens a viewer like this:


Untitled1* demoData

Filter

	SEQN	SDDSRVYR	RIDSTATR	RIAGENDR	RIDAGEYR	RIDAGEMN	RIDRETH1	RIDRETH3	RIDEXMON	RIDEXAGM	DMQMILIZ	DMQADFC	DMDBORN4
1	83732	9	2	1	62	NA	3	3	1	NA	2	NA	
2	83733	9	2	1	53	NA	3	3	1	NA	2	NA	
3	83734	9	2	1	78	NA	3	3	2	NA	1	2	
4	83735	9	2	2	56	NA	3	3	2	NA	2	NA	
5	83736	9	2	2	42	NA	4	4	2	NA	2	NA	
6	83737	9	2	2	72	NA	1	1	1	NA	2	NA	
7	83738	9	2	2	11	NA	1	1	1	141	NA	NA	
8	83739	9	2	1	4	NA	3	3	2	54	NA	NA	
9	83740	9	2	1	1	13	2	2	2	14	NA	NA	
10	83741	9	2	1	22	NA	4	4	1	NA	2	NA	
11	83742	9	2	2	32	NA	1	1	1	NA	2	NA	
12	83743	9	2	1	18	NA	5	6	1	217	2	NA	
13	83744	9	2	1	56	NA	4	4	1	NA	2	NA	
14	83745	9	2	2	15	NA	3	3	2	185	NA	NA	
15	83746	9	2	2	4	NA	5	6	1	52	NA	NA	
16	83747	9	2	1	46	NA	3	3	1	NA	2	NA	

As in a spreadsheet, each row represents a case or observation, while each column represents a variable (since this is from a survey). Always remember, though, that to R this

data frame is basically a list of objects, with each column being an object. They each have their own class. Try the following code:

```
class(demoData$RIAGENDR)

## [1] "numeric"
```

You should get“numeric” as the response. While demoData has the“data.frame” class, demoData$RIAGENDR has the“numeric” class. Because SAS works primarily with numeric data, all of the objects in demoData have the“numeric” class. This is important as not all of these variables should be treated as numeric. For instance, SEQN is an ID number for which an average or standard deviation has no meaning whatever.

1.2.2 Data Dictionaries & Skip Patterns

In order to gain insight on just what these numbers mean, we must consult the data dictionary or codebook, found here. It provides a summary of the dataset, its purpose, and a variable-by-variable description.

Compare the variables `DMQMILIZ` and `DMQADFC`. The first asks whether a person has served in the US Armed Forces, and the second asks whether a person served in a foreign country. Pretend that you want to know what proportion of Americans served in a foreign country. You might divide the number of“Yes” responses to `DMQADFC` (n = 258) by the number of respondents to the question, 527 (the number of non-missing responses). This gives us an estimate of 49.0%.

This would be incorrect for several reasons, but the most important is that DMQADFC is the target of a skip pattern in the survey Looking at `DMQMILIZ` you can see that any response other than“Yes” causes the surveyor to skip DMQADFC. We should have been suspicious that only 527 responses were recorded, but looking at the items before it confirms that most respondents were not asked this question! More accurately, we can divide 258“Yes” responses to `DMQADFC` by the total number of responses to the previous question, 6,149. This yields a more accurate estimate of 4.2%. Be on the lookout for skip patterns in your data explorations, as they may not be as obvious as this. Data analysis requires this sort of careful critical thinking.

1.2.3 Using R to Look at Your Data

While the data dictionary gives us some information on each variable, we can use R to explore it more deeply. The `summary` function is useful here. Let’s take a look at the age variable, `RIDAGEYR`.

```
summary(demoData$RIDAGEYR)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     9.0    27.0    31.9    53.0    80.0
```

This will give you a few summary statistics about your data: quartiles, the median, mean, minimum, and maximum. With that, we have a sense of our age distribution!

1.3 Combining Data Sets

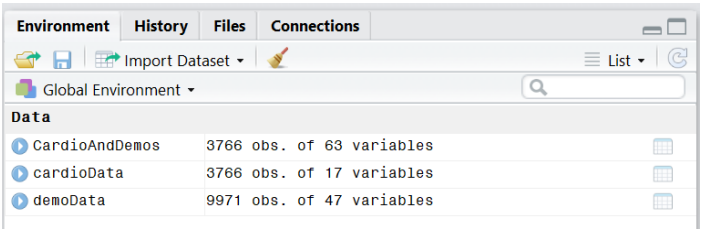
Manipulating datasets is another essential skill needed for data analysis.

1.3.1 Adding Additional Variables

For instance, sometimes datasets are merged together when variable sets are stored in separate datasets for various reasons. The NHANES data we have been using for this module is a good example of this. To combine two sets of variables for the same individuals you can use the 'merge' function after reading in your second data set. To merge these datasets a common identifier must be specified, in this case SEQN:

```
#Import New Data Set
cardioData <- read_xpt(file = "https://wwwn.cdc.gov/Nchs/Nhanes/2015-2016/CDQ
_I.XPT")

# Merge Dataset based on the respondent identifier SEQN
CardioAndDemos <- merge(x = cardioData, y = demoData, by="SEQN")
```

As you can see from the figure below, the merge function matches across the SEQN variable and adds your new variables to each person who had a record in both of the specified data sets:



1.3.2 Adding Additional Observations

For a variety of reasons, you might want to instead add additional observations to your data set, without necessarily changing the variables you are interested in. In this case, you can use the `rbind` function (short for "row bind") to add your new observations:

```
CompleteSample <- rbind(<dataframe1>, <datframe2>)
```

1.3.3 Dropping Variables

There are also several ways to reduce your data frame by dropping variables that you are not interested in using. The first is to create a list of the variables you want to keep and assign it to an object. (Note: objects with multiple elements are created using the "concatenate" or `c()` command).

```
keepvars <- c("RIDAGEYR", "DMQMILIZ", "DMQADFC")
```

```
Values
  keepvars          chr [1:3] "RIDAGEYR" "DMQMILIZ" "DMQADFC"
```

Now you can create a new data set which keeps only variables in this list:

```
newdata <- CardioAndDemos[keepvars]
ncol(CardioAndDemos)

## [1] 63
```

```
Data
CardioAndDemos    3766 obs. of 63 variables
cardioData        3766 obs. of 17 variables
demoData          9971 obs. of 47 variables
newdata           3766 obs. of 3 variables
```

1.3.4 Dropping Observations

Other times you might want to only select a subset of observations, dropping those which you are not interested in. Often this is done based on participant responses to a specific question. Perhaps the easiest way to do this is to use the 'which' function. For example, in the code below we create a subset dataset which contains records for males (i.e., RIAGENDR = 1) under the age of 18 (i.e., RIDAGEYR>18):

```
# Create a subset of the data containing only males under the age of 18
YoungMales <- CardioAndDemos[which(CardioAndDemos$RIAGENDR==1 & CardioAndDemo
s$RIDAGEYR>18), ]
```

```
Data
CardioAndDemos    3766 obs. of 63 variables
cardioData        3766 obs. of 17 variables
demoData          9971 obs. of 47 variables
newdata           3766 obs. of 3 variables
YoungMales        1811 obs. of 63 variables
```

1.4 Recoding Variables

1.4.1 Recoding Categorical Variables

In addition to merging data sets, you will also find yourself wanting to edit variables. When doing this I suggest always creating a new variable, rather than writing over your existing data. In the example below, we collapse the race/ethnicity variable into a white vs. non-white variable. This is often done when sub-groups are not large enough to perform a specific analysis or when your research question involves a specific dichotomy.

The following code does several things. It creates a new object in the data frame called `WHITE`. To this variable, it assigns the results of a logical test, "Does `RIDRETH1` equal 3 for

this respondent?" If so, it assigns the value `TRUE`, if not then `FALSE`. Then it summarizes the new variable.

```
# Create a new variable based on whether respondent is white or not
CardioAndDemos$WHITE <- CardioAndDemos$RIDRETH1 == 3
summary(CardioAndDemos$WHITE)

##    Mode   FALSE    TRUE
## logical    2475    1291
```

In R, the logical class is the same as a numeric variable with only 0s and 1s.

1.4.2 Categorizing Continuous Data

A similar strategy can be used to collapse a continuous variable into a categorical variable. Take for instance the example below where we create three age groups for the age of individuals when they first started smoking regularly (SMD030). Here we can take advantage of R's `factor` class. Factors are especially useful for categorical data. While they behave a bit like string objects, they are, in fact, objects which can take any of a set of "levels", or categories. Each category also has a number behind it. Here we create three categories by first assigning characters based on a person's smoking initiation age, then coercing the object into a factor object.

```
# Read in Smoking Data Set
smoke <- read_xpt(file = "https://wwwn.cdc.gov/Nchs/Nhanes/2015-2016/SMQ_I.XP
T")

# Create an empty column
smoke$SMOKINGAGE <- NA

# Collapse Ages 7 to 18 -- 1
smoke$SMOKINGAGE[smoke$SMD030 >= 7 & smoke$SMD030 <= 18] <- "07-18"
# Collapse Ages 19 to 30 -- 2
smoke$SMOKINGAGE[smoke$SMD030 >= 19 & smoke$SMD030 <= 30] <- "19-30"
# Collapse Ages 31 to 58 -- 3
smoke$SMOKINGAGE[smoke$SMD030 >= 31 & smoke$SMD030 <= 58] <- "31-58"
smoke$SMOKINGAGE <- as.factor(smoke$SMOKINGAGE)
summary(smoke$SMOKINGAGE)

## 07-18 19-30 31-58  NA's
##  1495   756    77  4673
```

The data dictionary tells us that 0, 777, and 999 are not ages at which the respondent started smoking. Since these were not included in the conditions we wrote, they have been left as NA.

Factors can be tricky. It is vital to remember that 1) they don't like having levels added once created - for instance we couldn't float in and add a person with a smoking age of "58+", since this isn't one of the levels that were there when we calls `as.factor` on the object. 2) Each variable is really a number hiding behind the category: if you call

`as.numeric` on the object, "07-18" would be turned into a "1", not "17" or an error. 3) The lowest category is the reference level; in certain types of analysis, all other categories may be compared to that one.

1.4.3 Combining Variables

Other times you will need to combine multiple variables into a single variable. This is often the case when you have a follow-up question that is asked only for a subset of your sample. Take for instance MCQ010 (i.e.,"Has a doctor ever told yoo that you have asthmas." and MCQ025 (i.e.,"How old were you when you were first told you had asthma?") from the NHANES"Medical Conditions" dataset. If we wanted to create a three level variable distinguishing between (1) adult diagnosis, (2) childhood diagnosis, and (3) no diagnosis, we would need to collapse MCQ010 and MCQ025 as follows:

```
# Read in Medical Conditions Data Set
medcond <- read_xpt(file = "https://wwwn.cdc.gov/Nchs/Nhanes/2015-2016/MCQ_I.
XPT")

# Create new empty variable
medcond$ASHTMAAGE <- NA
# Adult diagnosis (18 or older)-- 1
medcond$ASHTMAAGE[medcond$MCQ010 ==1 & medcond$MCQ025 >=18] <- 1
# Childhood diagnosis (Younger than 18) -- 2
medcond$ASHTMAAGE[medcond$MCQ010 ==1 & medcond$MCQ025 <=17] <- 2
# No diagnosis -- 3
medcond$ASHTMAAGE[medcond$MCQ010 ==2] <-3
```

1.5 Descriptive Statistics

1.5.1 Frequencies

Now that you've had some exposure to creating and working with variables, lets examine how we can explore and summarize our data. To begin, let's look at the help file for the `table` function. Its first argument says,"… one or more objects which can be interpreted as factors (including character strings), or a list (or data frame) whose components can be so interpreted.""…" means"any number of items".

Our gender variable is an object which can be interpreted as a factor, i.e. a categorical variable. It's a numeric object, yes, but `table` is going to convert it to a factor before calculating. Call `table` as follows:

```
table(demoData$RIAGENDR)

##
##    1    2
## 4892 5079
```

This will give you a quick frequency table, with the same values as the data dictionary. Note that the $ symbol is used in R to subset the data, or to specify an object within an object.

The $ symbol usually only works on data frames and lists. The following is a more common way to accomplish the same task, using square brackets.

```
table(demoData[ , "RIAGENDR"])

##
##    1    2
## 4892 5079
```

Note that this method requires you to name the object using quotation marks. Also, and crucially, you have to specify all of the dimensions of the object when using this method, separated by commas. Our data frame has two dimensions,“rows” and“columns”; I have simply left the row space blank to mean“all rows”.

1.5.2 Cross Tabulations

`table` also allows us to perform cross tabulations. All we need to do is provide another“object which can be interpreted as a factor”. Let’s try military service.

```
table(demoData$RIAGENDR, demoData$DMQMILIZ)

##
##        1    2
##   1  486 2481
##   2   41 3141
```

This produces a frequency table where gender is cross tabulated by whether the respondent served in the military. This is good, but it’s difficult to know which 1 means“Male” and which means“Yes”. First, in R rows are always the first dimension in a specification, just as in the subsetting example. Second, it will be easier if we give names to our dimensions. Look again at your help documentation for `table`. There is an argument called“dnn”.“the names to be given to the dimensions in the result (the dimnames names).” Let’s provide some names, in the same order as the variables were provided:

```
table(demoData$RIAGENDR, demoData$DMQMILIZ, dnn = c("Gender","Military servic
e"))

##       Military service
## Gender    1    2
##      1  486 2481
##      2   41 3141
```

Note that the names have to be“concatenated” into a single object, which we do using the `c` function. This should give us a table where the margins are named.

Finally, suppose that we are interested in the proportions, rather than frequencies, of cell. For that, we can use the `prop.table` function. The help documentation says that `prop.table` takes a table as its input. It just so happens that the `table` function produces a table object! All that remains is to pass the table we already made to `prop.table`.

```
prop.table(x = table(demoData$RIAGENDR, demoData$DMQMILIZ, dnn = c("Gender",
"Military")))
```

```
##       Military
## Gender           1           2
##      1 0.079037242 0.403480241
##      2 0.006667751 0.510814767
```

Voila! This produces a proportion table out of our frequency table. But the proportions use the entire table as the denominator, meaning that 7.9% of people are men who served in the military, 51.1% are women who did not, etc. This is not always useful, so the `prop.table` function take an argument called"margin". Margin should be a number indicating the dimension you want used as the denominator for your proportions. Remember that R starts with rows, so use"1" if you want the rows to act as the denominator.

```
prop.table(table(demoData$RIAGENDR, demoData$DMQMILIZ, dnn = c("Gender", "Mil
itary")), margin = 1)
```

```
##       Military
## Gender          1          2
##      1 0.16380182 0.83619818
##      2 0.01288498 0.98711502
```

There: 16.4% of men served in the military while 1.3% of women did so.

1.5.3 Means & Medians

The `summary` function already provides us with means and medians. We can find the mean or median by using the functions `mean` and `median`. This is simple enough, though note that unless you specify"na.rm = TRUE" (short for"remove missing"), an object with any missing elements will return NA. This is true of many of R's calculation functions; they take missingness quite seriously. Suppose, though, that we want to know the mean of something in a cross tabulation? For this, we can use the `by` function. Let's find the mean age for men and women.

```
by(data = demoData$RIDAGEYR, INDICES = demoData$RIAGENDR, FUN = mean)
```

```
## demoData$RIAGENDR: 1
## [1] 31.62367
## ------------------------------------------------------------
## demoData$RIAGENDR: 2
## [1] 32.16499
```

Look at the help documentation."data" can be any object for which our desired function makes sense. It can even be an entire data frame. We used the age variables."INDICES" must be a factor (or interpretable as one)."FUN" is the function. In this case, we don't use parentheses, just the name of the function. The `by` function splits the age object and applies the `mean` function to it.

1.5 Skills Check #1

Objectives. This skill check will help students to:

- Input datasets into R
- Merge two datasets.
- Recode continuous variables as categorical variables.
- Recategorize categorical variables into new categories.
- Summarize data by calculating basic descriptive statistics.
- Generate stratified descriptive statistics.

Tasks. Complete the following tables using data from the NHANES demographic, questionnaire, and examination data. Please note that the NHANES data require sample weights, which are not covered in this guide. You do not need to use these here, but you should be aware that the results obtained in your analyses will not be representative or generalizable.

Table 1. Unweighted Descriptive Statistics of the National Health and Nutrition Examination Survey (NHANES), 2015-2016

	Overall (N = _____)	
	n	%
Age (in Years)		
<10		
10-30		
30-59		
≥60		
Sex		
Male		
Female		
Race/Ethnicity		
White		
Black/Hispanic		
Other		
Education Level		
Less than High School Diploma		
Greater than High School Diploma		
Marital Status		
Married/Living with Partner		
Other		
Annual Income		
<$15,000		
$15,000-$34,999		
$35,000-$64,999		
≥$65,000		

Table 2. Unweighted Descriptive Statistics for Participants with and without Hypertension (Source: NHANES, 2015-2016)

	Hypertension[1]			
	Yes (n = _____)		**No (n = _____)**	
Continuous Variables	Median	Q1, Q3	Median	Q1, Q3
Age (in Years)				
Minutes spent doing…				
vigorous activities, per day				
moderate activities, per day				
sedentary activity, per day				
Playing active video games.				
BMI				
Categorical Variables	n	%	n	%
Sex				
Male				
Female				
Race/Ethnicity				
White				
Black/Hispanic				
Other				
Education Level				
Less than High School Diploma				
Greater than High School Diploma				
Marital Status				
Married/Living with Partner				
Other				
Annual Income				
<35,000				
$35,000-$64,999				
≥$65,000				
Smoking - Current				
No, I do not smoke				
Yes, I smoke every day or some days				

[1] Hypertension, or high blood pressure, is defined as being told by a doctor or other healthcare professional that they had hypertension on two or more visits.

2 Basic Biostatistical Analyses

In this section we will show how to perform basic biostatistical analyses.

2.1 Epidemiological Methods for Cross-Sectional Data

2.1.1 Chi-Squared

Are men and women different in terms of whether they have ever experienced chest pain? Since sex and gender have been shown to have some impact on heart health, my guess is that yes, they are different.

In our survey, experiencing chest pain is a categorical variable, `CDQ001`. Let's make a frequency table.

```
table(CardioAndDemos$CDQ001)
```

```
##
##    1    2    7    9
## 1042 2716    1    5
```

According to our data dictionary, 1,042 people said they had experienced chest pain, 2,716 said they had not, 1 refused to answer, and 5 said they did not know.

Now let's take a look by sex.

```
table(CardioAndDemos$CDQ001, CardioAndDemos$RIAGENDR, dnn = c("Chest Pain", "
Sex"))
```

```
##           Sex
## Chest Pain    1    2
##          1  502  540
##          2 1305 1411
##          7    1    0
##          9    3    2
```

```
prop.table(x = table(CardioAndDemos$CDQ001, CardioAndDemos$RIAGENDR, dnn = c(
"Chest Pain", "Sex")))
```

```
##           Sex
## Chest Pain            1            2
##          1 0.1333687566 0.1434643996
##          2 0.3467056323 0.3748671626
##          7 0.0002656748 0.0000000000
##          9 0.0007970244 0.0005313496
```

The numbers and the proportions are different, but not dramatically so. Still, these differences might be present in the population, but only if they are statistically significantly different. For this we will need a χ^2, or "chi-squared", test. First, though, we should work with clean data without the missing responses cluttering up the test.

```
CardioAndDemos$ChestPain <- CardioAndDemos$CDQ001
CardioAndDemos$ChestPain[CardioAndDemos$ChestPain == 7 | CardioAndDemos$Chest
Pain == 9] <- NA

table(CardioAndDemos$ChestPain, CardioAndDemos$RIAGENDR, dnn = c("Chest Pain"
, "Sex"))

##           Sex
## Chest Pain    1    2
##          1  502  540
##          2 1305 1411

prop.table(x = table(CardioAndDemos$ChestPain, CardioAndDemos$RIAGENDR, dnn =
c("Chest Pain", "Sex")), margin = 2)

##           Sex
## Chest Pain         1         2
##          1 0.2778085 0.2767811
##          2 0.7221915 0.7232189
```

There! Without the missing responses, we have some proportion estimates. Of people over 40, 13.4% of males have experienced chest pain while 14.4% of females have. But can we say with any confidence that men and women in the population have this difference?

```
chisq.test(x = CardioAndDemos$RIAGENDR, y = CardioAndDemos$ChestPain)

##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  CardioAndDemos$RIAGENDR and CardioAndDemos$ChestPain
## X-squared = 0.0011443, df = 1, p-value = 0.973
```

With a p-value of 0.973, it seems that we cannot say the observed interaction of gender and chest pain incidence is not due to random chance. Therefore, we do not reject the null hypothesis. As far as we know, these two variables are independent of one another.

Let's try another variable. Let's see if the age of smoking initiation is related to sex.

```
# Merge the smoking age dataset with the demographic set
SmokeAndDemo <- merge(x = demoData, y = smoke, by = "SEQN")

# Our smoking age columsn is already there. Make frequency tables
table(SmokeAndDemo$SMOKINGAGE, SmokeAndDemo$RIAGENDR, dnn = c("Age of smoking
initiation", "Sex"))

##                           Sex
## Age of smoking initiation   1   2
##                     07-18 959 536
##                     19-30 438 318
##                     31-58  31  46
```

```
prop.table(x = table(SmokeAndDemo$SMOKINGAGE, SmokeAndDemo$RIAGENDR, dnn = c(
"Age of smoking initiation", "Sex")), margin = 2)
```

```
##                            Sex
## Age of smoking initiation          1          2
##                       07-18 0.67156863 0.59555556
##                       19-30 0.30672269 0.35333333
##                       31-58 0.02170868 0.05111111
```

It appears that women over 40 who smoke are somewhat more likely to have started later in life. At the very least, it appears that the two categories are *related*, and a chi-squared test will help us know how likely this is to be the case.

```
chisq.test(x = SmokeAndDemo$SMOKINGAGE, y = SmokeAndDemo$RIAGENDR)
```

```
##
##  Pearson's Chi-squared test
##
## data:  SmokeAndDemo$SMOKINGAGE and SmokeAndDemo$RIAGENDR
## X-squared = 23.09, df = 2, p-value = 9.685e-06
```

The test gives us a chi-squared statistic of 23.09 which, at 2 degrees of freedom, has an associated p-value of .000009685. It seems very unlikely that the two categories are unrelated to each other. But this test doesn't tell us much beyond that.

2.1.2 T-tests

While chi-squared tests test the difference between categorical distributions, Student's t tests allow us to test that two means are different, or that a mean is significantly different from a specific value (usually 0). Let us consider age at smoking initiation again. We know it is related to gender (at least when we create the categories like we did). Why not see if the mean age of smoking initiation is different in the two groups?

```
# Copy the numeric smoking ages into a new column for cleaning
SmokeAndDemo$NUMERICSMOKINGAGE <- SmokeAndDemo$SMD030

# Clean up non-responses, as indicated in the data dictionary
SmokeAndDemo$NUMERICSMOKINGAGE[SmokeAndDemo$NUMERICSMOKINGAGE == 0] <- NA
SmokeAndDemo$NUMERICSMOKINGAGE[SmokeAndDemo$NUMERICSMOKINGAGE == 777] <- NA
SmokeAndDemo$NUMERICSMOKINGAGE[SmokeAndDemo$NUMERICSMOKINGAGE == 999] <- NA

summary(SmokeAndDemo$NUMERICSMOKINGAGE)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    7.00   15.00   18.00   18.41   20.00   58.00    4673
```

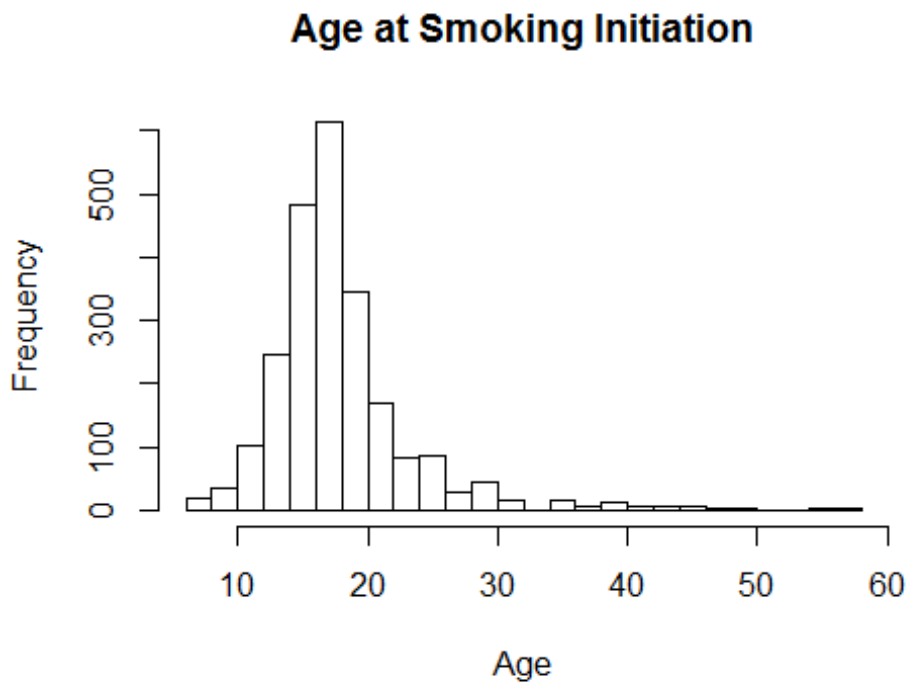
```
# Create a summary of age of smoking initiation by gender.
by(data = SmokeAndDemo$NUMERICSMOKINGAGE, INDICES = SmokeAndDemo$RIAGENDR, FU
N = summary)
```

```
## SmokeAndDemo$RIAGENDR: 1
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     7.0    15.0    17.0    17.9    20.0    58.0    1973
## ------------------------------------------------------------
## SmokeAndDemo$RIAGENDR: 2
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    7.00   16.00   18.00   19.22   21.00   58.00    2700
```

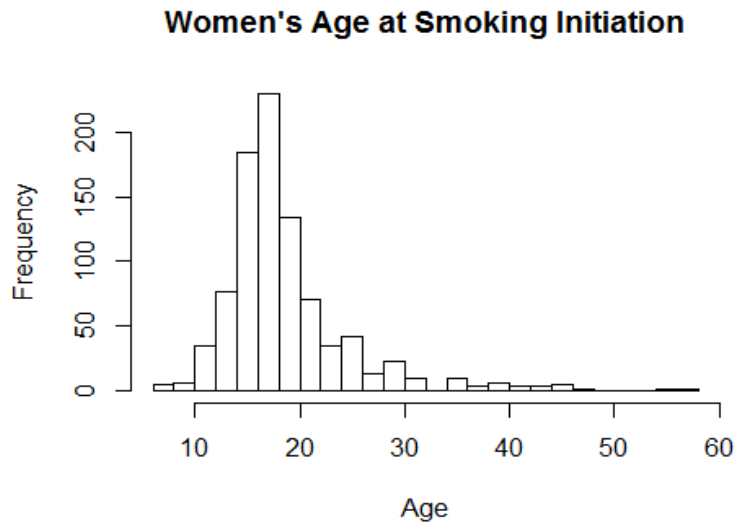
The means are different, but are they significanly so? To perform a t test, we make certain assumptions. First, we assume that the two groups' measurements are continuous or ordinal. Second, we assume that the sample is representative, preferably from a random sample. Third, the means we compare should be from normally distributed data. Fourth, the sample should be more than just a few cases. Finally, the variance of the two groups should be similar.

For the third and fifth assumptions, we should eyeball our data to see if it is normal. This is best done with a histogram of ages of smoking initiation using the `hist` function.

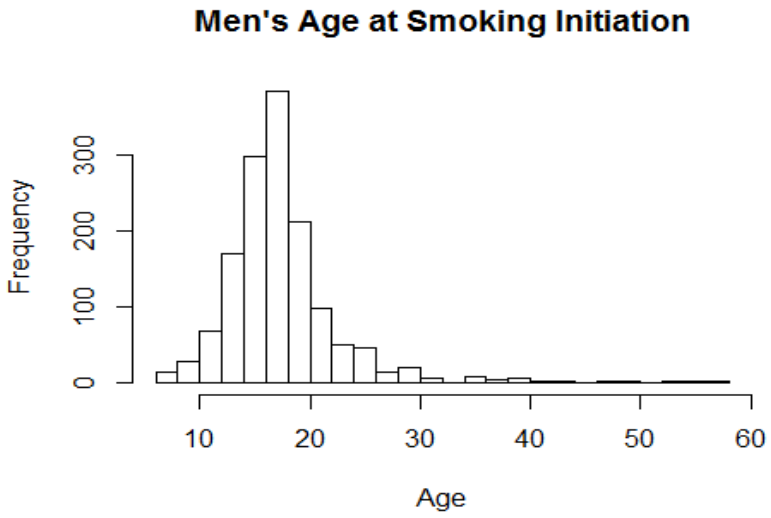
```
# Create a histogram of the full sample of ages. Give it a main title and a l
abel for the x axis.
hist(x = SmokeAndDemo$NUMERICSMOKINGAGE, breaks = 25, main = "Age at Smoking
Initiation", xlab = "Age")
```




```
# Look at the distribution of women's ages.
hist(x = SmokeAndDemo$NUMERICSMOKINGAGE[SmokeAndDemo$RIAGENDR == 2], breaks =
25, main = "Women's Age at Smoking Initiation", xlab = "Age")
```




```
# Look at the distribution of men's ages.
hist(x = SmokeAndDemo$NUMERICSMOKINGAGE[SmokeAndDemo$RIAGENDR == 1], breaks =
25, main = "Men's Age at Smoking Initiation", xlab = "Age")
```



It appears that our distributions are roughly normal, albeit a bit right-skewed. To perform Student's t test, we use the `t.test` function.

```
# Perform Student's t test. This test assumes equal variances, so set var.equ
al to `TRUE`
t.test(x = SmokeAndDemo$NUMERICSMOKINGAGE[SmokeAndDemo$RIAGENDR == 2], y = Sm
okeAndDemo$NUMERICSMOKINGAGE[SmokeAndDemo$RIAGENDR == 1], var.equal = TRUE)

##
##  Two Sample t-test
##
```

```
## data:  SmokeAndDemo$NUMERICSMOKINGAGE[SmokeAndDemo$RIAGENDR == 2] and Smok
eAndDemo$NUMERICSMOKINGAGE[SmokeAndDemo$RIAGENDR == 1]
## t = 5.5712, df = 2326, p-value = 2.821e-08
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.8542193 1.7821952
## sample estimates:
## mean of x mean of y
##  19.21667  17.89846
```

The results handily give us the means of the two groups. As in our categorical exploration, women are adopting smoking later in life by an average of 1.32 years. The 95% confidence interval means that the "true" population difference between 0.85 and 1.78 years, 19 times out of 20. The p-value is .00000002821, quite low. It is very unlikely that this difference is due to chance, so we reject the null hypothesis that the average age of smoking initiation in women is not different from that of men.

2.1.3 Mann-Whitney *U* Test

We can also use a non-parametric test if, for example, we do not believe we can assume normal distributions. This may be good in our case, since our distributions are not quite normal.

```
# Perform Mann-Whitney $U$ test
wilcox.test(x = SmokeAndDemo$NUMERICSMOKINGAGE[SmokeAndDemo$RIAGENDR == 2], y
= SmokeAndDemo$NUMERICSMOKINGAGE[SmokeAndDemo$RIAGENDR == 1])

##
##  Wilcoxon rank sum test with continuity correction
##
## data:  SmokeAndDemo$NUMERICSMOKINGAGE[SmokeAndDemo$RIAGENDR == 2] and Smok
eAndDemo$NUMERICSMOKINGAGE[SmokeAndDemo$RIAGENDR == 1]
## W = 719850, p-value = 9.012e-07
## alternative hypothesis: true location shift is not equal to 0
```

2.1.4 Wilcoxon signed rank test

The wilcoxon signed rank test can also be used if your data call for it. To do this we use the 'paired = TRUE' statement:

```
# Perform a Wilcoxon signed rank

wilcox.test(x = <dat$var1>, y = <dat$var2>, paried = TRUE)
```

2.1.5 Kruskal-Wallis test

Or, if appropriate the Kruskal-Wallis test. To do so we can first compare the medians for each group:

```
# Perform a Kruskal-Wallis test
```

```
by(data = SmokeAndDemo$NUMERICSMOKINGAGE, INDICES = SmokeAndDemo$RIDRETH1, FU
N = median, na.rm = TRUE)

## SmokeAndDemo$RIDRETH1: 1
## [1] 17
## ------------------------------------------------------------
## SmokeAndDemo$RIDRETH1: 2
## [1] 18
## ------------------------------------------------------------
## SmokeAndDemo$RIDRETH1: 3
## [1] 17
## ------------------------------------------------------------
## SmokeAndDemo$RIDRETH1: 4
## [1] 18
## ------------------------------------------------------------
## SmokeAndDemo$RIDRETH1: 5
## [1] 18
```

And then run the Kruskal-Wallis test.

```
kruskal.test(x = SmokeAndDemo$NUMERICSMOKINGAGE, g = SmokeAndDemo$RIDRETH1)

##
##  Kruskal-Wallis rank sum test
##
## data:  SmokeAndDemo$NUMERICSMOKINGAGE and SmokeAndDemo$RIDRETH1
## Kruskal-Wallis chi-squared = 34.995, df = 4, p-value = 4.657e-07
```

2.1.6 Correlation

In addition to these non-parametric tests, you can also run a correlation test using the steps provided below:

```
# Load in and merge dataset on body measurements
body <- read_xpt(file = "https://wwwn.cdc.gov/Nchs/Nhanes/2015-2016/BMX_I.XPT
")
BodyAndDemos <- merge(x = demoData, y = body, by = "SEQN")

# Clean income variable.
## Copy to new variable
BodyAndDemos$HHINCOME <- BodyAndDemos$INDHHIN2
## Replace missing values, as well as non-ordinal values (i.e. 12 and 13, "$2
0,000 and over" and "under $20,000" respectively)
BodyAndDemos$HHINCOME[BodyAndDemos$HHINCOME == 77 | BodyAndDemos$HHINCOME ==
99 | BodyAndDemos$HHINCOME == 12 | BodyAndDemos$HHINCOME == 13] <- NA
## Move top values down to the empy ranks
BodyAndDemos$HHINCOME[BodyAndDemos$HHINCOME == 14] <- 11
BodyAndDemos$HHINCOME[BodyAndDemos$HHINCOME == 15] <- 12

# Summarize the BMI data alone and by household income
summary(BodyAndDemos$BMXBMI)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   11.50   19.90   25.20   26.02   30.60   67.30     788

by(data = BodyAndDemos$BMXBMI, INDICES = BodyAndDemos$HHINCOME, FUN = summary
)

## BodyAndDemos$HHINCOME: 1
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   14.50   19.73   24.85   25.73   29.70   53.90      30
## ------------------------------------------------------------
## BodyAndDemos$HHINCOME: 2
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   13.80   19.52   25.55   26.43   31.02   64.60      35
## ------------------------------------------------------------
## BodyAndDemos$HHINCOME: 3
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   13.90   21.40   26.70   27.12   32.00   57.20      46
## ------------------------------------------------------------
## BodyAndDemos$HHINCOME: 4
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   12.70   20.35   25.60   26.47   30.95   55.40      55
## ------------------------------------------------------------
## BodyAndDemos$HHINCOME: 5
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   13.10   19.68   25.45   26.20   31.65   64.50      55
## ------------------------------------------------------------
## BodyAndDemos$HHINCOME: 6
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   13.40   20.20   25.70   26.42   30.80   61.10      68
## ------------------------------------------------------------
## BodyAndDemos$HHINCOME: 7
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   12.20   20.70   25.30   26.22   30.40   62.70      90
## ------------------------------------------------------------
## BodyAndDemos$HHINCOME: 8
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   13.70   20.20   25.20   26.39   31.00   67.30      53
## ------------------------------------------------------------
## BodyAndDemos$HHINCOME: 9
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   12.30   19.50   25.70   26.24   31.20   58.70      51
## ------------------------------------------------------------
## BodyAndDemos$HHINCOME: 10
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   13.70   19.70   25.35   25.63   30.02   64.50      34
## ------------------------------------------------------------
## BodyAndDemos$HHINCOME: 11
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   12.90   19.48   24.90   25.83   30.30   63.90      82
## ------------------------------------------------------------
```

```
## BodyAndDemos$HHINCOME: 12
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   11.50   18.80   24.00   24.85   29.30   54.20     116

# Perform a correlation test of Pearson's product-moment correlation
cor.test(x = BodyAndDemos$HHINCOME, y = BodyAndDemos$BMXBMI)

##
##  Pearson's product-moment correlation
##
## data:  BodyAndDemos$HHINCOME and BodyAndDemos$BMXBMI
## t = -5.4697, df = 7809, p-value = 4.647e-08
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.08384058 -0.03965535
## sample estimates:
##         cor
## -0.06177824

# Extract $R^2$, which is the square of the correlation coefficient
cor(x = BodyAndDemos$HHINCOME, y = BodyAndDemos$BMXBMI, use = "complete.obs")
^2

## [1] 0.00381655
```

2.2 Skills Check #2

Objectives. This skill check will help students to:

- Calculate basic measures of association and tests of significance.

Tasks. In skills check #1, we asked you to create two tables examining data from the NHANES demographic, questionnaire, and examination. Expanding and building upon this analysis, we will identify factors associated with hypertension by conducting the appropriate univariate tests of significance (i.e., chi-square, t-test, or similar non-parametric test) to test for differences in hypertensions across groups for each variable.

In doing so, be sure to report the following for each variable in the table below:

(1) your methods *(e.g., Which test did you conduct? Did you recategorize any variables or create a new variables from others?)*,
(2) your rationale for the methods you used *(e.g., Why a t-test as opposed to a non-parametric test? Why a categorical variable rather than a continuous variable?)*,
(3) the results from all analyses conducted for this skills check *(e.g., n, %, median, mean, z-score, t-score, p-value)*, and
(4) the correct interpretations of the results *(e.g., "Men were more/less likely to report having been diagnosed with hypertension (p = xxx)")*.

Table 1. Univariate Tests of Association Examining Factors Associated with Hypertension (Source: NHANES, 2015-2016)

Age
Age started smoking regularly
Age last smoked regularly
Sex
Race/Ethnicity
Education Level
Marital Status
Annual Income
Blood Pressure
Number of cigarettes smoked per day when they quit.
Average number of cigarettes per day over last 30 days.
Minutes spent doing vigorous activities, per day
Minutes spent doing moderate activities, per day
Minutes spent doing sedentary activity, per day
BMI
Waist Circumference

3 Regression

3.1 Linear regression

In R, regression models are generally stored, rather than merely run. Here we, given the correlation we observed between income level and BMI, regress BMI on income level using simple linear regression (`lm` stands for "linear model"). This requires the use of a formula argument. You can read more about formulae in the help documentation, but to be brief: A formula is usually written around a tilde, ~, which roughly means "by". For `lm`, the dependent variable is on the left side of the tilde, while the explanatory variable is on the right. Thus, you can read the formula, `BMXBMI ~ HHINCOME` as "BMI by income level". This is how R understands our mathematical equation:

$BMI_i = \alpha + \beta income_i + \epsilon_i.$

Since data is a separate argument and must be a data frame, we can simply name the variables and R will know it comes from the data frame in the `data` argument.

```
# Create a regression model.
BMImodel <- lm(formula = BMXBMI ~ HHINCOME, data = BodyAndDemos)
```

The model is now stored in an object called `BMImodel`. It is a list. Let's see what it contains using the `str` ("structure") function:

```
str(BMImodel)
```

```
## List of 13
##  $ coefficients : Named num [1:2] 27.155 -0.151
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "HHINCOME"
##  $ residuals    : Named num [1:7811] 2.16 4.25 2.4 16.76 -5.8 ...
##   ..- attr(*, "names")= chr [1:7811] "1" "2" "3" "4" ...
##  $ effects      : Named num [1:7811] -2297.45 -43.72 2.34 16.77 -5.83 ...
##   ..- attr(*, "names")= chr [1:7811] "(Intercept)" "HHINCOME" "" "" ...
##  $ rank         : int 2
##  $ fitted.values: Named num [1:7811] 25.6 26.5 26.4 25.6 26.1 ...
##   ..- attr(*, "names")= chr [1:7811] "1" "2" "3" "4" ...
##  $ assign       : int [1:2] 0 1
##  $ qr           :List of 5
##   ..$ qr   : num [1:7811, 1:2] -88.3799 0.0113 0.0113 0.0113 0.0113 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:7811] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:2] "(Intercept)" "HHINCOME"
##   .. ..- attr(*, "assign")= int [1:2] 0 1
##   ..$ qraux: num [1:2] 1.01 1.01
##   ..$ pivot: int [1:2] 1 2
##   ..$ tol  : num 1e-07
##   ..$ rank : int 2
##   ..- attr(*, "class")= chr "qr"
##  $ df.residual  : int 7809
##  $ na.action    :Class 'omit'  Named int [1:1733] 9 15 33 46 48 50 52 57 6
```

```
0 63 ...
##   .. ..- attr(*, "names")= chr [1:1733] "9" "15" "33" "46" ...
##  $ xlevels      : Named list()
##  $ call         : language lm(formula = BMXBMI ~ HHINCOME, data = BodyAndD
emos)
##  $ terms        :Classes 'terms', 'formula'  language BMXBMI ~ HHINCOME
##   .. ..- attr(*, "variables")= language list(BMXBMI, HHINCOME)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "BMXBMI" "HHINCOME"
##   .. .. .. ..$ : chr "HHINCOME"
##   .. ..- attr(*, "term.labels")= chr "HHINCOME"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(BMXBMI, HHINCOME)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "BMXBMI" "HHINCOME"
##  $ model        :'data.frame':   7811 obs. of  2 variables:
##   ..$ BMXBMI  : num [1:7811] 27.8 30.8 28.8 42.4 20.3 28.6 18.1 15.7 28 28
.2 ...
##   ..$ HHINCOME: num [1:7811] 10 4 5 10 7 11 6 12 7 6 ...
##   ..- attr(*, "terms")=Classes 'terms', 'formula'  language BMXBMI ~ HHINC
OME
##   .. .. ..- attr(*, "variables")= language list(BMXBMI, HHINCOME)
##   .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:2] "BMXBMI" "HHINCOME"
##   .. .. .. .. ..$ : chr "HHINCOME"
##   .. .. ..- attr(*, "term.labels")= chr "HHINCOME"
##   .. .. ..- attr(*, "order")= int 1
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
##   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, "predvars")= language list(BMXBMI, HHINCOME)
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. .. ..- attr(*, "names")= chr [1:2] "BMXBMI" "HHINCOME"
##   ..- attr(*, "na.action")=Class 'omit'  Named int [1:1733] 9 15 33 46 48
50 52 57 60 63 ...
##   .. .. ..- attr(*, "names")= chr [1:1733] "9" "15" "33" "46" ...
##  - attr(*, "class")= chr "lm"
```

That's a lot of information. In the help documentation on `lm`, the Value section describes each of the top-level elements. While you don't need to understand every element of this model, it is helpful to know that the object `BMImodel` contains a lot of information about your model that you can explore: coefficients (i.e. the α and β of your equation), residuals (i.e., the ϵ_i of your equation), fitted values (i.e. the $\hat{BMI}_i$ of your model), and even the data

you fed to the function (including the BMI_i and $income_i$ used). Fortunately, `summary` knows how to deal with this model, so we'll just use it to see some basic info.

```
summary(BMImodel)
```

```
##
## Call:
## lm(formula = BMXBMI ~ HHINCOME, data = BodyAndDemos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.896  -6.148  -0.801   4.648  41.355
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 27.15462    0.23046  117.83  < 2e-16 ***
## HHINCOME    -0.15117    0.02764   -5.47 4.65e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.992 on 7809 degrees of freedom
##   (1733 observations deleted due to missingness)
## Multiple R-squared:  0.003817,   Adjusted R-squared:  0.003689
## F-statistic: 29.92 on 1 and 7809 DF,  p-value: 4.647e-08
```

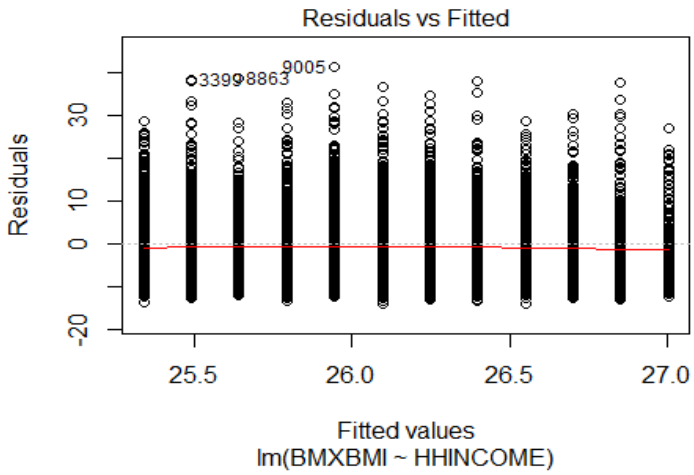
Here `summary` gave us the formula we specified, some information on the residuals, our coefficients with their standard errors and significance tests, and some diagnostic results. It appears that income level significantly explains variation in BMI and that for each income level increase there is an average decrease of 0.15117 BMI. The p-value is well below the 0.001 threshold. But is this a good model of BMI?

3.1.1 Linear regression diagnosis

In our model, we saw that income level did indeed explain some variation in BMI, but how much of BMI does it explain? We already knew the answer to this question: it is the correlation coefficient squared, reported here as "Multiple R-squared": 0.003817. So less than 1%. Not only does our model explain almost none of the variation in BMI, it leaves so much unexplained that the potential for omitted variable bias is immense. The relationship between income and BMI could actually be explained better by some other variable and may therefore be a spurious correlation.

There are other potential problems, including heteroskedasticity. Regression analysis assumes normally distributed errors across the fitted values, or homoskedasticity. There is a simple way to check for heteroskedasticity The `plot` function will create a bunch of plots about our model, but we just want the first one so we'll use `which = 1`:

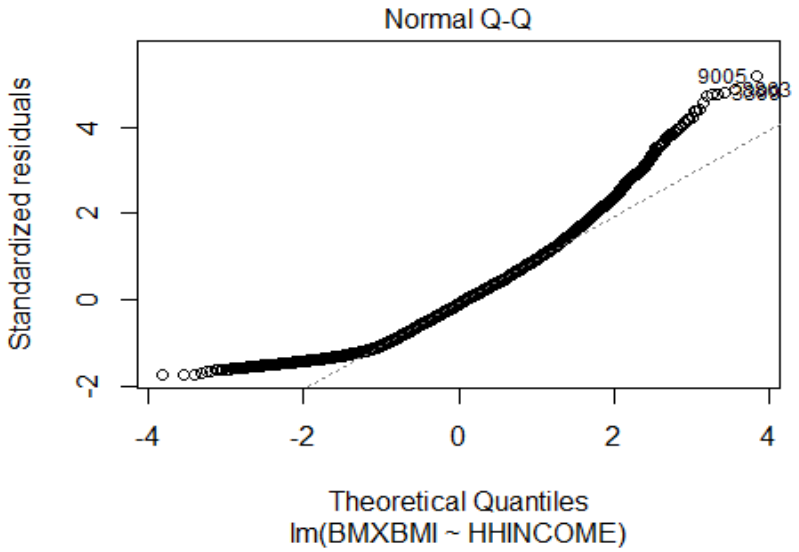
```
# Create a residual vs. plot for our lm object.
plot(BMImodel, which = 1)
```



Fortunately, across our fitted values the spread of the residuals appears fairly constant. The red line, a running average, also appears rather flat. A few outliers are noted, but heteroscedasticity doesn't appears to be a major problem.

Let's also check to see if our model approximates a normal distribution overall using the Q-Q plot.

```
# Create a residual vs. plot for our lm object.
plot(BMImodel, which = 2)
```

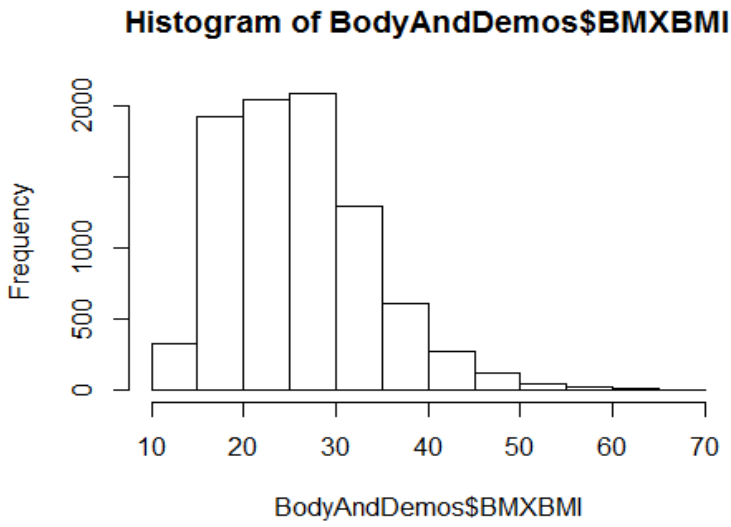


If our data were normal, the points would fall roughly along the dotted line. It appears that this is not exactly the case; this is a subjective decision, but while we can tolerate some abnormality in our data this does seem to violate this assumption. We would need to refine this model as we seem to have some extreme values (likely from the BMI variable).

3.2 Logistic regression

Linear models are created using the `lm` function, while logistic models rely on the `glm` or "generalized linear model" function. For this, we require a dependent variable in a logical class, meaning it takes the value `TRUE` or `FALSE`. For this exercise, let us create one based on whether a person is overweight or not.

```
# Look at the distribution of BMIs
hist(BodyAndDemos$BMXBMI)
```




```
# Create our new variable
BodyAndDemos$OVERWEIGHT <- BodyAndDemos$BMXBMI >= 25

#How many people are overweight in our sample?
summary(BodyAndDemos$OVERWEIGHT)

##    Mode   FALSE    TRUE    NA's
## logical    4246    4510     788
```

Over half of our sample are considered "overweight" by this standard. By doing this, we have reduced the information in our data but avoided some of the problems associated with the previous variable (normality, etc., which is good). Let's run the model. We need to specify which distribution `glm` will rely on, which for logistic regression is the "binomial" distribution.

```
# Create a gender factor. This is easier to interpret in the logistic regress
ion output
BodyAndDemos$GENDER <- as.factor(BodyAndDemos$RIAGENDR)

# Create a logistic model. To use multiple variables, separate them by `+`
BMImodel2 <- glm(formula = OVERWEIGHT ~ HHINCOME + GENDER, family = "binomial
```

```
", data = BodyAndDemos)
summary(BMImodel2)

##
## Call:
## glm(formula = OVERWEIGHT ~ HHINCOME + GENDER, family = "binomial",
##     data = BodyAndDemos)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.310  -1.197   1.064   1.158   1.230
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.281460   0.063013   4.467 7.94e-06 ***
## HHINCOME    -0.033759   0.006945  -4.861 1.17e-06 ***
## GENDER2      0.058131   0.045379   1.281      0.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10823  on 7810  degrees of freedom
## Residual deviance: 10797  on 7808  degrees of freedom
##   (1733 observations deleted due to missingness)
## AIC: 10803
##
## Number of Fisher Scoring iterations: 3

# Confidence intervals for a linear model
confint(BMImodel)

##                  2.5 %      97.5 %
## (Intercept) 26.7028559 27.60639222
## HHINCOME    -0.2053467 -0.09699241

# Return odds ratios for logistic model
exp(coefficients(BMImodel2))

## (Intercept)    HHINCOME     GENDER2
##   1.3250633   0.9668042   1.0598536

# Odds ratio confidence intervals for the logistic model
exp(confint(BMImodel2))

##                 2.5 %    97.5 %
## (Intercept) 1.1712370 1.4994421
## HHINCOME    0.9537214 0.9800446
## GENDER2     0.9696623 1.1584495
```

3.3 Skills Check #3

Deliverables. For this skills check you will turn in the fully-annotated R code you used to answer the questions below along with a brief write-up providing and interpreting the statistical results for all requested analyses.

Objectives. This skillscheck will help students to:

- Conduct a regression analysis.

Tasks. In previous skills checks, you provided descriptive results from the NHANES survey and identified factors associated with hypertension at the univariate level. In this skills check, you will construct a multivariable model predicting hypertension among respondents of the NHANES survey. In completing this skills check, it is up to you which specific variables to include and exclude from the analysis. You should, however, provide justifications for these decisions based on appropriate univariate tests. After you conduct your multivariable regression model, you should report your results in a single table which provides the adjusted odds ratio with 95% confidence intervals.