Lab #3, Part #1 – Standardizing Data and Binomial Simulation

We recently discussed various ways in which data can be distributed, and the importance of standardization of the normal distribution in understanding percentiles and their relationship to area under a histogram. In this lab, we'll use a large survey data set to look at some distributions and transform data in R Studio. For this lab, include answers to all questions and all figures within the lab write-up that you'll submit as part of Lab 3 with the associated problem numbers (2.1, 3.1, 3.2, etc.). Be sure to also include all of your R code.

The data you'll be using today is from the Centers for Disease control called the Behavioral Risk Factor Surveillance System (BRFSS). The BRFSS is an annual telephone survey of 350,000 people in the United States. As its name implies, the BRFSS is designed to identify risk factors in the adult population and report emerging health trends. For example, respondents are asked about their diet and weekly physical activity, their HIV/AIDS status, possible tobacco use, and even their level of healthcare coverage. The BRFSS Web site (http://www.cdc.gov/brfss) contains a complete description of the survey, including the research questions that motivate the study and many interesting results derived from the data.

We will focus on a random sample of 20,000 people from the BRFSS survey conducted in 2000. While there are over 200 variables in this data set, we will work with a small subset.
This returns the names `genhlth`, `exerany`, `hlthplan`, `smoke100`, `height`, `weight`, `wtdesire`, `age`, and `gender`. Each one of these variables corresponds to a question that was asked in the survey. For example, for `genhlth`, respondents were asked to evaluate their general health, responding either excellent, very good, good, fair or poor. The `exerany` variable indicates whether the respondent exercised in the past month (1) or did not (0). Likewise, `hlthplan` indicates whether the respondent had some form of health coverage (1) or did not (0). The `smoke100` variable indicates whether the respondent had smoked at least 100 cigarettes in her lifetime. The other variables record the respondent's `height` in inches, `weight` in pounds as well as their desired weight, `wtdesire`, `age` in years, and `gender`.

Activity #1: Load the Data and Set Working Directory.
The first thing we'll want to do is load up the data and set your working directory. You'll want to download the data directly with the `source()` function, and then set your working directory to your Lab 3 folder (hopefully you've already made one in your lab directory). Today, we'll use the directory to directly save figures as picture files instead of copying them over from R Studio by hand.

```
#Load the data
source("http://www.openintro.org/stat/data/cdc.R")

#Set working directory
setwd(“c:/…/Lab 3”)

#take a look at the data
head(cdc)
nrow(cdc)
```

Activity #2: Summarizing the Data.
As we've done in previous labs, go ahead and summarize each of the variables in the data using the `summary()`, `sd()`, and `table()` functions. Note that for categorical variables, you should be using `table()` and reporting the proportion of observations in each category. To get percentages in each category, instead of just counts, you can divide the table by the number of rows in the data (20,000) like so:

```
#make a table of genhlth with percentages by category
table(cdc$genhlth)/20000
```

In reporting results, make a nice and tidy table in your lab write-up that looks like what you see below. As we move forward in the course, we'll continue to work on presenting information neatly and clearly. Give the table a title like "Summary Statistics for All Variables."

2.1. Summary Statistics for All Variables

Variable	Mean (%)	SD	Min	Max
genhlth				
Excellent				
Very Good				
Good				
Fair				
Poor				
exerany				
Yes				
No				
hlthplan				
Yes				
No				
smoke100				
Yes				
No				
height				
weight				
wtdesire				
age				
gender				
Female				
Male				

Activity #3: Making Histograms.
We'll again use the `hist()` function today, this time to visualize the numeric values in our data. However, we'll save them as a .png file instead of copying and pasting them into your lab write-up. To do this, we'll use the `png()` function. You'll call the `png()` function prior to making your histogram, then turn it off after your code using `dev.off()`. Let's start with the `weight` variable with the code below and name it "weightHist.png":

```
#make a histogram with weight and save to a png file
png(file = "weightHist.png")

hist(cdc$weight, main = "Histogram of Weight",
     xlab = "Weight (lbs.)", ylab = "Frequency", col = "steelblue")

dev.off()
```

The `png()` function simply initializes a new file, which R will write the rest of the code below that line into. Since you only want to put the single histogram in there, you turn off that new file with `dev.off()`.

3.1. Go to your directory and open up the .png file to make sure it looks right. Then do the same for the other numeric variables in the data. Remember to include titles and axis labels (with measurement units) for each of your histograms. Feel free to add color if you want and be sure that each histogram file you make has a different name.

Activity #4: A New Variable.
Notice that there is a `weight` and desired weight (`wtdesire`) variable in the data. Let's make a new variable that is the difference between current weight and desired weight called `weightDiff`. You can do this with the following code:

```
#make weightDiff variable
cdc$weightDiff <- cdc$weight - cdc$wtdesire
head(cdc)
```

Hopefully you'll see it there tacked onto the end of your data set when you look at it.

4.1. Go ahead and create another histogram for this new variable and save it as a .png file like the others.

Activity #5: Making Z-Scores.
Recall that you can make a z-score by subtracting the mean value, μ, from the given value, x_i, and dividing by the standard deviation, σ. We can do this in R easily, as it serves as a calculator. Since we have a data set, we should be able to do this for entire columns of the data. Start with the code

below for the `height` variable, and then do the same for all other numeric variables in the `cdc` data set.

```
#make z-scores of each as new variables
cdc$height_z <- (cdc$height – mean(cdc$height))/sd(cdc$height)
head(cdc)
```

It turns out that R makes it even easier to calculate z-scores from our data. There is a function called `scale()` that allows us to do the same operation that we used arithmetic for from our z-score equation. Go ahead and make another z-score for the `weightDiff` variable now, this time using the `scale()` function. Call it `weightDiff_z2`.

5.1. Does it look the same as the one you calculated manually?

```
cdc$weightDiff_z2 <- scale(cdc$weightDiff)
head(cdc)
```

Activity #6: Making Histograms of Standardized Data.
Now that you have standardized versions of your data, I want you to make a number of histograms that directly compare what these distributions look like, now on the same standardized scale. You'll create a single .png file for this, and make use of the `par(mfrow = c())` functionality that we've used before. Make a 2x2 matrix of histograms for `height, weight, wtdesire`, and `age`.

The code below will get you started and notice that I'm limiting the x-axis bounds to be between -4 and 4 with the `xlim = c()` argument (do this for all histograms). Also note that I'm using new arguments in the `png()` function, which makes the window larger to fit all 4 histograms.

```
#make a 2x2 histogram matrix
png(file = "standardizedHist.png")
par(mfrow = c(2,2)

hist(cdc$height_z, main = "Histogram of Standardized Height",
     xlab = "Height (in.)", ylab = "Frequency", col = "steelblue",
     breaks = 10, xlim = c(-4,4))

hist(cdc$weight_z, …

dev.off()
```

6.1. Describe each distribution and compare them to one another in your lab write-up. Do any of the distributions look normally distributed? Are they skewed or symmetric? Does the standardization help with comparing the different variables? Why or why not?

Activity #7. Percentiles and Area Under the Curve.
Given what we see in these data, let's stick with using the weight variable for now. Using the `head()` function, look at the **first 6 rows** of the data. For each of the `weight_z` observations in these first 6 rows, calculate the following with the `pnorm()` function:

7.1. The percentage of weights in the sample that are below the given weight.
7.2. The percentage of weights in the sample that are above the given weight.
7.3. The percentage of weights in the sample that are between the given weight and the mean weight from the sample.
7.4. Given a random draw from the sample, what is the probability that someone at or above this weight would be chosen?

To get you started, here is the code to get the answer to #1 above for the first row:

```
#get the percent below the given observation
pnorm(cdc$weight_z[1])
```

Put these in a neat and tidy table. Once you finish this, answer the following question below:

7.5. Given the distribution of weight that you see in the histogram, do you think these percentiles calculated from the standard normal distribution are applicable to the weight distribution in this sample? Why or why not? How else might you calculate percentiles from the sample, if the standard normal distribution probability table does not work?

Activity #8. Let's Get Binomial.
Let's move onto thinking about the binomial distribution. For this, we'll move away from the numeric variables and use the `exerany` variable, which is a self-report measure indicating whether the participant exercised at all in the past month. An answer of "Yes" is coded as 1, while an answer of "No" is coded as 0. We'll treat the "Yes" answers at a success.

Let's say we want to know the probability that, given randomly choosing 6 rows in the data, that exactly 2 of those people will have exercised within the past month. First use the `table()` function in R to get the proportion of successes in the data, and assume this is the population rate.

8.1. Then, use the formula below from the lectures to calculate the result. Report this in percentage terms, rounded to 2 decimal places (i.e. 5.79%).

$$\frac{n!}{k!\,(n-k)!}p^k(1-p)^{n-k}$$

Activity #9: Simulating the Result.

Now that you've calculated this numerically, let's make sure we did things about right by simulating the same result using R. The hope is that, given enough samples, we should end up with a proportion of sample observations that have approximately the same proportion that we calculated in the previous problem.

In this case, we'll return to the `sample()` function we used in the previous Kobe Bryant lab to draw lots and lots of samples from our data. As with the previous question, you'll want each sample to be N = 6. Use the code below to first take a single sample:

```
#one sample from the data
s1 <- sample(cdc$exerany, 6, replace = TRUE)
```

9.1. Did you get exactly 2 successes? Explain why you think this happened.

Next, use a new function called replicate() to do this 100 times with the following code. This function simply tells R to do another function over and over again the number of times you tell it (the number of replications will be the first argument you enter in the function).

```
s100 <- replicate(100, sum(sample(cdc$exerany, 6,
     replace = TRUE)))
```

Notice that instead of just using sample, I sum the number of successes for each sample and use this as the number of interest. Remember, we're interested in knowing how often 2 comes up.

9.2. How often did you get 2 successes this time (report in percentage terms rounded to 2 decimal places)? You can use the `table()` function to show the breakdown across different total successes. Is this exactly what you found in Activity #8? If so, why? If not, why not?

Now, repeat the above, but this time do the sampling 1,000,000 times and call it `sMIL`. This will take a few seconds, but should not take long to run.

9.3. What is the percentage of samples with N = 6 that you get exactly 2 successes? How does this compare to the value you calculated in Activity #8?

9.4. Repeat Activity #8, but this time with exactly 1, 3, 4, 5, and 6 successes, respectively. Does the simulation show those proportions for each? Explain in your own words what is going on. Include a histogram of `sMIL` in your answer.