Homework #3

Due: Tuesday, October 12 @ 6pm [35*points*]

Problem 1: [6*points*]

Recall the set of measurements from HW1:

52	16	180	1	199	8	3	23	156	63
808	25	5	554	85	1	64	52	7	192

a. Write an R expression to generate a vector of these values and assign it to the variable x [1*point*]

```
x <- 52, 16, 180, 1, 199, 8, 3, 23, 156, 63, 808, 25, 5, 554, 85, 1, 64, 52, 7, 192
```

b. Read the help page for the `rep()` function. Using `rep()` and `c()`, write an R expression to generate those values in any order and assign them to `y`. Show `y` after the assignment. [1*point*]

```
# get help page for rep
#?rep
y <- c(rep(1, 2), rep(52,2), 16, 180, 199, 8, 3, 23, 156, 63, 808, 25, 5, 554, 85, 64, 7, 192)
```

c. Read the help pages for `any()` and `all()`, and briefly describe what they do. [1*point*]

"any()" - given a set of logical vectors, is at least one of the values true? "all()" - are all values true

d. What do you expect to get from `all(y==x)`, and why? Check your intuition in R. [1*point*]

all(y==x) is probably going to be false because it is checking if the first element of x == first element of y, then x[2] == y[2] and if x and y have the same value but in different order, they will not pass the "all" threshold

```
all(y==x)
```

```
## [1] FALSE
```

e. Suppose you wanted to see if two vectors contained exactly the same values, *regardless of the order they were in*. How might you go about doing that? Write an R expression to test `x` and `y` this way. [2*points*]

Many ways to solve this problem. Anything that works goes! Some examples:

```
x %in% y
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE TRUE
```

```r
intersect(x, y)
```

```
##  [1]  52  16 180   1 199   8   3  23 156  63 808  25   5 554  85  64   7 192
```

```r
setdiff(x, y)
```

```
## numeric(0)
```

```r
sort(x) == sort(y)
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE TRUE
```

```r
# this is probably my favorite way... since there are replicate data we don't just want to match values
identical(sort(x), sort(y))
```

```
## [1] TRUE
```

Problem 2: [*9points*]

Using the same data as above,

a. Compute: [*3points*]

   - The mean of `x`
   - The median of `x`
   - The sample standard deviation of `x`
   - The mean and sample SD of `2x`
   - The mean and sample SD of `x + 10`
   - The mean and sample SD of `2x + 10`
   - The mean and sample SD of `2(x + 10)`

```r
mean(x)
```

```
## [1] 124.7
```

```r
median(x)
```

```
## [1] 52
```

```r
sd(x)
```

```
## [1] 205.7382
```

```r
mean(2*x) # note this equals 2*(mean(x))
```

```
## [1] 249.4
```

```r
sd(2*x) # note this equals 2*(sd(x))
```

```
## [1] 411.4765
```

```r
mean(x + 10) # note this equals mean(x) + 10
```

```
## [1] 134.7
```

```r
sd(x + 10) # note this equals sd(x)!!!
```

```
## [1] 205.7382
```

```r
mean(2*x+10)
```

```
## [1] 259.4
```

```r
sd(2*x + 10)
```

```
## [1] 411.4765
```

```r
mean(2*(x + 10)) # order of operations!
```

```
## [1] 269.4
```

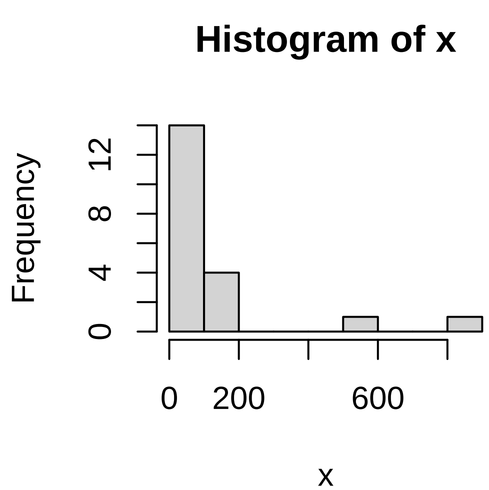
```r
sd(2*(x + 10))
```

```
## [1] 411.4765
```

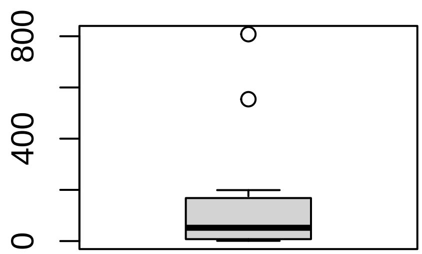
b. Explore the help pages and online materials to figure out how to plot: [*2points*]

- A histogram of `x`
- A boxplot of `x`

```r
hist(x)
```



```r
boxplot(x)
```

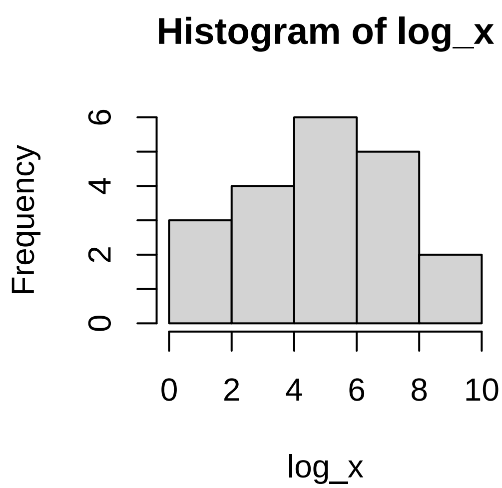


c. Consider now the $log_2(x)$ for this data. Write an R expression that computes it assign it to a new variable of your choosing. [$1points$]

```r
log_x <- log(x, base = 2)
```

d. Plot a histogram of your new $log_2(x)$ variable. How does it compare to the histogram you got for `x`? [$1points$]

```r
hist(log_x)
```



The log transformed data is no longer right skewed. It looks more normally distributed.

e. Suppose we added two additional observations to `x`, both of which were exactly equal to the mean of `x` (as obtained in part (a) above). Write an R expression to have `x` include those two additional values. Then compute the new mean and SD, as you did in HW1. Are they what you expected? [$2points$]

```r
mean(c(x, mean(x), mean(x)))
```

```
## [1] 124.7
```

```r
sd(c(x, mean(x), mean(x)))
```

```
## [1] 195.6961
```

Problem 3: [8*points*]

In this problem, we'll explore how R deals with missing data. Suppose you had a vector `y <- c(1,1,2,3,4,10)`: [1*point*/*ea*]

a. Write an expression to set the element of `y` that is equal to 10 to `NA`.

```r
y <- c(1,1,2,3,4,10)

y[y==10] <- NA
```

b. Imagine the second element of `y` was erroneous. Give two ways one might get rid of it.

```r
y[2] <- NA
```

```r
y[2] <- NA
y <- y[-2]
y <- y[1,3:5]
```

c. Show `y`. Without using R, what do you expect the mean of `y` to be?

1, NA, 2, 3, 4, NA

I would expect the mean of y to be 2.5, $(1 + 2 + 3 + 4)/4$

d. What does `mean(y)` give? How does this compare to your expectation above?

```r
mean(y)
```

```
## [1] NA
```

Looks like the NA values are messing up the mean calculations

e. Read the help page for `mean()` and give an expression for the mean of the non-missing values of y.

```r
mean(y, na.rm = T)
```

```
## [1] 2.5
```

f. Write an expression to test whether *all* elements of `y` are greater than 1.

```r
# MUST use the na.rm = T option
all(y > 1, na.rm = T)
```

```
## [1] FALSE
```

g. Write an expression to test whether *any* element of `y` is `NA`.

```r
any(is.na(y))
```

```
## [1] TRUE
```

h. Write an expression to count the number of elements of `y` that are not `NA`.

```r
sum(!is.na(y))
```

```
## [1] 4
```

Problem 4: [$4 points$]

We will continue to use y in from the previous problem in this exercise.

a. Suppose you were to take many, many random samples from the non-`NA` elements of `y` (with replacement). On average, what fraction of them would you expect to be > 2? [$1 point$]

Probability = relative frequency, there are 2 values greater than 2 in vector y out of 4 total non-NA values, so with a probability of 0.5

b. Write an expression to take a sample, with replacement, of size 20 from the non-`NA` elements of `y` [$2 points$]

```r
set.seed(1)
s <- sample(y[!is.na(y)], 20, replace = T)

# count how many are > 2
sum(s > 2)
```

```
## [1] 7
```

c. How many of them did you expect to be > 2? How many actually were > 2? [$1 point$]

I expected 10/20 to be greater than 2 and I observed 7 (Note: this number will vary from time to time as these are RANDOM samples.)

Problem 5: [$8 points$]

As we discussed in class, R has a number of probability distribution functions built in. You can see the list of them with ?distributions . Here, we'll use the functions for the normal distribution, abbreviated `*norm` (i.e. `pnorm()`, `dnorm()`, `qnorm()`, and `rnorm()`).

Remember: If you don't specify mean or sigma when you call these functions, it assumes a standard normal with `mean=0` and `sigma=1` by default. Hence, `rnorm(10)` will get you 10 random numbers from a a standard normal.

Let's practice using these by computing the following. Give both the R code you used and the numerical value in your answers. Be sure to think about what you get - do the results seem reasonable (e.g., no probabilities > 1, values that "make sense" given the means & SD's you're putting in, etc.). [$1 point/ea$]

NOTE: this notation is N(mean, variance) NOT N(mean, SD) like I said in class. For grading purposes, either will be considered correct. But please keep in mind for future.

a. What is the probability that `x ~ N(10,2)` will be ≤ 10?

```
pnorm(q = 10, mean = 10, sd = sqrt(2))
```

```
## [1] 0.5
```

b. What is the probability that `x ~ N(-1, 1)` will be greater than 1.3?

```
1-pnorm(q = 1.3, mean = -1, sd = sqrt(1))
```

```
## [1] 0.01072411
```

```
# or:
#pnorm(q = 1.3, mean = -1, sd = sqrt(1), lower.tail = F)
```

c. What is the probability that `x ~ N(1,1)` will be more extreme than ± 2 (i.e. greater than 2 or less than -2)?

```
pnorm(q = -2, mean = 1, sd = sqrt(1)) + pnorm(q = 2, mean = 1, sd = sqrt(1), lower.tail = F)
```

```
## [1] 0.1600052
```

d. What is the probability that `x ~ N(0,3)` falls between 2 and 4? (*Hint: consider the total area under the curve and ask where x doesn't fall.*)

```
pnorm(q = 4, mean = 0, sd = sqrt(3)) - pnorm(q = 2, mean = 0, sd = sqrt(3))
```

```
## [1] 0.1136459
```

e. Assuming `x ~ N(-3,2)`, what is the **q** such that half the area under the curve lies to the right of q?

```
qnorm(p = 0.5, mean = -3, sd = sqrt(2))
```

```
## [1] -3
```

f. Assuming **x** is normally distributed with mean 0 and SD=0.3, what is **q** such that $P(x \geq q) = 0.05$?

```
qnorm(p = 0.95, mean = 0, sd = 0.3)
```

```
## [1] 0.4934561
```

```
# or
#qnorm(p = 0.05, mean = 0, sd = 0.3, lower.tail = F)
```

g. Consider a **z score**. What is **q** such that the probability that $z \geq q$ OR $z \leq (-q)$ is 0.05?

```r
# z score means mean = 0, sd = 1
# prob on both the right and left tails added = 0.05, so the prob of just the left tail is 0.025
qnorm(p = 0.05/2, mean = 0, sd = 1, lower.tail = F)
```

```
## [1] 1.959964
```

h. Assuming `x ~ N(0, 1)`, for what value **q** is 50% of the under the curve in a band between -q and +q?

```r
# same reasoning as above, if we want the middle 50% then we can use the lower 25% to find the correct
qnorm(p = 0.5/2, mean = 0, sd = sqrt(1), lower.tail = F)
```

```
## [1] 0.6744898
```