Biometry: Midterm Study Guide

Robert Dellinger

9/28/2022

Lecture Review

Basic Statistics and Introduction to R

(1) (4 pts) The following data are numbers of protozoa sampled from a microcosm and counted on a hemacytometer: 6, 11, 4, 5, 7, 3, 5, 1, 5, 6. Calculate the statistics listed below for this sample of protozoan densities. Do these calculations "by hand" using a calculator.

```
#clearing the environment
rm(list=ls())

#calculating the mean
sum.of.all.data.points = 6 + 11 + 4 + 5 + 7 + 3 + 5 + 1 + 5 + 6
number.of.data.points =length(c(6, 11, 4, 5, 7, 3, 5, 1, 5, 6))

Mean = sum.of.all.data.points/number.of.data.points
print(Mean)
```

```
## [1] 5.3
```

```
#calculating the variance
sum.of.squared.difference.from.mean = (6-Mean)^2 + (11-Mean)^2 + (4-Mean)^2 + (5-Mean)^2 + (7-Mean)^2 +

Variance = sum.of.squared.difference.from.mean/(number.of.data.points -1)
print(Variance)
```

```
## [1] 6.9
```

```
#calculating the standard deviation
Standard.Deviation = sqrt(Variance)
print(Standard.Deviation)
```

```
## [1] 2.626785
```

```
#calculating the standard error
Standard.Error = Standard.Deviation/sqrt(number.of.data.points)
print(Standard.Error)
```

```
## [1] 0.8306624
```

```
#calculating the coefficient of variation
Coefficient.of.Variation = (Standard.Deviation/Mean) #to get percentage *100
print(Coefficient.of.Variation)
```

```
## [1] 0.4956198
```

```
#finding the median (value that is directly in the middle after being ordered)
Median <- median(c(6, 11, 4, 5, 7, 3, 5, 1, 5, 6))
print(Median)
```

```
## [1] 5
```

```
#finding the mode (value that is repeated the most)
#creating a function for mode
mode <- function(v) {
 uniqv <- unique(v)
 uniqv[which.max(tabulate(match(v, uniqv)))]
}

Mode <- mode(c(6, 11, 4, 5, 7, 3, 5, 1, 5, 6))
print(Mode)
```

```
## [1] 5
```

(2a) (4 pts) Being the insightful biologist that you are, you notice that protozoa seem more dense at the bottom of the microcosm, perhaps because there is more food available there. You want to know if there is statistical support for this casual observation. You sample 20 replicate microcosms and measure the densities of protozoa. In 10 of the microcosms, you take the sample from the top and in the other 10 microcosms, you take the sample from the bottom. The data are as follows:

Top (# per uL): 3, 1, 0, 5, 4, 3, 6, 3, 4, 7

Bottom (# per uL): 3, 12, 3, 4, 7, 8, 7, 5, 15, 9

Using R, calculate the following statistics for both top and bottom: Mean, Standard deviation, Variance, 95% Confidence Interval.

```
#clearing the environment
rm(list=ls())

#Wrangling the Data
Top<- c(3,1,0,5,4,3,6,3,4,7)
Bottom<-c(3,12,3,4,7,8,7,5,15,9)
dataframe<-as.data.frame(cbind(Top, Bottom)) #binds strings together into a data frame
#glimpse(dataframe)
dataframe<-gather(data=dataframe, key=Location, value=Protozoa, Top:Bottom, factor_key=TRUE) # convert
```

```r
#calculating statistics for data using functions and the summarize() command

se <- function(x) (sd(x) / sqrt(length(x))) #creating function for standard error

mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
} #creating function for Mode

#summarized statistics by group
summary.statistics <- dataframe %>%
  group_by(Location) %>%
  summarize(Mean=mean(Protozoa), Standard.Deviation=sd(Protozoa), Variance=var(Protozoa), Standard.Erro
print(summary.statistics)
```

```
## # A tibble: 2 x 7
##   Location  Mean Standard.Deviation Variance Standard.Error Median  Mode
##   <fct>    <dbl>              <dbl>    <dbl>          <dbl>  <dbl> <dbl>
## 1 Top        3.6               2.12     4.49          0.670    3.5     3
## 2 Bottom     7.3               3.92    15.3           1.24     7       3
```

```r
#Constructing a Confidence Interval
    # 99%CI = 2.58*sd/sqrt(n) or 2.58*se
    # 95%CI = 1.96*sd/sqrt(n) or 1.96*se
    # 90%CI = 1.65*sd/sqrt(n) or 1.65*se

topCI<-1.96*summary.statistics[1,5] #selecting for row (1st), then column (5th)
bottomCI<-1.96*summary.statistics[2,5] #selecting for row (2nd), then column (5th)

print(topCI)
```

```
##   Standard.Error
## 1       1.313184
```

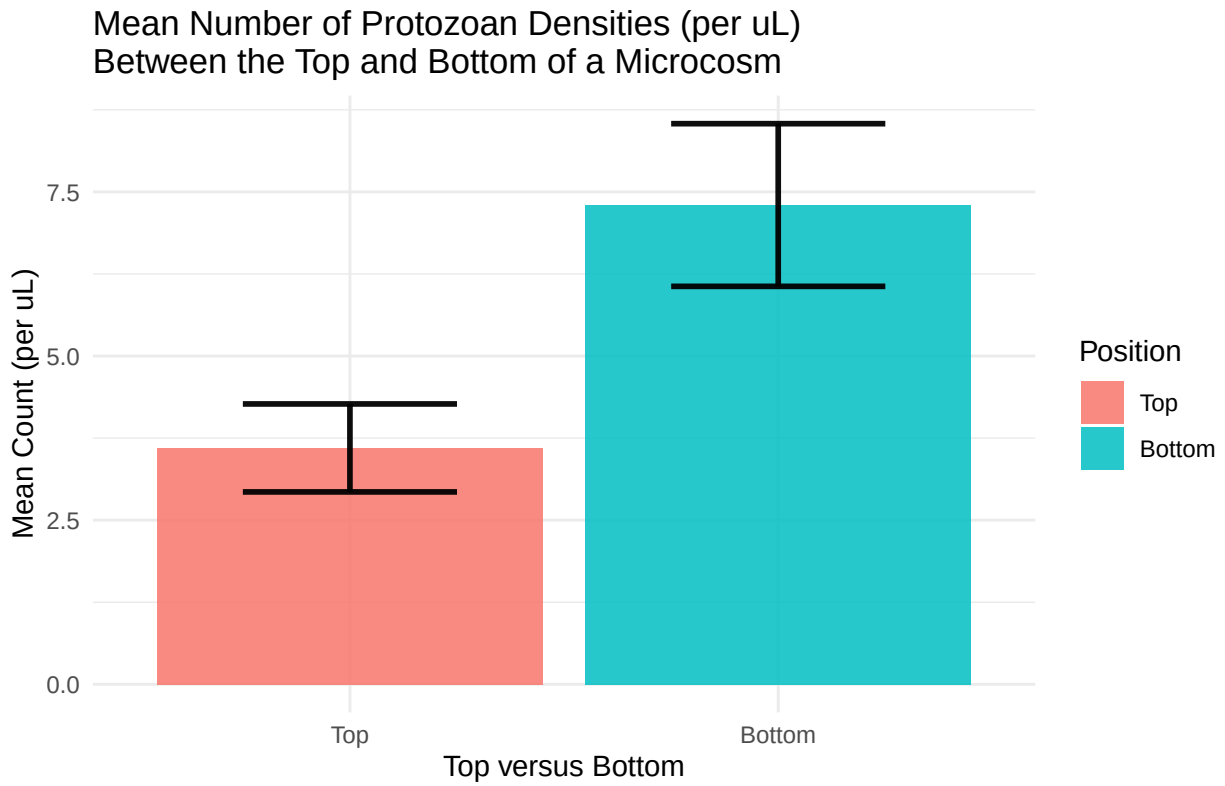
```r
print(bottomCI)
```

```
##   Standard.Error
## 1       2.427905
```

(2b) (5 pts) Make a publication-quality bar graph in R that presents means and standard errors for each group (top vs bottom). Provide a figure legend that describes the graph and includes a statement about whether you think protozoa densities differ between the top and bottom of the microcosm.

```r
graph.data <- summary.statistics %>%
  select(Location, Mean, Standard.Error)
#glimpse(graph.data)

ggplot(graph.data) +
  geom_bar(aes(x=Location, y=Mean, fill=Location), stat="identity", alpha=0.85) +
```

```
  geom_errorbar(aes(x=Location, ymin=Mean-Standard.Error, ymax=Mean+Standard.Error),
                position=position_dodge(0.9), width=0.5, colour="black", alpha=0.95, size=1)+
  theme(legend.position = "right", legend.title=element_text(size=20),
        legend.text=element_text(size=14)) +
  theme_minimal() +
labs(y = "Mean Count (per uL)", x="Top versus Bottom", title = "Mean Number of Protozoan Densities (per
```



Mean number of protozoan between the top and bottom of a microcosm differ.
Given that the error between bars plots do not overlap, the differences are statistically relevant.

(3a) (3 pts) The Excel file named "kelp bass gonad mass" contains the weights of gonads from several hundred kelp bass collected by Dr. Mark Steele's lab. Estimate the mean, median, s2, s, CV, skewness, and kurtosis.

```
#clearing the environment
rm(list=ls())

#read in data
dataframe <- read_csv(here("Data", "Kelp_Bass_Gonad_Data.csv"))
```

```
## Rows: 599 Columns: 1
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (1): gonad_mass
##
## i Use `spec()` to retrieve the full column specification for this data.
```

```
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
#glimpse(dataframe)

se <- function(x) (sd(x) / sqrt(length(x))) #creating function for standard error

cv <- function(x) (sd(x) / mean(x)) #creating function for cv

#summarize variables into a data frames
summary.statistics <- dataframe %>%
  summarize(Mean=mean(gonad_mass), Median=median(gonad_mass),  Variance=var(gonad_mass),
            Standard.Deviation=sd(gonad_mass), Standard.Error=se(gonad_mass), Coefficient.Variation=cv(g
            Kurtosis=kurtosis(gonad_mass))
print(summary.statistics)
```

```
## # A tibble: 1 x 8
##    Mean Median Variance Standard.Deviation Standard.Error Coefficient.Variation
##   <dbl>  <dbl>    <dbl>              <dbl>          <dbl>                 <dbl>
## 1  8.24   6.42     57.9               7.61          0.311                 0.924
## # ... with 2 more variables: Skewness <dbl>, Kurtosis <dbl>
```

(3b) (3 pts) What effect would adding 5.0 to each observation of gonad mass have on the values of the mean, median, s2, s, CV, skewness, and kurtosis? (You don't need to show the new values, but just describe how the statistics have changed.)

```
#Altering data by ading 5 to each observation
dataframe$gonad_mass_5<-dataframe$gonad_mass+5

#summarize variables into a data frames
summary.statistics <- dataframe %>%
  summarize(Mean=mean(gonad_mass_5), Median=median(gonad_mass_5),  Variance=var(gonad_mass_5),
            Standard.Deviation=sd(gonad_mass_5), Standard.Error=se(gonad_mass_5), Coefficient.Variation=
            Kurtosis=kurtosis(gonad_mass_5))
print(summary.statistics)
```

```
## # A tibble: 1 x 8
##    Mean Median Variance Standard.Deviation Standard.Error Coefficient.Variation
##   <dbl>  <dbl>    <dbl>              <dbl>          <dbl>                 <dbl>
## 1  13.2   11.4     57.9               7.61          0.311                 0.575
## # ... with 2 more variables: Skewness <dbl>, Kurtosis <dbl>
```

Median and mean: change proportionally to the transformation performed (increase by 5)

Variance and standard deviation: there is no difference when a constant is added to each value since the dispersion away from the central value (mean) does not change.

Coefficient of variation (CV): changes when a constant is added to the data set since this is a ratio between s and the mean. When a constant is added, the mean always changes but s does not, changing the ratio between the two.

Kurtosis and skewness: Remain the same because these measures can only be affected by transformations that affect the shape of the distribution, such as log, arcsin, or square-root. Adding or multiplying constants to a data set does not change the shape of the distribution.

(3c) (3 pts) What would be the effect of adding 5.0 and then multiplying by 10.0?

```
#To add 5 and then multiply by 10, let's use mass5 (already added 5) and multiply by 10
dataframe$gonad_mass_10<-dataframe$gonad_mass_5*10

#summarize variables into a data frames
summary.statistics <- dataframe %>%
  summarize(Mean=mean(gonad_mass_10), Median=median(gonad_mass_10),  Variance=var(gonad_mass_10),
            Standard.Deviation=sd(gonad_mass_10), Standard.Error=se(gonad_mass_10), Coefficient.Variati
            Kurtosis=kurtosis(gonad_mass_10))
print(summary.statistics)
```

```
## # A tibble: 1 x 8
##    Mean Median Variance Standard.Deviation Standard.Error Coefficient.Variation
##   <dbl>  <dbl>    <dbl>              <dbl>          <dbl>                 <dbl>
## 1  132.   114.    5794.               76.1           3.11                 0.575
## # ... with 2 more variables: Skewness <dbl>, Kurtosis <dbl>
```

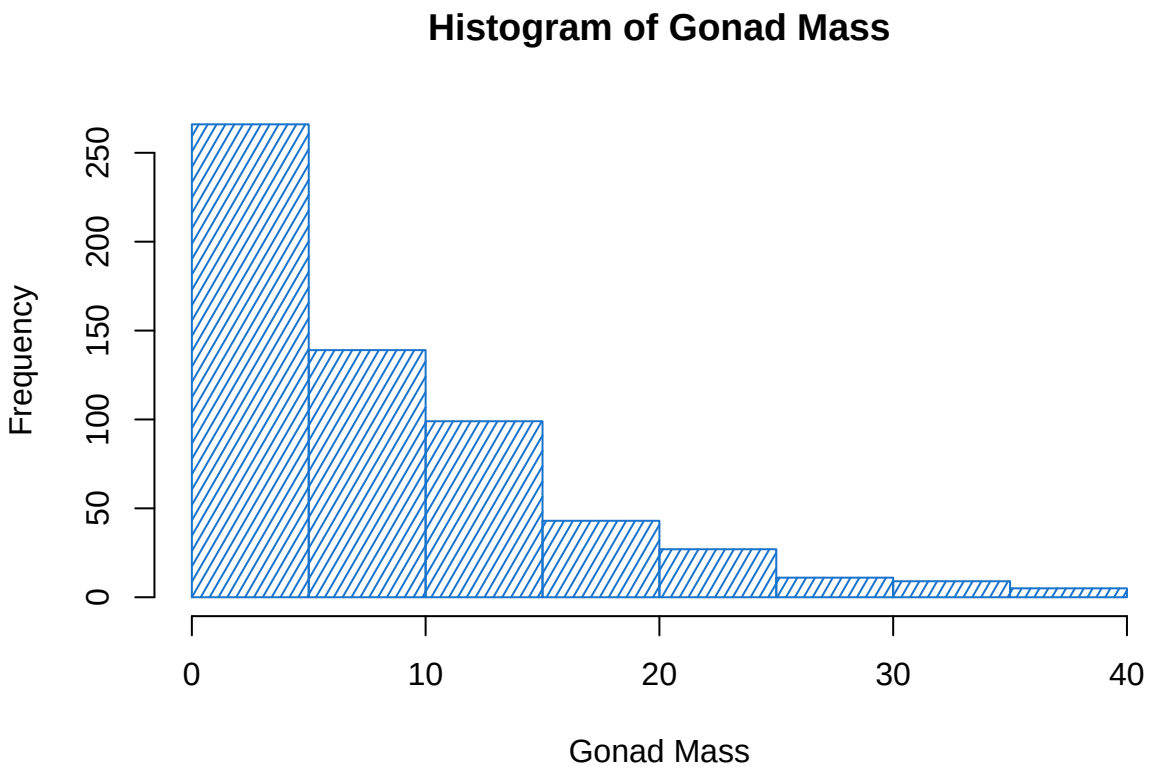
Median and mean: change proportionally to the transformation performed Variance and standard deviation: When the data set is multiplied by a constant, this changes the dispersion away from the central value, therefore changing the value of s2 and s.

Coefficient of variation (CV): When constants are multiplied to the data set, s changes proportionately to the mean, so the CV remains the same.

Kurtosis and skewness: Remain the same because these measures can only be affected by transformations that affect the shape of the distribution, such as log, arcsin, or square-root. Adding or multiplying constants to a data set does not change the shape of the distribution.

(3d) (3 pts) Make a histogram of all raw observations (untransformed values) in the kelp bass gonad mass data set. Do these data look relatively normal or not? Add the histogram below.

```
#creating a histogram plot using base R
hist(dataframe$gonad_mass, col="dodgerblue3", density=25, angle=60, #creating hatch pattern
     main="Histogram of Gonad Mass", xlab="Gonad Mass") #labels
```




```
skewness(dataframe)
```

```
##    gonad_mass  gonad_mass_5 gonad_mass_10
##      1.428559      1.428559      1.428559
```

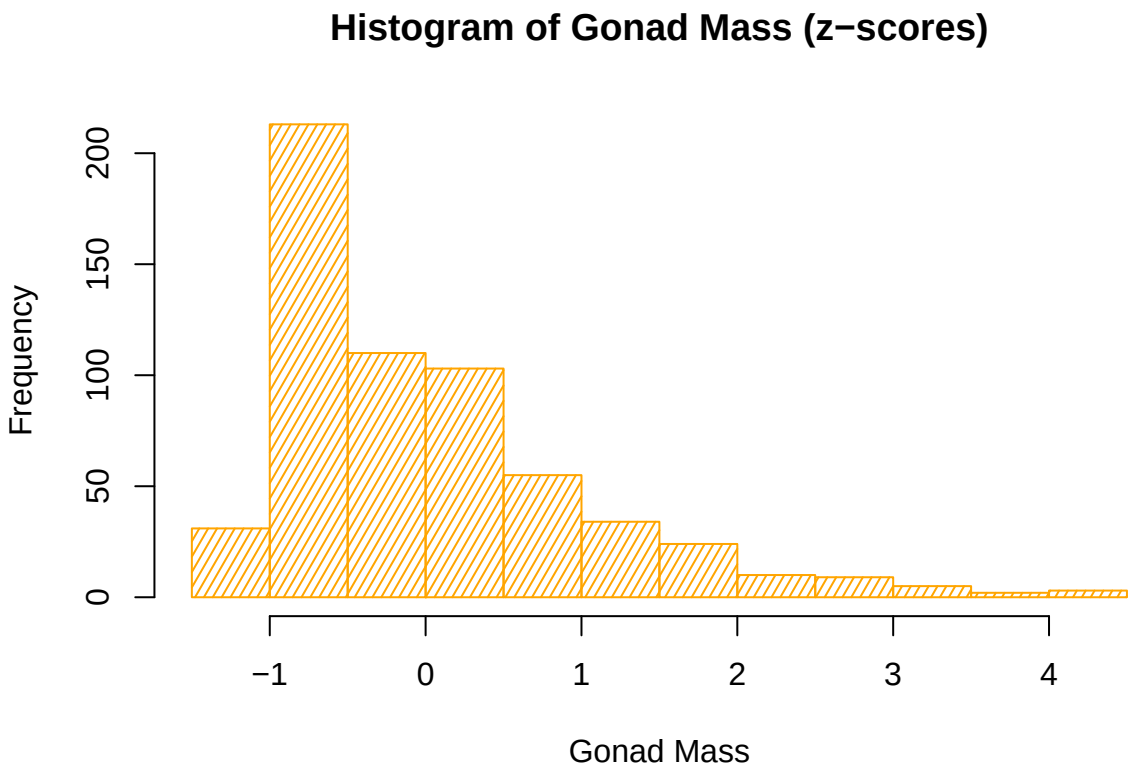
```
kurtosis(dataframe)
```

```
##    gonad_mass  gonad_mass_5 gonad_mass_10
##      5.159599      5.159599      5.159599
```

No, these data do not look normal. They are heavily skewed to the right, which is supported by our skewness metric above.

(3e) (3 pts) Convert all raw observations in the kelp bass data set into Z-scores. Make a histogram of this new data set. How does this histogram differ from the one for the raw observations? Add the new histogram below.

```
#converting data to z-scores
zscore.dataframe<-scale(dataframe$gonad_mass, center=TRUE, scale=TRUE) #Center centers the data on the

#plotting histogram using base R
hist(zscore.dataframe, col="orange", density=25, angle=60, #creating hatch pattern
     main="Histogram of Gonad Mass (z-scores)", xlab="Gonad Mass") #labels
```




```
#use skewness and kurtosis to compare how the histograms differ
skewness(zscore.dataframe)
```

```
## [1] 1.428559
```

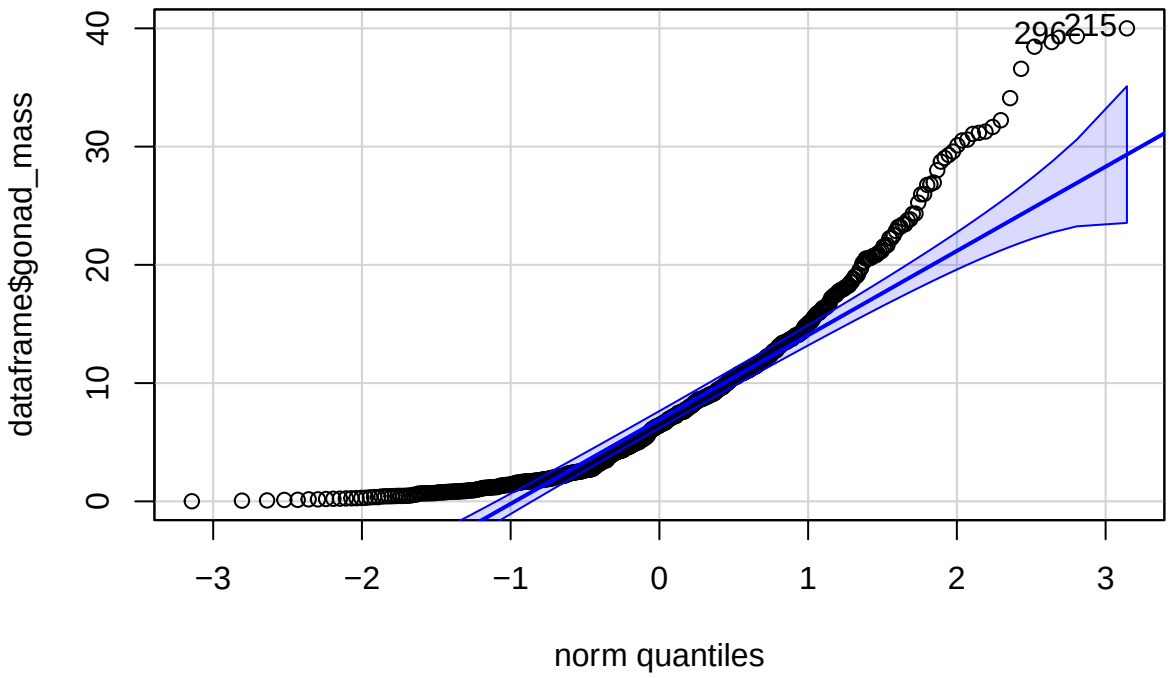
```
kurtosis(zscore.dataframe)
```

```
## [1] 5.159599
```

This still looks VERY skewed to the right. Skewness value is about the same.

(3f) (3 pts) Use the original kelp bass gonad data to create a Normal Probability Plot. Do the data appear to be normally distributed? Add the plot below.

```
#probability plot
qqp(dataframe$gonad_mass, "norm")
```



No, the data does not appear to be distributed normally as there are a lot of data points outside of the confidence limits for a normal distribution.

(4) (5 pts) Round the following numbers to three significant figures and state their implied limits before and after rounding.

```
#clear the environment
rm(list=ls())

options(digits=10) #set R options to 10 digits

#implied limits
implied.limit.1 <- c(106.5-0.05, 106.5+0.05)

implied.limit.2 <- c(0.068191-0.0000005, 0.068191+0.0000005)

implied.limit.3 <- c(3.049-0.0005, 3.049+0.0005)

implied.limit.4 <- c((2.03456-0.000005)*10^6, (2.03456+0.000005)*10^6)

implied.limit.5 <- c(2.914-0.0005, 2.914+0.0005)

implied.limit.6 <- c(20.15000-0.000005, 20.15000+0.000005)

print(c(implied.limit.1, implied.limit.2, implied.limit.3, implied.limit.4, implied.limit.5, implied.li
```

```
##  [1] 1.0645000e+02 1.0655000e+02 6.8190500e-02 6.8191500e-02 3.0485000e+00
##  [6] 3.0495000e+00 2.0345550e+06 2.0345650e+06 2.9135000e+00 2.9145000e+00
## [11] 2.0149995e+01 2.0150005e+01
```

```
#Rounding values to 3 significant digits #to round up "ceiling" to round down "floor"
plyr::round_any(106.5, 1, f = ceiling)  # [1] 107 3 sigfigs
```

```
## [1] 107
```

```
plyr::round_any(3.049, .01, f = ceiling) # [1] 3.05 3 sigfigs
```

```
## [1] 3.05
```

```
plyr::round_any(0.068191, .0001, f = ceiling) # [1] 0.0682 3 sigfigs
```

```
## [1] 0.0682
```

```
format(signif(2.03456*10^6, digits=3), scientific=TRUE) # [1] "2.03e+06" 3 sig figs
```

```
## [1] "2.03e+06"
```

```
plyr::round_any(2.914, .01, f = floor) # [1] 2.91 3 sigfigs
```

```
## [1] 2.91
```

```
plyr::round_any(20.15000, .1, f = ceiling) #[1] 20.2 3 sigfigs
```

```
## [1] 20.2
```

```
#Implied limits after rounding
implied.limit.1 <- c(107-0.5, 107+0.5)

implied.limit.2 <- c(0.0682-0.00005, 0.0682+0.00005)

implied.limit.3 <- c(3.05-0.005, 3.05+0.005)

implied.limit.4 <- c((2.03-0.005)*10^6, (2.03+0.005)*10^6)

implied.limit.5 <- c(2.91-0.005, 2.91+0.005)

implied.limit.6 <- c(20.2-0.05, 20.2+0.05)


print(c(implied.limit.1, implied.limit.2, implied.limit.3, implied.limit.4, implied.limit.5, implied.li
```

```
##  [1] 1.065e+02 1.075e+02 6.815e-02 6.825e-02 3.045e+00 3.055e+00 2.025e+06
##  [8] 2.035e+06 2.905e+00 2.915e+00 2.015e+01 2.025e+01
```

(5) (5 pts) For each of the following questions, define the sampling unit and the statistical population.

(a) What proportion of blue whales in the Pacific Ocean are reproductively mature?

```
**statistical population:** all blue whales in the Pacific
**sampling unit:** a whale
```

(b) How many mitochondria per cell?

```
**statistical population:** all cells
**sampling unit:** a cell
```

(c) How many seeds per white flowered plant?

```
**statistical population:** all white flowered plants
**sampling unit:** a white flower
```

(d) How many bacteria per 1mL in a sewage treatment plant?

```
**statistical population:** all water in the treatment plant
**sampling unit:** 1mL sample
```

(e) How much time do bees spend each time they visit a flower?

```
**statistical population:** all visits by bees
**sampling unit:** a bee visit
```

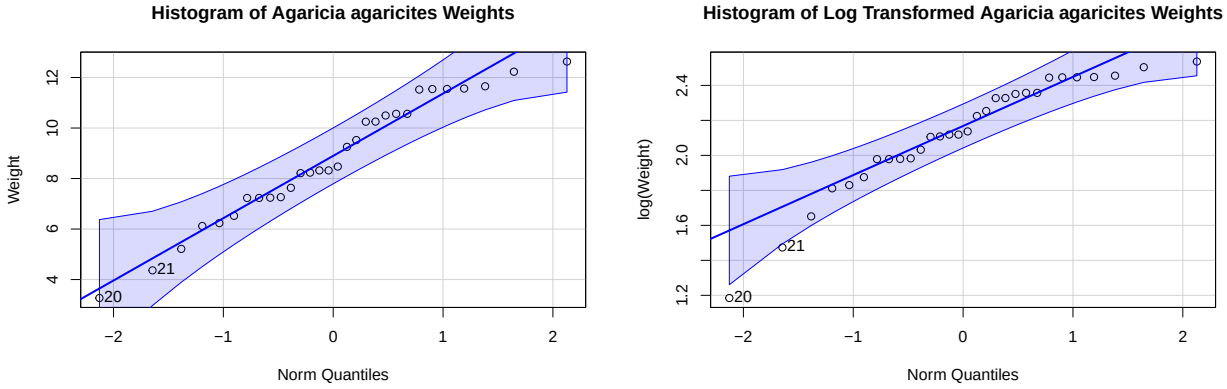
(f) How many bees visit in a 5-minute observation period?

```
**statistical population:** all 5 min periods
**sampling unit:** a 5 min period
```

(6a) (5 pts) Carla (former MS student in Peter Edmunds' lab) sampled the weights (in grams) of 30 individuals of the coral, Agaricia agaricites. The data are available in the file "Agaricia.csv". Are the data normally distributed? Does log-transformation improve the normality or not? Support your answer with whatever graph(s) you think are appropriate.

```
#clear the environment
rm(list=ls())

#read in data
dataframe<- read_csv(here("Data", "Agaricia.csv"))
#glimpse(dataframe)

#to visualize normalcy we can use one of the three following commands
#boxplot: boxplot(dataframe$weight, main="weight")
#histogram: hist(mydata$weight)
#probability plot: qqp(dataframe$weight, "norm")
```

```r
#creating a probability plot
qqp(dataframe$weight, "norm", main="Histogram of Agaricia agaricites Weights", xlab ="Norm Quantiles", y

#boxplot(dataframe$logweight, main="logweight")
#hist(dataframe$logweight)

#log transforming data
dataframe$logweight<-log(dataframe$weight)

#creating a probability plot of log transformed data
qqp(dataframe$logweight, "norm", main="Histogram of Log Transformed Agaricia agaricites Weights", xlab =
```



There are multiple ways to look at normality including a boxplot, histogram, and probability plot. Using a probability plot, one can see that all of the data points stay within the confidence interval; therefore, the data looks normal.

There are multiple kinds of transformations that we can use on our data set including: log = good for right skewed data or may be used to improve or linearity (for regressions) square roots = good for counts arcsin = good for ratios and percentages

Since this variable is a continuous variable, we would use a log transformation. The log transformation looks like a less normal distribution compared to the untransformed data, therefore we would use the orginal untransformed data.

#(6b) (4 pts) Use the Agaricia data set to estimate the mean ± 95% CI of the untransformed data sample by resampling the data with bootstrapping (just use 1000 resamplings). Plot the frequency distribution of estimates for the mean and indicate the 95% confidence intervals on the plot.

```r
#preview sample mean
mean.weight <- mean(dataframe$weight)
#glimpse(mean.weight)

#bootstrapping means
bootstrapped.mean<-replicate(1000, {
  samples<-sample(dataframe$weight,replace=TRUE);
  mean(samples)  }) #take the mean of the subsample
#this output provides 1000 different estimates of the mean, based on 1000 random samples
sortedboots<-sort(bootstrapped.mean) #sorting means

mean <- mean(bootstrapped.mean)
print(mean)
```

```
## [1] 8.80821178
```

```
#constructing the 95% confidence intervals using (25th and 975th place)
lowCI<-sortedboots[25]
highCI<-sortedboots[975]
upperCI<-highCI - mean(bootstrapped.mean)
lowerCI<-mean(bootstrapped.mean) - lowCI
confidence.interval <-c(lowerCI, upperCI)
print(confidence.interval)
```

```
## [1] 0.8565784467 0.8707915533
```

```
#histogram of bootstrapped means
hist(sortedboots, col="lightcyan", main="Bootstrapped Histogram of Agaricia agaricites Weights", xlab="
abline(v=lowCI, col="darkcyan") #adding vertical lines for the low and high CIs
abline(v=highCI, col="darkcyan")
```