Keegan Moynahan *** Worked on with Steph

Lab 7

Q1) n=123

```
require(palmerpenguins)
dat_pen = subset(penguins, species != "Adelie")
dat_pen2 = subset(dat_pen, species != "Chinstrap")
# Choose significance level
alpha = 0.05

# 2: Calculate sample standard error:
n = sum(!is.na(dat_pen2$bill_length_mm))
print(n)
```

Q2) sd = 0.0898

```
Gent_sd = sd(!is.na(dat_pen2$bill_length_mm))
print(Gent_sd)
```

Q3) t value = 1.9796

```
t_crit = abs(qt(alpha / 2, df = n - 1))
print(t_crit)
```

Q4) sse = 0.2779

```
sse = sd(dat_pen2$bill_length_mm, na.rm = TRUE) / sqrt(n)
print(sse)
```

Q5) lower 0.4418 upper 1.5420

```
# Choose significance level
alpha = 0.05

# 2: Calculate sample standard error:
n = sum(!is.na(dat_pen2$bill_length_mm))
sse = sd(dat_pen2$bill_length_mm, na.rm = TRUE) / sqrt(n)
```

```
# 3: Calculate critical t-values:
t_crit = abs(qt(alpha / 2, df = n - 1))

# 4: Calculate the CI radius:
ci_radius = sse * t_crit

# The CI is the sample mean +/- the radius:
anst_ci = c(
  lower = mean(!is.na(dat_pen2$bill_length_mm)) - ci_radius,
  upper = mean(!is.na(dat_pen2$bill_length_mm)) + ci_radius)

print(round(anst_ci, 4))
```

Q6) lower = 46.96917 upper = 48.06098

Q7)

```
require(boot)

boot_mean = function(x, i)
{
  return(mean(x[i], na.rm = TRUE))
}

myboot =
  boot(
    data = dat_pen2$bill_length_mm,
    statistic = boot_mean,
    R = 10000)
print(myboot)
```

Q8)

```
str(myboot)
```

```
gent_mean = mean(!is.na(dat_pen2$bill_length_mm))
print(gent_mean)
myboot$t0
mean(myboot$t) - myboot$t0
sd(myboot$t)
quantile(
  myboot$t,
  c(0.025, 0.975))
Q9)
# This clears the current R session's environment
rm(list = ls())

# Re-read my data:
moths = read.csv(here("data", "moths.csv"))
moth_dat = moths[,-1]

rarefaction_sampler = function(input_dat, n_iterations)
{
  n_input_rows = nrow(moth_dat)

  results_out = matrix(
    nrow = n_iterations,
    ncol = n_input_rows)
  n = nrow(moth_dat) #number of rows or sample observations
  m = 10000 #number of bootstrap iterations

  moth_result = matrix(
    nrow = m,
    ncol = n)
```

```
 # The outer loop: runs once for each bootstrap iteration.  index variable is i
 for(i in 1:n_iterations)
 {
  # The inner loop: simulates increasing sampling intensity
  # Sampling intensity ranges from 1 site to the complete count of
  # sites in the input data (n)
  for(j in 1:n)
  {
   # sample the input data row indices, with replacement
   rows_j = sample(n, size = j, replace=TRUE)

   # Creates a new data matrix
   t1 = input_dat[rows_j, ]

   # Calculates the column sums
   t2 = apply(t1, 2, sum)

   # Counts the number of columns in which any moths were observed
   results_out[i, j] = sum(t2 > 0)
  }
 }
 return(results_out)
}

rarefact = rarefaction_sampler(moth_dat, 10000)
head(rarefact)
```

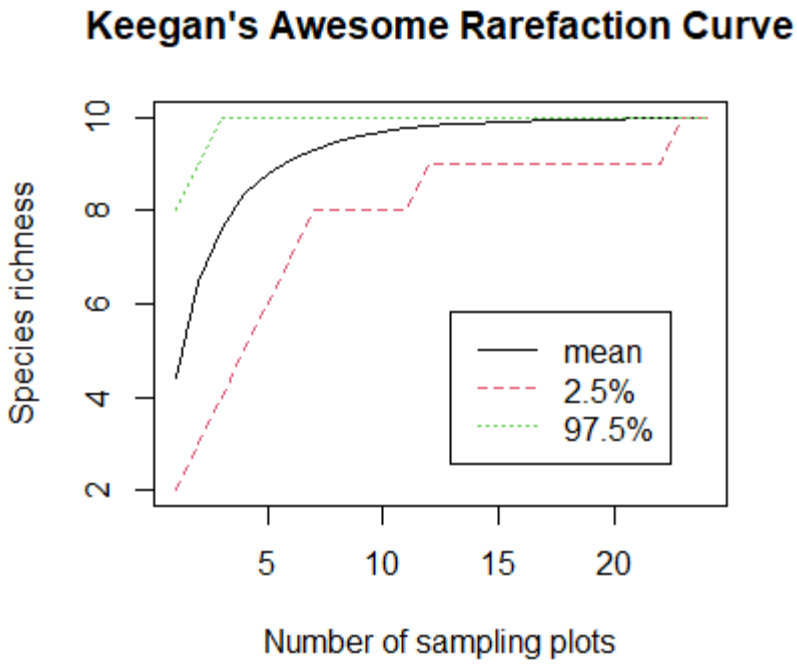
Q10) The most difficult part of building the function was trying to figure out where to put the assigned values that made the code run. At first I thought we needed to re-build the top of the function but after

looking at what was already there I realized all I needed to do was just assign the values before they were being used in the function. After that it was obvious why they were needed.

Q11)

```
moths = read.csv(here("data", "moths.csv"))

rarefact = rarefaction_sampler(moths[,-1], 10000)


rare_mean = apply(rarefact, 2, mean)

rare_quant = apply(rarefact, 2, quantile, probs=c(0.025, 0.975))

rare = t(rbind(rare_mean, rare_quant))


matplot(
  rare,
  type='l',
  xlab='Number of sampling plots',
  ylab='Species richness',
  main="Mike's Awesome Rarefaction Curve")
```

Q12)



Q13) I would say 20 sites because the mean and majority of the data end at the20. You would also see most of the species between 0 and 10.