Week 2 Recitation Notes

### 1. Notebook saving

1. Create notebook and save somewhere.
2. Move it and click "no"
3. Show that it creates a new file, but keeps the old one open
4. Move it and show that you need to click "yes"
5. Show them on trunk that you can download the homework, and use that file.

### 2. Warmup (lecture slides)

bee<-read.csv("BeeNectar.csv")  
nrow(bee)

## [1] 525

summary(bee$Nectar)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00027 0.01943 0.03249 0.03475 0.04740 0.13928

mean(bee$Nectar)

## [1] 0.03475465

sd(bee$Nectar)

## [1] 0.02086053

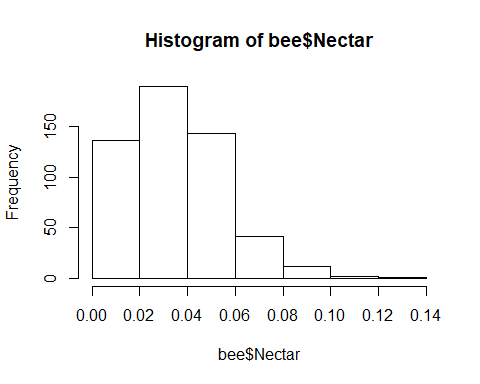
var(bee$Nectar)

## [1] 0.0004351615

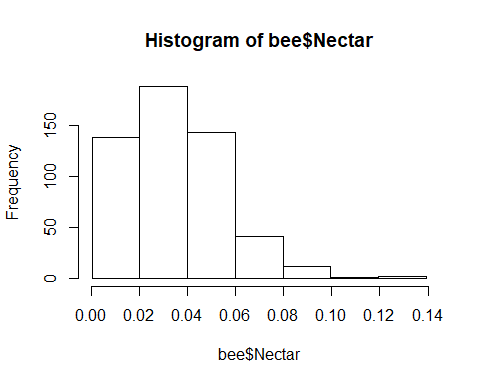
sqrt(var(bee$Nectar))

## [1] 0.02086053

hist(bee$Nectar, breaks=8)



hist(bee$Nectar, breaks=seq(min(bee$Nectar),max(bee$Nectar), length.out=8))



* **Histogram binning**: Rules of Thumb (**always** use this guideline in this class!)

|  |  |
| --- | --- |
| Sample Size | Number of Bins |
| Less than 16 | Not Enough Data |
| 16--31 | 5 |
| 32--63 | 6 |
| 64--127 | 7 |
| 128--255 | 8 |

### 3. Guided learning

#### 3a.Introduce seq and sample

num<-seq(1, 10, 1) # sequence of numbers, ranging from = to = and by =  
num<-seq(1, 10, length.out=5)  
  
seq(min(bee$Nectar),  
 max(bee$Nectar),   
 length.out=8) # select for number of "breaks" or bins

## [1] 0.00027000 0.02012857 0.03998714 0.05984571 0.07970429 0.09956286  
## [7] 0.11942143 0.13928000

sample(num, 2)

## [1] 10.00 3.25

#### 3b. Introduce repeating function

Not equal numbers. Hat full of marbles, and you know the number of each. Randomly sample from the hat!

y<-rep("yellow", 20)  
r<-rep("red", 5)  
b<-rep("blue", 100)  
  
color<-c(y, r, b)  
  
sample(color, 10)

## [1] "yellow" "blue" "blue" "blue" "blue" "blue" "blue"   
## [8] "blue" "red" "blue"

rep(5, 22)

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

rep(c(5, 10), 11)

## [1] 5 10 5 10 5 10 5 10 5 10 5 10 5 10 5 10 5 10 5 10 5 10

rep(c(5, 10, 33), each = 3)

## [1] 5 5 5 10 10 10 33 33 33

rep(c("A", "B", "C"), 3)

## [1] "A" "B" "C" "A" "B" "C" "A" "B" "C"

rep(c("A", "B", "C"), each = 3)

## [1] "A" "A" "A" "B" "B" "B" "C" "C" "C"

rep(c("A", "B", "C"), length.out=11)

## [1] "A" "B" "C" "A" "B" "C" "A" "B" "C" "A" "B"

### 4. EXERCISE 1: sampling from Nectar column.

ten<-sample(bee$Nectar, 10)  
fo<-sample(bee$Nectar, 40)  
  
mean(ten)

## [1] 0.033313

mean(fo)

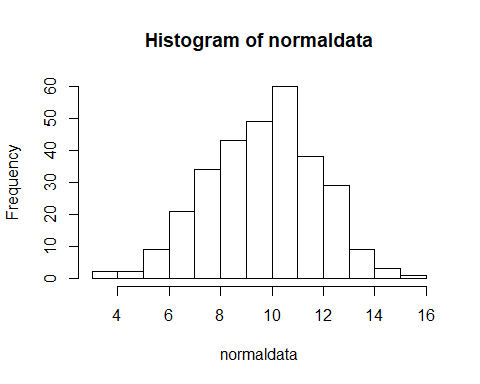
## [1] 0.038229

mean(bee$Nectar)

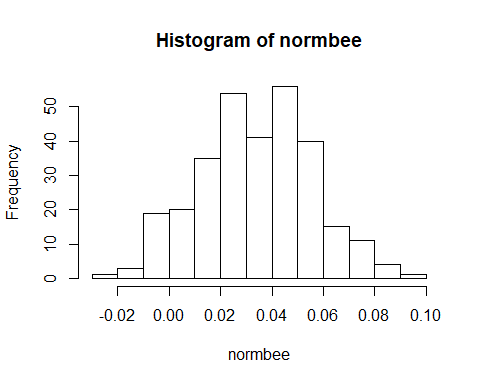
## [1] 0.03475465

### 5. Introduce rnorm() that randomly generates numbers given some specified mean and standard deviation

normaldata<-rnorm(300, mean=10, sd=2) # run this a few times, what do you notice?  
  
hist(normaldata)



# DO YOU THINK YOUR BEE DATA IS NORMALLY DISTRIBUTED??  
normbee<-rnorm(300, mean=mean(bee$Nectar), sd=sd(bee$Nectar)) # Enter in your mean and standard from the nectar weight  
  
hist(normbee)



### 6. EXERCISE 2. Create a boxplot of the nectar weight of bees from different colonies

class(bee$Colony)

## [1] "integer"

bee$Colony<-as.factor(bee$Colony)  
class(bee$Colony)

## [1] "factor"

# A. Summary stats  
  
summary(bee$Pollen)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000000 0.000000 0.000000 0.001994 0.000000 0.038210

summary(bee$ITS)

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 2.80 3.58 3.82 3.84 4.07 5.32 1

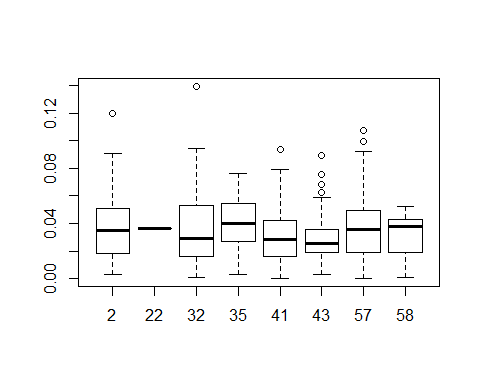
sd(bee$Pollen)

## [1] 0.005320508

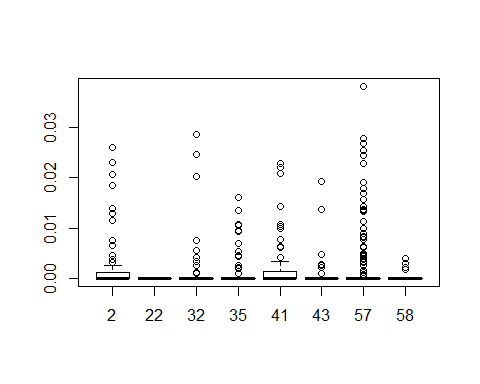
sd(bee$ITS, na.rm=T)

## [1] 0.4045449

# B. Boxplots  
  
with(bee, boxplot(Nectar ~ Colony))



with(bee, boxplot(Pollen ~ Colony))

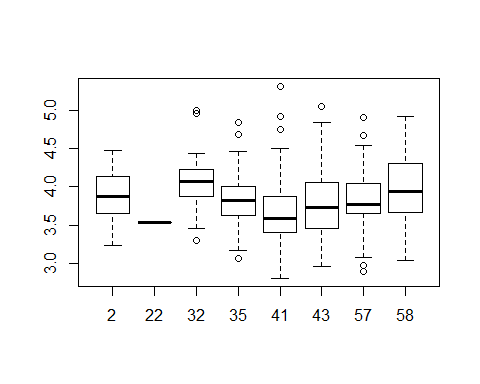


with(bee, boxplot(ITS ~ Colony))  
  
# C. Advanced problem  
  
library(tidyverse)

## Loading tidyverse: ggplot2  
## Loading tidyverse: tibble  
## Loading tidyverse: tidyr  
## Loading tidyverse: readr  
## Loading tidyverse: purrr  
## Loading tidyverse: dplyr

## Conflicts with tidy packages ----------------------------------------------

## filter(): dplyr, stats  
## lag(): dplyr, stats



new.data<-bee %>% group\_by(Colony) %>% summarise(mu.nec = round(mean(Nectar), 6),  
 sd.nec = round(sd(Nectar), 6),  
 mu.poll = round(mean(Pollen), 6),  
 sd.poll = round(sd(Pollen), 6),   
 range.nec = round(max(Nectar) - min(Nectar), 6))