STAC33 TUT 4

Uzair Mirza

07/02/2022

Introduction

We will be discussing problems from CH:5 and CH:8 from PASIS during this week's TUT.
CH:5 focuses on data processing while CH:8 focuses on the relation of Power with Sample size.

Example 8.7 Calculating power and sample size for estimating mean

load tidyverse

Example 5.6 Dolphins

Polution level `mercury`, `age`, `sex`

a. Read and display some of the data

b. Display the `mercury` and `sex`

- `select()` -> to choose the cols

```
#a
my_url <- "http://ritsokiguess.site/datafiles/dolphins.csv"
dolphins <- read_csv(my_url)
```

```
## Rows: 45 Columns: 3
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): sex
## dbl (2): mercury, age
```

```
##
## 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.
```

```
head(dolphins)
```

```
## # A tibble: 6 x 3
##   mercury   age sex
##     <dbl> <dbl> <chr>
## 1    1.7   0.21 male
## 2    1.72  0.33 male
## 3    8.8   2    male
## 4    5.9   2.2  male
```

```
## 5  101     8.5  male
## 6   85.4  11.5  male
```

```
#b
dolphins %>% select(mercury, sex)
```

```
## # A tibble: 45 x 2
##    mercury sex
##      <dbl> <chr>
##  1    1.7  male
##  2    1.72 male
##  3    8.8  male
##  4    5.9  male
##  5  101    male
##  6   85.4  male
##  7  118    male
##  8  183    male
##  9  168    male
## 10  218    male
## # ... with 35 more rows
```

c. Display the col with Name having 3 chars
-Regex: here **NOT SO IMP** bc doable with online help

```
dolphins %>%
select(matches("^...$"))
```

```
## # A tibble: 45 x 2
##      age sex
##    <dbl> <chr>
##  1  0.21 male
##  2  0.33 male
##  3  2    male
##  4  2.2  male
##  5  8.5  male
##  6 11.5  male
##  7 11.5  male
##  8 13.5  male
##  9 16.5  male
## 10 16.5  male
## # ... with 35 more rows
```

d. Display only the mercury levels for the females.

- `filter()` -> var + condition
- Order matters

```
dolphins %>% filter(sex == "female") %>%
select(mercury)
```

```
## # A tibble: 17 x 1
##    mercury
##      <dbl>
##  1    2.5
##  2    9.35
```

```
##  3    4.01
##  4   29.8
##  5   45.3
##  6  101
##  7  135
##  8  142
##  9  180
## 10  174
## 11  247
## 12  223
## 13  167
## 14  157
## 15  177
## 16  475
## 17  342
```

e. What is the mean mercury concentration for all the dolphins whose age is less than 15?

- Order: address the condition then calculate the statistic

```
dolphins %>% filter(age<15) %>%
summarize(m = mean(mercury))
```

```
## # A tibble: 1 x 1
##       m
##   <dbl>
## 1  55.5
```

e. What is the mean mercury concentration for all the dolphins whose age is greater than 25?

```
dolphins %>%
group_by(age>25) %>%
summarize(m = mean(mercury))
```

```
## # A tibble: 2 x 2
##   `age > 25`     m
##   <lgl>      <dbl>
## 1 FALSE       142.
## 2 TRUE        309.
```

The Data

```
# load the data set
df <- iris
# add some 'NA' in the data set
df <- df %>% add_row(Sepal.Length = 5, Sepal.Width = 5,
                Petal.Length = NA, Petal.Width = 0.2,
                Species = "setosa")

# add an outlier `.length = 25`
df <- df %>% add_row(Sepal.Length = 21, Sepal.Width = 5,
                Petal.Length = 3, Petal.Width = 0.2,
                Species = "setosa")
```

Goal of the study

Sepal length to petal length ratio for the collected flowers

Get the interested Variables(columns) This is the case of our study are just the 3 variables;
- Sepal.Length
- Petal.Length
- Species

Will use `select()`, `%>%`

```
df.1 <- df %>%
  select(Sepal.Length, Petal.Length, Species)
```

-What is %>%, select()?

Check if there are any `NA` What is `NA`?

Will use `any() is.na()`

```
any(is.na(df.1))
```

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

-What is.na() does? In -> Out

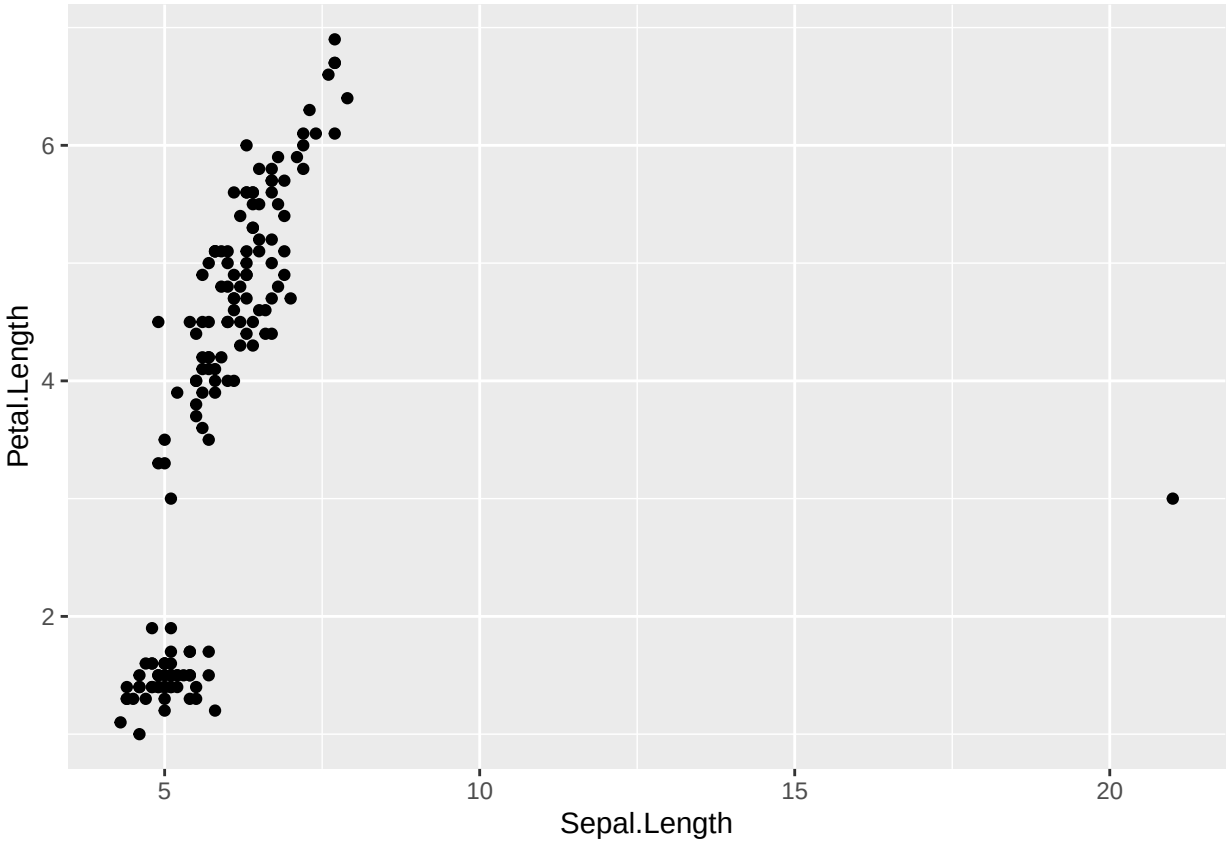
Do we remove the `NA`? -Data is expensive to collect
-Variable `NA` not so important
-Size of dataset ~ Weights/value

Remove the `NA` value, in our case.

```
df2 <- na.omit(df.1)
```

```
ggplot(df2, aes(x= Sepal.Length, y= Petal.Length))  + geom_point()
```

Check for outliers + Remove them



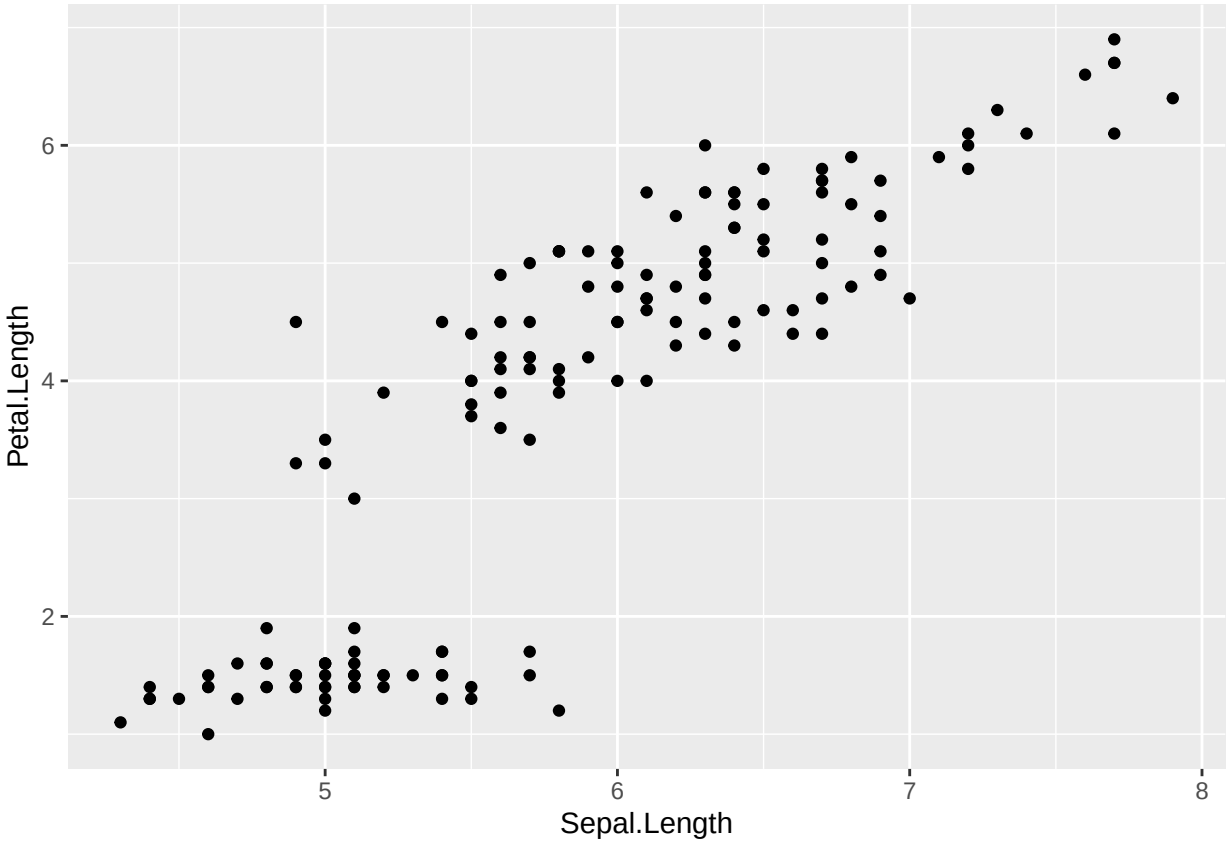
Hmmm that one value!!!

Options:
- Manually go over and find this value
- **Filter** this observation out

```
df3 <- df2 %>%
  filter(Sepal.Length < 10)
```

-`filter()` In -> Out
- Multiple logical statements & |

```
ggplot(df3, aes(x= Sepal.Length, y= Petal.Length))  + geom_point()
```



Get the ratios This will require making of a new variable `Ratio`, using `mutate()`

```
df3 <- df3 %>%
  mutate(Ratio = Sepal.Length/Petal.Length )
```

-What `mutate()` is

How about arrange the ratios in acceding order

```
df3 <- df3 %>% arrange(desc(Ratio))
```

Get the average ratio for each type of flower Will require group_by(), summarise()

```
df3 %>%
  group_by(Species) %>%
  summarise(n = n(),
            AvgRatio = sum(Ratio)/n)
```

```
## # A tibble: 3 x 3
##   Species        n AvgRatio
##   <chr>      <int>    <dbl>
## 1 setosa        50     3.46
## 2 versicolor    50     1.40
## 3 virginica     50     1.19
```

What is simulation?

When you have facts(estimate about means, SD, distribution) about the population and would like to do some studies. So instead of sampling from the population you work with these facts to construct samples to study.
Why?
Because cheaper
Easy to study and generate
Reliable (bc of LLN)

What is Power?

- Prob of correctly rej the H_0
- Not making Type 2 error
- That is the resulting p-value $< \alpha$
- Power relation with sample-size of simulation
  LLN

Example 8.7 Calculating power and sample size for estimating mean

Goal: **ESTIMATE** a population mean
Facts: $\sigma = 20$
Aprox Normally Distributed
We will be testing the null hypothesis that the population mean is 100. Suppose the population mean is actually 110, and we want to determine how likely we are to (correctly) reject the null hypothesis in this case, using a two-sided (**but one-sample**) test with $\alpha = 0.05$.

a. We will take a sample of size n = 30. **Calculate** the power of this test
   $\delta = \mu - \mu_0$ (110 - 100 = 10)

```
power.t.test(n=30,delta=10,sd=20, type="one.sample", alternative="two.sided")
```

```
##
##      One-sample t test power calculation
##
##               n = 30
##           delta = 10
##              sd = 20
##       sig.level = 0.05
##           power = 0.7539627
##     alternative = two.sided
```

If instead Professor Ken has mentioned **ESTIMATE** then you use the simulation method.
Run your simulation `sim = 1:1000`. Get the sample `rnorm(30, 110, 20)` -> Calculate the test-stat `t.test(samples, mu = 100)` + p-value for each simulation.
Count the ratio of p-value $< \alpha$ OVER THE simulation size
This is your estimated power.

```r
set.seed(420)

tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(samples = list(rnorm(30, 110, 20))) %>%
  mutate(ttest = list(t.test(samples, mu = 100))) %>%
  mutate(pvals = ttest$p.value) %>%
  count(pvals<=0.05)
```

```
## # A tibble: 2 x 2
## # Rowwise:
##   `pvals <= 0.05`     n
##   <lgl>           <int>
## 1 FALSE             201
## 2 TRUE              799
```

```r
# power estimate
772/1000
```

```
## [1] 0.772
```

b. Find the sample size necessary to obtain a power of at least 0.80 under these conditions. What sample size do you need?

```r
power.t.test(delta=10,power=0.80,sd=20,type="one.sample",alternative="two.sided")
```

```
##
##      One-sample t test power calculation
##
##               n = 33.3672
##           delta = 10
##              sd = 20
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
```

ALWAYS ROUND UP `n` as it is integer and we need **AT LEAST** .8.

Similar approach BUT with simulation
Same code as above, play around with the `n` see what happens

```r
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(samples = list(rnorm(35, 110, 20))) %>%
  mutate(ttest = list(t.test(samples, mu = 100))) %>%
  mutate(pvals = ttest$p.value) %>%
  count(pvals<=0.05)
```

```
## # A tibble: 2 x 2
## # Rowwise:
##   `pvals <= 0.05`     n
##   <lgl>           <int>
## 1 FALSE             182
## 2 TRUE              818
```

```r
# make a function
sim_power = function(n){
  tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(samples = list(rnorm(n, 110, 20))) %>%
  mutate(ttest = list(t.test(samples, mu = 100))) %>%
  mutate(pvals = ttest$p.value) %>%
  count(pvals<=0.05)
}

sim_power(34)
```

```
## # A tibble: 2 x 2
## # Rowwise:
##   `pvals <= 0.05`     n
##   <lgl>           <int>
## 1 FALSE             199
## 2 TRUE              801
```