Assignment 8

Vishal

2022-11-23

1. Obstacles to valid scientific inference

1.1 (Q1)

1. Measurement distortions:
The measurement of a test is the probability that the test will reject the null hypothesis. When, in fact, a real difference is to be found.
E.g.

2. Selection bias:
Sample selection bias is a type of bias caused by choosing non-random data for statistical analysis. The bias exists due to a flaw in the sample selection process, where a subset of the data is systematically excluded due to a particular attribute.

E.g. In a study of smoking and chronic lung disease, theassociation of exposure with disease will tend to be weaker if controls are selected from a hospital population (because smoking causes many diseases resulting in hospitalization) than if controls are selected from the community.
In this example, hospital controls do not represent the prevalence of exposure (smoking) in the community from which cases of chronic lung disease arise. The exposure-disease association has been distorted by selection of hospital controls.

3. Confounding variables:
A confounding variable is a third variable that influences both the independent and dependent variables. Failing to account for confounding variables can cause you to wrongly estimate the relationship between your independent and dependent variables.

Let's say we take test of 200(100 boys and 100 girls) school students. We found that lack of excercise leads to weight gain. One problem with this experiment is that it lacks control variables. For example, the use of placebos, or random assignments to groups. So you really can't say for sure whether lack of exercise leads to weight gain. One confounding variable is how much people eat. It's also possible that boys eat more than girls; this could also make sex a confounding variable. Nothing was mentioned about starting weight, occupation or age either. A poor study design like this could lead to bias. For example, if all of the girls in the study were teen-aged, and all of the boys were aged 10, age would have a direct effect on weight gain. That makes age a confounding variable.

2. paired t-test and effect size

```
data("Barley")
detach('package:PairedData', unload = T)
detach('package:MASS', unload = T)

head(Barley)
```

```
##   Farm Glabron Velvet
## 1  F01      49     42
## 2  F02      47     47
## 3  F03      39     38
## 4  F04      37     32
## 5  F05      46     41
## 6  F06      52     41
```

2 (Q1)

```
t.test(Barley$Glabron, Barley$Velvet, conf.level = 0.01)
```

```
##
##  Welch Two Sample t-test
##
## data:  Barley$Glabron and Barley$Velvet
## t = 2.1764, df = 21.132, p-value = 0.04101
## alternative hypothesis: true difference in means is not equal to 0
## 1 percent confidence interval:
##  5.882188 5.951146
## sample estimates:
## mean of x mean of y
##  48.33333  42.41667
```

2 (Q1)

```
cohensD(Barley$Glabron, Barley$Velvet)
```

```
## [1] 0.8884976
```

2 (Q3)

There are three assumptions made in t-test. They are as follows:
1. Independence: It assumes that each observation is independent of each other.
2. Normality: The difference between the pairs is assumed to be normally distributed. **3. No extreme outliers:** There shouldn't be any extreme outliers in the differences.

3. Implementing unpaired t-test

```
peng_AC <- penguins %>%
  drop_na(species,body_mass_g) %>%
  filter(species != "Gentoo")
head(peng_AC %>% select(species, flipper_length_mm, body_mass_g), 5)
```

```
## # A tibble: 5 x 3
##   species flipper_length_mm body_mass_g
##   <fct>               <int>       <int>
## 1 Adelie                181        3750
## 2 Adelie                186        3800
## 3 Adelie                195        3250
## 4 Adelie                193        3450
## 5 Adelie                190        3650
```

```
val_col <- "body_mass_g"
group_col <- "species"
data <- peng_AC

data_new <- data %>%
  rename(group = (!!group_col), val = (!!val_col)) %>%
  group_by(group) %>%
  drop_na(val) %>%
  summarise(mn = mean(val))

data_new
```

```
## # A tibble: 2 x 2
##   group        mn
##   <fct>     <dbl>
## 1 Adelie    3701.
## 2 Chinstrap 3733.
```

```
t_test_function <- function(data, val_col, group_col, val_equal = F) {

  temp <- data %>%
    rename(group = (!!group_col), val = (!!val_col)) %>%
    group_by(group) %>%
    drop_na(val) %>%
    summarise(mn = mean(val), sd = sd(val), vr = var(val), sample_size = n())

  if (val_equal == F) {
    print("Calculating using Student's t-test")
    sd_combined <- sqrt(((temp$sample_size[1] - 1) * temp$sd[1] ^ 2 +
                         (temp$sample_size[2] - 1) * temp$sd[2] ^ 2) /
                         (temp$sample_size[1] + temp$sample_size[2] - 2))

    t_stat <- (temp$mn[1] - temp$mn[2]) / (sd_combined * sqrt(1 / temp$sample_size[1] + 1 / temp$sample_
    effect_size <- (temp$mn[1] - temp$mn[2]) / (sd_combined)
    p_val <- 2 * (1 - pt(abs(t_stat), df = temp$sample_size[1] + temp$sample_size[2] - 2))
```

```
    result_df <- data.frame(t_stat = t_stat, effect_size = effect_size, p_val = p_val)
    return(result_df)
  }

  else {
    t_stat <- (temp$mn[1] - temp$mn[2]) / (sqrt(((temp$sd[1] ^ 2) / temp$sample_size[1]) +
                                          ((temp$sd[2] ^ 2) / temp$sample_size[2])))
    effect_size <- (temp$mn[1] - temp$mn[2]) / sqrt((temp$vr[1] + temp$vr[2])/ 2)
    p_val <- 2 * (1 - pt(abs(t_stat), df = temp$sample_size[1] + temp$sample_size[2] - 2))

    result_df <- data.frame(t_stat = t_stat, effect_size = effect_size, p_val = p_val)
    return(result_df)
  }
}

t_test_function(peng_AC, val_col = "body_mass_g", group_col = "species", val_equal = T)
```

```
##       t_stat effect_size     p_val
## 1 -0.5430902 -0.07664236 0.5876251
```

4. Useful concepts in statistical hypothesis testing

4 (Q1)

1. Null hypothesis: The null hypothesis is a typical statistical theory which suggests that no statistical relationship and significance exists in a set of given single observed variable, between two sets of observed data and measured phenomena.

2. Alternative hypothesis: The alternative hypothesis is a statement used in statistical inference experiment. It is contradictory to the null hypothesis and denoted by Ha or H1. We can also say that it is simply an alternative to the null. In hypothesis testing, an alternative theory is a statement which a researcher is testing. This statement is true from the researcher's point of view and ultimately proves to reject the null to replace it with an alternative assumption. In this hypothesis, the difference between two or more variables is predicted by the researchers, such that the pattern of data observed in the test is not due to chance.

3. Test statistic: A test statistic describes how closely the distribution of your data matches the distribution predicted under the null hypothesis of the statistical test you are using.

4. Type 1 error: Type I error is a false positive conclusion. So, if the researcher incorrectly rejects a true null hypothesis it is called as "Type I error".

5. Type 2 error: Type II error is a false negative conclusion. It occurs when a researcher fails to reject a null hypothesis which is really false.

6. The size of a test: It is the maximum probability of committing a Type I error i.e. incorrectly rejecting a true null hypothesis when it is true.

7. The power of a test: It is the probability of making a correct decision if the alternative hypothesis is true i.e. it is the probability of rejecting the null hypothesis when the alternative hypothesis is true.

8. The significance level: The significance level for a given hypothesis testing is a value for which a p-value less than or equal to is considered statistically significant.

9. The p-value: It is the probability of obtaining results at least as extreme as the observed results of a statistical hypothesis test, assuiming that the null hypothesis is correct.

10. Effect size: It measures the strength of the relationship between two variables on a numeric scale. In statistics analysis, the effect size is usually measured using either standardized mean difference, odd ratio or correlation coefficient.

4 (Q2)

(1) Yes, the p-value is the probability that the null hypothesis is true. Whereas, the (1 - p-value) is the probability that the alternative hypothesis is true.

(2) If p-value is less than 0.05 then is considered to be statistically significant and the null hypothesis is rejected. Although, if the p-value is greater than 0.05 (i.e. significance level) means that deviation from the null hypothesis is not statistically significant and the null hypothesis is not rejected.

5. Investigating test size for an unpaired Student's t-test

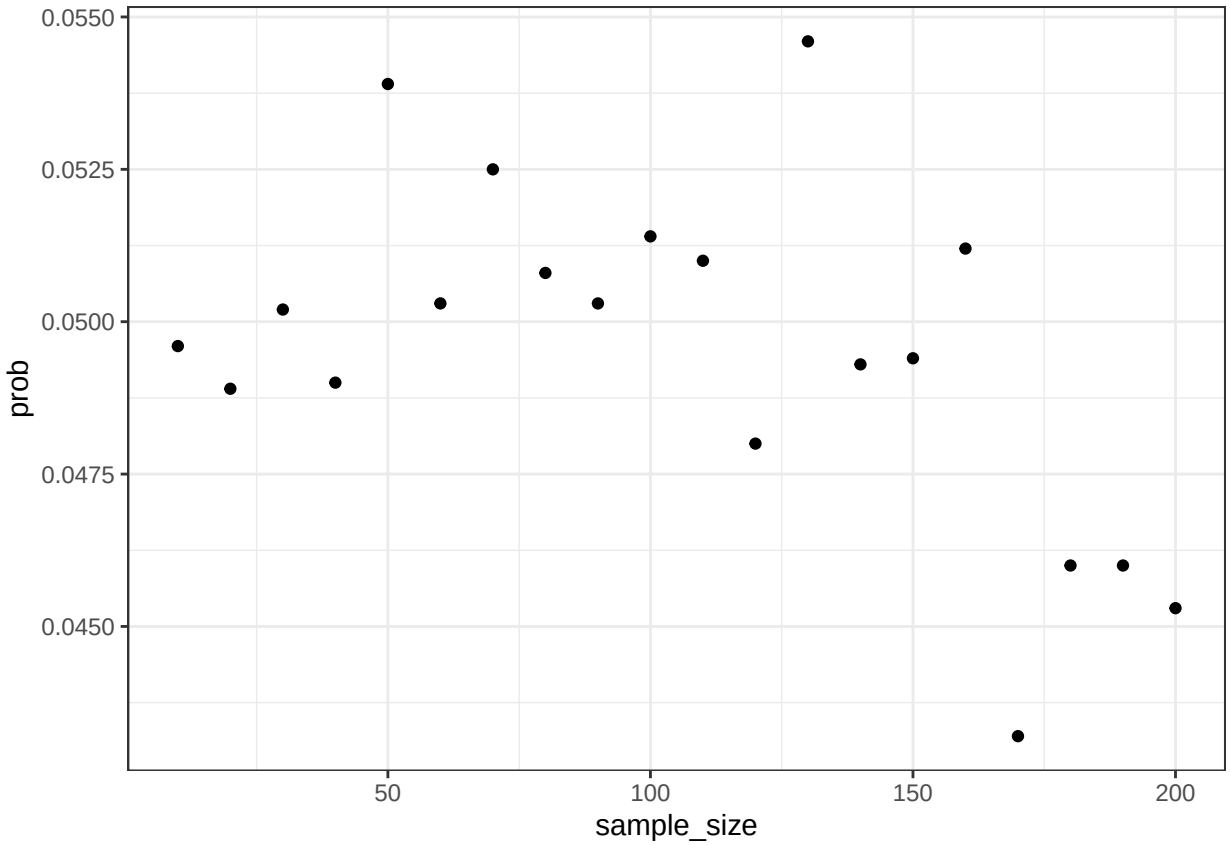
```
s_sizes <- seq(10, 200, 10)

sampleSizesVec <- c()
probVec <- c()

for (i in s_sizes) {
  num_trials <- 10000
  sample_size <- i
  mu_0 <- 1
  mu_1 <- 1
  sigma_0 <- 3
  sigma_1 <- 3
  alpha <- 0.05
  set.seed(0) # set random seed for reproducibility

  single_alpha_test_size_simulation_df <- data.frame(trial =
                                                       seq(num_trials)) %>%
    mutate(sample_0=map(.x = trial, .f = ~rnorm(n = sample_size, mean =
                                                  mu_0, sd = sigma_0)),
    sample_1=map(.x = trial, .f = ~rnorm(n = sample_size, mean = mu_1, sd =
                                           sigma_1))) %>%
    # generate p values
```

```
    mutate(p_value = pmap(.l = list(trial, sample_0, sample_1),
                          .f = ~t.test(..2, ..3, var.equal =
                                         TRUE)$p.value))%>%
    # type I error
    mutate(type_1_error = p_value < alpha)

  prob <- single_alpha_test_size_simulation_df %>%
    pull(type_1_error) %>%
    mean() # estimate of coverage probability

  sampleSizesVec <- c(sampleSizesVec, i)
  probVec <- c(probVec, prob)
}

probssDf <- data.frame(sample_size = sampleSizesVec, prob = probVec)
head(probssDf)
```

```
##   sample_size   prob
## 1          10 0.0496
## 2          20 0.0489
## 3          30 0.0502
## 4          40 0.0490
## 5          50 0.0539
## 6          60 0.0503
```

5 (Q1)

```
ggplot(probssDf, aes(x = sample_size, y = prob)) +
  geom_point() +
  theme_bw()
```



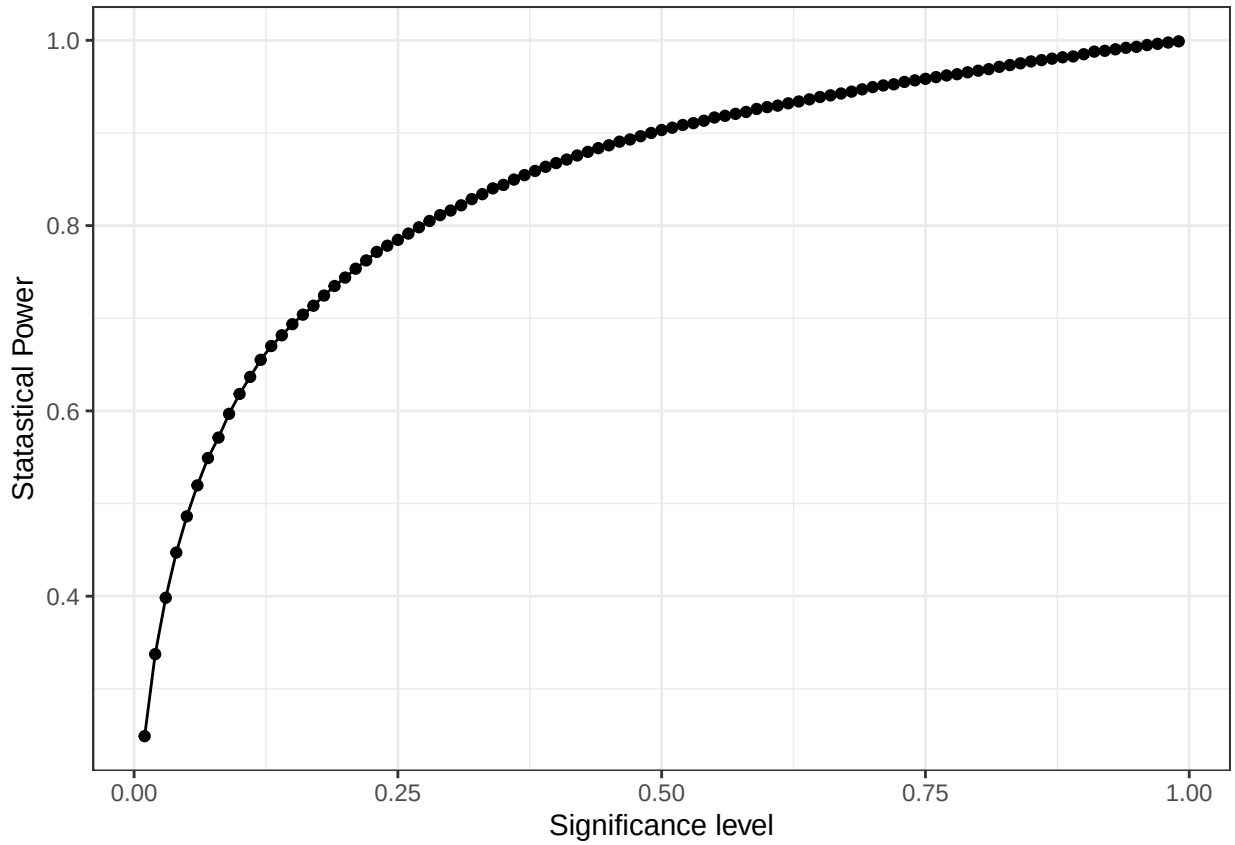
6. The statistical power of an unpaired t-test

```
alphas = seq(0.01, 0.99, 0.01)

alphaVec <- c()
probVec <- c()

for (i in alphas) {
  num_trials <- 10000
  n_0 <- 30
  n_1 <- 30
  mu_0 <- 3
  mu_1 <- 4
  sigma_0 <- 2
  sigma_1 <- 2
  alpha <- i
  set.seed(0) # set random seed for reproducibility

  multiple_alpha_stat_power_simulation_df <- data.frame(trial = seq(num_trials)) %>%
    # generate random Gaussian samples
    mutate(sample_0 = map(.x = trial, .f =~ rnorm(n = n_0, mean = mu_0,
                                                  sd = sigma_0)),
           sample_1 = map(.x = trial, .f =~ rnorm(n = n_1, mean = mu_1,
                                                  sd = sigma_1))) %>%
    # for each sample, generate p value; check examples of pmap() with ?map
    mutate(p_value = pmap(.l = list(trial, sample_0, sample_1),
                          .f =~ t.test(..2, ..3, var.equal =
```

```
                                          TRUE)$p.value)) %>%
    # estimate of coverage probability
    mutate(reject_null = p_value < alpha )

  prob <- multiple_alpha_stat_power_simulation_df %>%
    # extract a column
    pull(reject_null) %>%
    # compute probability
    mean()

  probVec <- c(probVec, prob)
  alphaVec <- c(alphaVec, i)

  probsDf2 <- data.frame(alpha = alphaVec, prob = probVec)
  head(probsDf2)
}
```

6 (Q1)

```
ggplot(probsDf2, aes(x = alpha, y = prob)) +
  geom_point() +
  geom_line() +
  theme_bw() +
  labs(y = "Statastical Power", x = "Significance level")
```



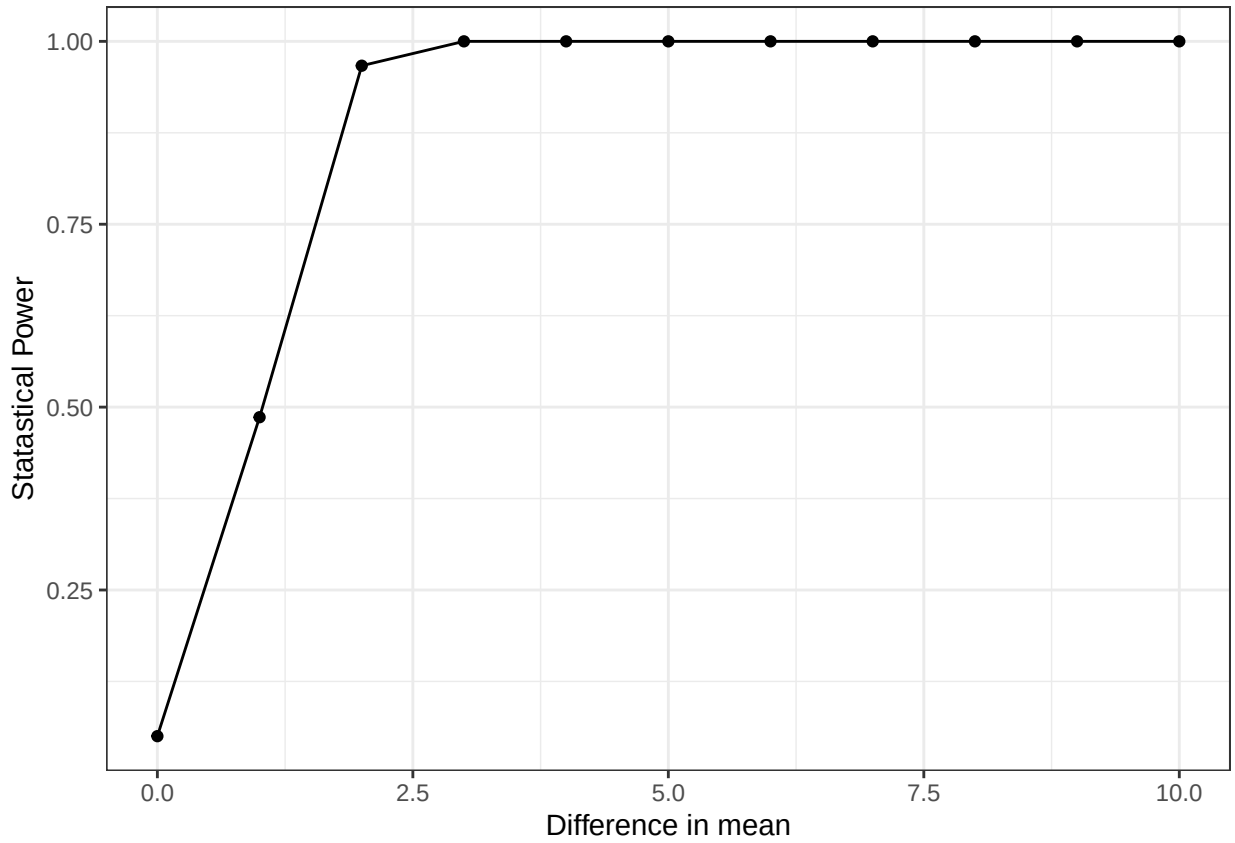
6 (Q2)

```
means <- seq(3, 13, 1)

mean_diff <- c()
probs <- c()

for (i in means) {
  num_trials <- 10000
  n_0 <- 30
  n_1 <- 30
  mu_0 <- 3
  mu_1 <- i
  sigma_0 <- 2
  sigma_1 <- 2
  alpha <- 0.05
  set.seed(0) # set random seed for reproducibility

  mean_diff_stat_power_simulation_df <- data.frame(trial =
                                                     seq(num_trials)) %>%
    # generate random Gaussian samples
    mutate(sample_0 = map(.x = trial, .f =~ rnorm(n = n_0, mean = mu_0,
                                                  sd = sigma_0)),
           sample_1 = map(.x = trial, .f =~ rnorm(n = n_1, mean = mu_1,
```

```
                                                  sd = sigma_1))) %>%
    # for each sample, generate p value; check examples of pmap() with ?map
    mutate(p_value = pmap(.l = list(trial, sample_0, sample_1),
                          .f =~ t.test(..2, ..3, var.equal =
                                         TRUE)$p.value)) %>%
    # estimate of coverage probability
    mutate(reject_null = p_value < alpha )

  prob <- mean_diff_stat_power_simulation_df %>%
    # extract a column
    pull(reject_null) %>%
    # compute probability
    mean()

  mean_diff <- c(mean_diff, abs(mu_0 - mu_1))
  probs <- c(probs, prob)

  probsDf3 <- data.frame(mean_diff = mean_diff, prob = probs)
}
head(probsDf3)
```

```
##   mean_diff   prob
## 1         0 0.0502
## 2         1 0.4862
## 3         2 0.9667
## 4         3 1.0000
## 5         4 1.0000
## 6         5 1.0000
```

```
ggplot(probsDf3, aes(x = mean_diff, y = prob)) +
  geom_line() +
  geom_point() +
  theme_bw() +
  labs(y = "Statastical Power", x = "Difference in mean")
```



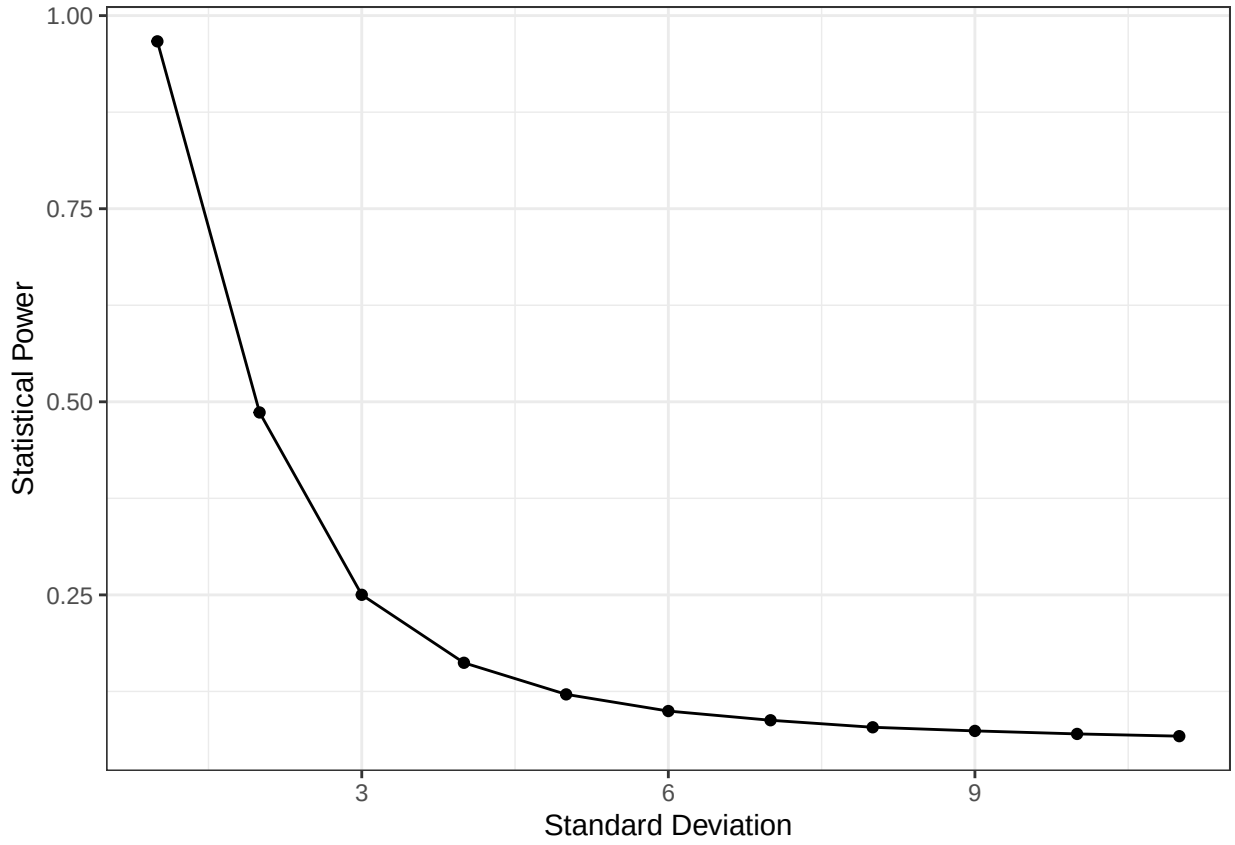
6 (Q3)

```
sigmaVals <- seq(1, 11, 1)

sigmaVec <- c()
probs <- c()

for (i in sigmaVals) {
  num_trials <- 10000
  n_0 <- 30
  n_1 <- 30
  mu_0 <- 3
  mu_1 <- 4
  sigma_0 <- i
  sigma_1 <- i
  alpha <- 0.05
  set.seed(0) # set random seed for reproducibility

  sigma_vals_stat_power_simulation_df <- data.frame(trial =
                                                      seq(num_trials)) %>%
    # generate random Gaussian samples
    mutate(sample_0 = map(.x = trial, .f =~ rnorm(n = n_0, mean = mu_0,
                                                  sd = sigma_0)),
           sample_1 = map(.x = trial, .f =~ rnorm(n = n_1, mean = mu_1,
                                                  sd = sigma_1))) %>%
    # for each sample, generate p value; check examples of pmap() with ?map
    mutate(p_value = pmap(.l = list(trial, sample_0, sample_1),
```

```
                          .f =~ t.test(..2, ..3, var.equal =
                                         TRUE)$p.value)) %>%
    # estimate of coverage probability
    mutate(reject_null = p_value < alpha )

  prob <- sigma_vals_stat_power_simulation_df %>%
    # extract a column
    pull(reject_null) %>%
    # compute probability
    mean()

  sigmaVec <- c(sigmaVec, i)
  probs <- c(probs, prob)

  probsDf4 <- data.frame(sigma = sigmaVec, prob = probs)
}
head(probsDf4)
```

```
##   sigma   prob
## 1     1 0.9667
## 2     2 0.4862
## 3     3 0.2501
## 4     4 0.1621
## 5     5 0.1212
## 6     6 0.0996
```

```
ggplot(probsDf4, aes(x = sigma, y = prob)) +
  geom_point() +
  geom_line() +
  theme_bw() +
  labs(x = "Standard Deviation", y = "Statistical Power")
```



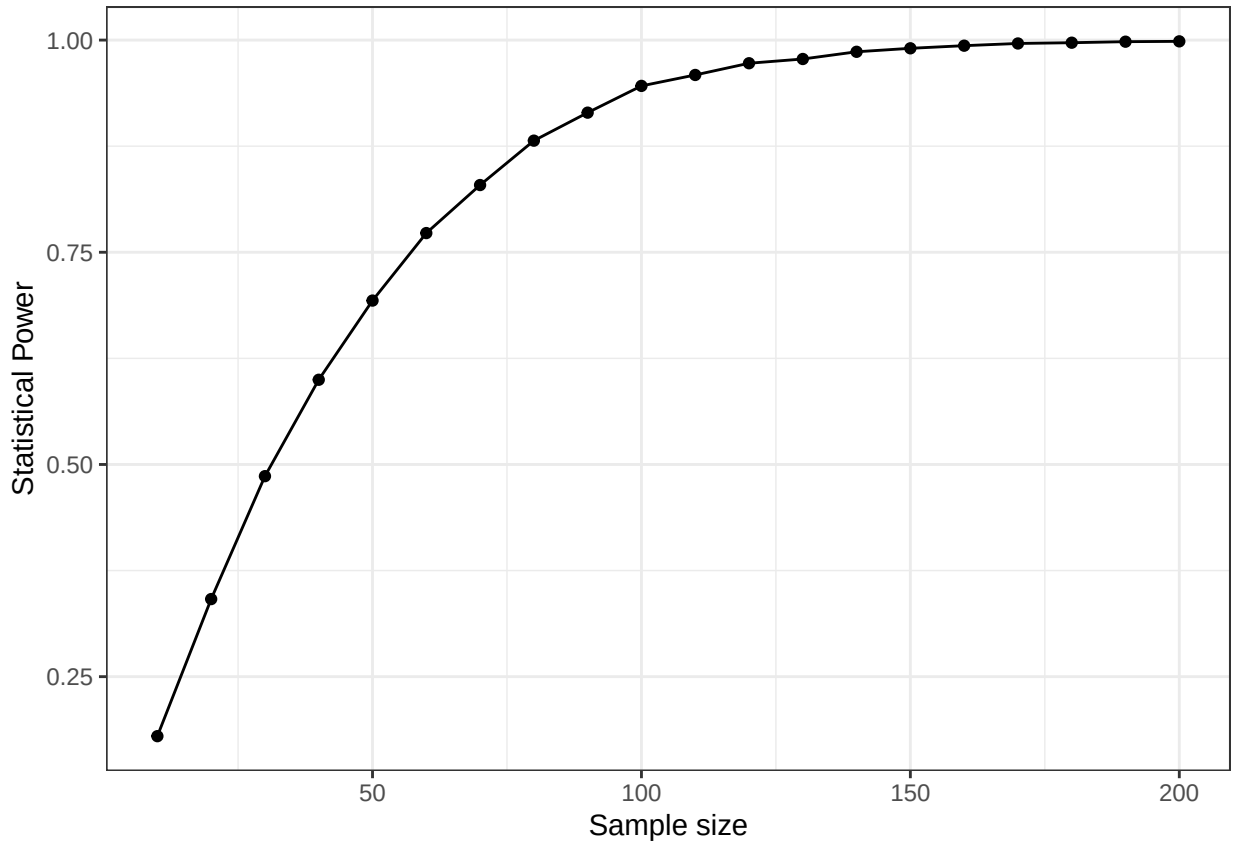
6 (Q4)

```
sampleSizes <- seq(10, 200, 10)

ssVec <- c()
probs <- c()

for (i in sampleSizes) {
  num_trials <- 10000
  n_0 <- i
  n_1 <- i
  mu_0 <- 3
  mu_1 <- 4
  sigma_0 <- 2
  sigma_1 <- 2
  alpha <- 0.05
  set.seed(0) # set random seed for reproducibility

  sample_sizes_stat_power_simulation_df <- data.frame(trial =
                                                        seq(num_trials)) %>%
    # generate random Gaussian samples
    mutate(sample_0 = map(.x = trial, .f =~ rnorm(n = n_0, mean = mu_0,
                                                  sd = sigma_0)),
           sample_1 = map(.x = trial, .f =~ rnorm(n = n_1, mean = mu_1,
```

```
                                                sd = sigma_1))) %>%
    # for each sample, generate p value; check examples of pmap() with ?map
    mutate(p_value = pmap(.l = list(trial, sample_0, sample_1),
                          .f =~ t.test(..2, ..3, var.equal =
                                         TRUE)$p.value)) %>%
    # estimate of coverage probability
    mutate(reject_null = p_value < alpha )

  prob <- sample_sizes_stat_power_simulation_df %>%
    # extract a column
    pull(reject_null) %>%
    # compute probability
    mean()

  ssVec <- c(ssVec, i)
  probs <- c(probs, prob)

  probsDf5 <- data.frame(sampleSize = ssVec, prob = probs)
}
head(probsDf5)
```

```
##   sampleSize   prob
## 1         10 0.1799
## 2         20 0.3414
## 3         30 0.4862
## 4         40 0.5998
## 5         50 0.6931
## 6         60 0.7726
```

```
ggplot(probsDf5, aes(x = sampleSize, y = prob)) +
  geom_point() +
  geom_line() +
  theme_bw() +
  labs(x = "Sample size", y = "Statistical Power")
```



7. Comparing the paired and unpaired t-tests on paired data

7 (Q1)

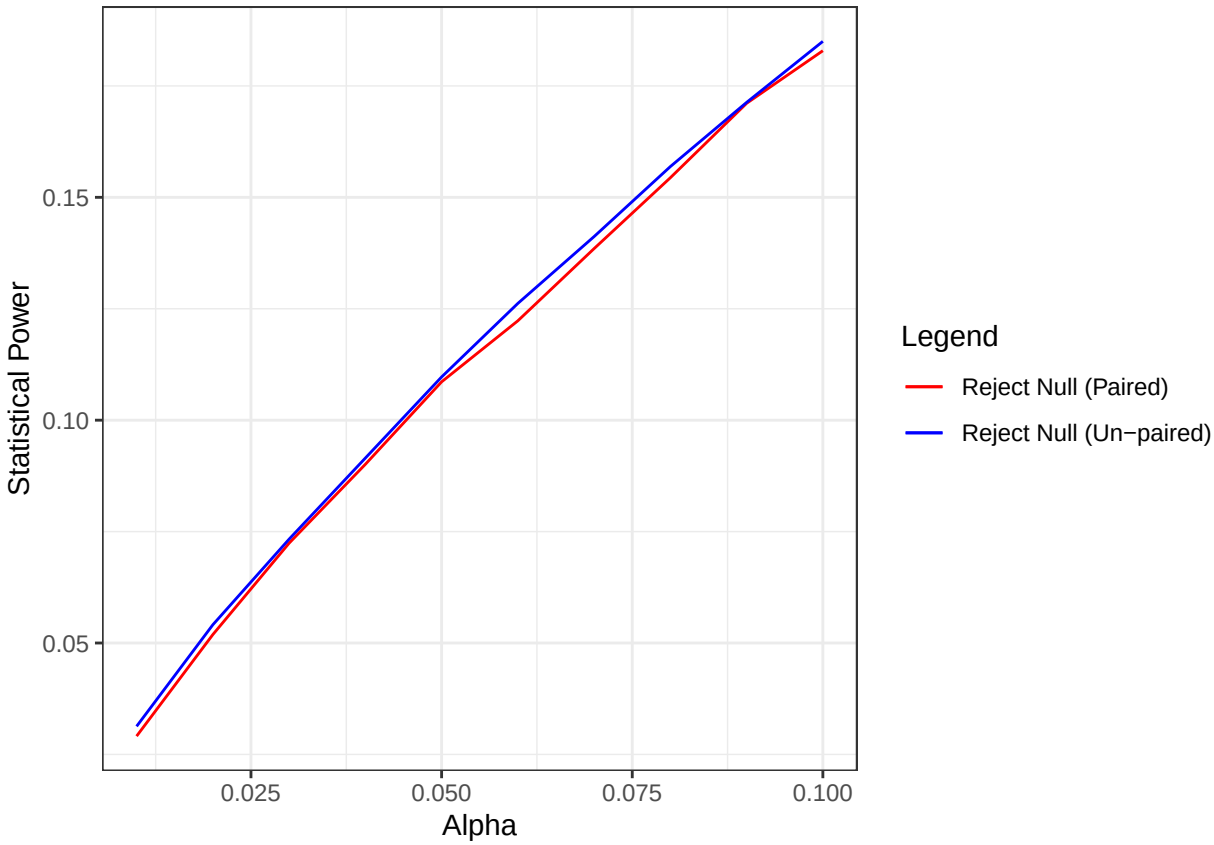
```
alphas <- seq(0.01, 0.1, 0.01)

alphasVec <- c()
statPower1 <- c()
statPower2 <- c()

for (i in alphas) {
  num_trials <- 10000
  n_0 <- 30
  n_1 <- 30
  mu_0 <- 10
  mu_1 <- 11
  sigma_0 <- 5
  sigma_1 <- 6
  alpha <- i
  set.seed(0) # set random seed for reproducibility

  alphas_stat_power_simulation_df <- data.frame(trial =
                                                  seq(num_trials)) %>%
```

```
    # generate random Gaussian samples
    mutate(sample_0 = map(.x = trial, .f =~ rnorm(n = n_0, mean = mu_0,
                                                  sd = sigma_0)),
           sample_1 = map(.x = trial, .f =~ rnorm(n = n_1, mean = mu_1,
                                                  sd = sigma_1))) %>%
    # for each sample, generate p value; check examples of pmap() with ?map
    mutate(p_value = pmap(.l = list(trial, sample_0, sample_1),
                          .f =~ t.test(..2, ..3, paired = T, var.equal =
                                       T)$p.value)) %>%
    mutate(p_val_up = pmap(.l = list(trial, sample_0, sample_1),
                          .f =~ t.test(..2, ..3, paired = F, var.equal =
                                       T)$p.value)) %>%
    # estimate of coverage probability
    mutate(reject_null = p_value < alpha) %>%
    mutate(reject_null_up = p_val_up < alpha)

  prob1 <- alphas_stat_power_simulation_df %>%
    # extract a column
    pull(reject_null) %>%
    # compute probability
    mean()

  prob2 <- alphas_stat_power_simulation_df %>%
    pull(reject_null_up) %>%
    mean()

  alphasVec <- c(alphasVec, i)
  statPower1 <- c(statPower1, prob1)
  statPower2 <- c(statPower2, prob2)

  probsDf6 <- data.frame(alpha = alphasVec, statPower1 = statPower1, statPower2 = statPower2)
}
head(probsDf6)
```

```
##   alpha statPower1 statPower2
## 1  0.01     0.0291     0.0313
## 2  0.02     0.0519     0.0541
## 3  0.03     0.0724     0.0733
## 4  0.04     0.0901     0.0915
## 5  0.05     0.1086     0.1097
## 6  0.06     0.1223     0.1262
```

```
colors <- c("Reject Null (Paired)" = "red", "Reject Null (Un-paired)" = "blue")
ggplot(probsDf6, aes(x = alpha)) +
  geom_line(aes(y = statPower1, color = "Reject Null (Paired)")) +
  geom_line(aes(y = statPower2, color = "Reject Null (Un-paired)", type = 2)) +
  theme_bw() +
  scale_color_manual(name = "Legend", values = colors) +
  labs(x = "Alpha", y = "Statistical Power")
```

```
## Warning: Ignoring unknown aesthetics: type
```


Statistical Power
0.15
0.10
0.05
0.025
0.050
0.075
0.100
Alpha
Legend
Reject Null (Paired)
Reject Null (Un−paired)