Causal Inference

INFO 4100 Learning Analytics

[Shuhui Zhu, sz649]]

In this homework, you will learn how to randomize individuals and analyze data from experiments, sometimes called A/B tests.

Learning Objectives:

1. Understand the difference between simple, complete, and block random assignment, and know how to implement them
2. Check balance of an experiment
3. Analyze experimental data using a t-test, linear regression, and Wilcox test.
4. Report results of an experiment

# Surprise Part 0: Grit Experiment

Recall the Grit Activity at the end of the Unsupervised Learning week? It was a randomized experiment!

Study design: Half the students were randomized to the “video” condition in which they saw the video TED talk about Grit. The other half was randomized to the “text” condition in which they saw the written transcript of the same talk. Then every student took the same knowledge recall test.

How were you randomized? Here is the code that I used. I started off with a list of course grades up to now. I used a pair-wise blocked design as you can see below.

random\_pair\_assignment = grades %>%   
 select(Username, Grade) %>%  
 filter(Grade > .25) %>% # remove staff/drops  
 arrange(Grade) %>% # sort by grade  
 mutate(pair = rep(1:(n()/2), each=2)) %>% # assign pairs  
 group\_by(pair) %>% # for each pair ...  
 mutate(condition = sample(c("video", "text"))) # randomize label

Now, load the results data and you can see what the outcome was. Did students remember more if they watched the video or if they read the text? Before you start, what do you think is better?

grit = read.csv("grit\_assign4100fa20score\_anon.csv")  
table(grit$condition) # 45 students in each condition

##   
## text video   
## 45 45

table(grit$pair) # 45 pairs

##   
## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26   
## 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2   
## 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45   
## 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

table(grit$pair, grit$condition) # 1 student each

##   
## text video  
## 1 1 1  
## 2 1 1  
## 3 1 1  
## 4 1 1  
## 5 1 1  
## 6 1 1  
## 7 1 1  
## 8 1 1  
## 9 1 1  
## 10 1 1  
## 11 1 1  
## 12 1 1  
## 13 1 1  
## 14 1 1  
## 15 1 1  
## 16 1 1  
## 17 1 1  
## 18 1 1  
## 19 1 1  
## 20 1 1  
## 21 1 1  
## 22 1 1  
## 23 1 1  
## 24 1 1  
## 25 1 1  
## 26 1 1  
## 27 1 1  
## 28 1 1  
## 29 1 1  
## 30 1 1  
## 31 1 1  
## 32 1 1  
## 33 1 1  
## 34 1 1  
## 35 1 1  
## 36 1 1  
## 37 1 1  
## 38 1 1  
## 39 1 1  
## 40 1 1  
## 41 1 1  
## 42 1 1  
## 43 1 1  
## 44 1 1  
## 45 1 1

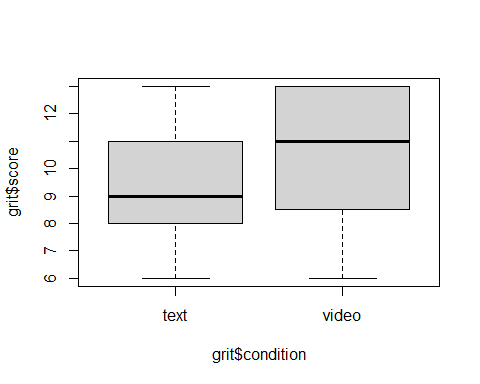
# missing data (some didn't do the activity!)  
table(is.na(grit$score))

##   
## FALSE TRUE   
## 53 37

# is missingness even acros condition?  
table(is.na(grit$score), grit$condition) # Yes 19v.18

##   
## text video  
## FALSE 26 27  
## TRUE 19 18

# how did student do depending condition  
boxplot(grit$score ~ grit$condition)



# is the difference significant?  
t.test(grit$score ~ grit$condition)

##   
## Welch Two Sample t-test  
##   
## data: grit$score by grit$condition  
## t = -1.2777, df = 50.783, p-value = 0.2072  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -2.1428459 0.4761792  
## sample estimates:  
## mean in group text mean in group video   
## 9.50000 10.33333

# Technicalluy, becasue I used blocked assignment  
# we need to account for the blocking structure  
# but there are not that many pairs where both   
# students completed the activity  
# However, it still shows that video seems better  
grit %>%   
 group\_by(pair) %>%  
 summarise(  
 c = paste(condition[1], "-", condition[2]),  
 s = score[1] - score[2]  
 ) %>%  
 filter(!is.na(s)) %>%  
 arrange(s)

## # A tibble: 14 x 3  
## pair c s  
## <int> <chr> <int>  
## 1 7 text - video -5  
## 2 28 text - video -5  
## 3 40 text - video -4  
## 4 43 text - video -4  
## 5 21 text - video -3  
## 6 31 text - video -3  
## 7 30 text - video -2  
## 8 15 text - video -1  
## 9 12 text - video 0  
## 10 18 text - video 0  
## 11 39 text - video 0  
## 12 9 text - video 1  
## 13 13 text - video 3  
## 14 42 text - video 4

# As you can see, for most complete pairs the text score  
# is lower than the video score

The data and code above was actually from last year’s class. Now, you can analyze the data for the experiment this semester (i.e. your data). One small difference is that this year I used simple random assignment (see explained below) to conditions. Load in the data (grit\_assign4100fa21score\_anon.csv), make a boxplot to visually inspect the distributions of scores in each condition, and then run a statistical test to see if the difference is significant.

From below, since the p-value is 0.037 which is smaller than 0.05. We could conclude that the difference is significant. Video cohort yields a better performance.

library(tidyr)  
### Add your analysis of this year's data here ###  
grit\_21 = read.csv('grit\_assign4100fa21score\_anon.csv')  
# check na  
table(is.na(grit\_21$score))

##   
## FALSE   
## 117

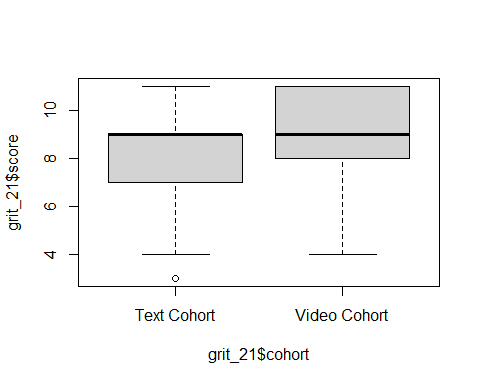
table(grit\_21$score) # 45 students in each condition

##   
## 3 4 5 6 7 8 9 10 11   
## 1 4 1 13 11 22 28 15 22

table(grit\_21$cohort, grit\_21$score)

##   
## 3 4 5 6 7 8 9 10 11  
## Text Cohort 1 3 0 10 5 13 17 9 7  
## Video Cohort 0 1 1 3 6 9 11 6 15

boxplot(grit\_21$score ~ grit\_21$cohort)



t.test(grit\_21$score ~ grit\_21$cohort)

##   
## Welch Two Sample t-test  
##   
## data: grit\_21$score by grit\_21$cohort  
## t = -2.1123, df = 111.53, p-value = 0.03689  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -1.40882083 -0.04502532  
## sample estimates:  
## mean in group Text Cohort mean in group Video Cohort   
## 8.215385 8.942308

# Part 1: Random Assignment

The simplest way to randomize is by the toss of a coin. For each person, you toss the coin and assign based on how it lands. This is called **simple random assignment**. Each toss is distributed as a Bernoulli trial with a probability parameter p. A fair coin has p=0.5. If you repeat this n times, the distribution of heads/successes is distributed as a Binomial random variable with p=0.5. To implement simple random assignment in R, use the rbinom() function. It takes 3 parameters as input: n the number of observations (i.e. how many participants/students to assign), size the number of trials (i.e. how many times you assign each person; almost always just once), and p the probability of success in each trial (0.5 for a fair coin).

One of the downsides of simple random assignment is that you can end up with uneven experimental group sizes by chance. There is no guarantee that in 30 coin tosses you will get 15 heads. (You can plot the distribution of how many heads you will get using the rbinom() function and plot it as a histogram; not required, but worth trying.)

To ensure that you have exactly half of the sample assigned to one of two conditions, you can specify a priori how many heads and how many tails you want. All you need to then is to randomize their order. This is called **complete random assignment**. To implement complete random assignment in R, use the rep() function to get a vector of 1s and 0s, e.g. if you randomize 30 learners you want 15 in the treatment condition (1s) and 15 in the control condition (0s). Once you have this vector, you can shuffle its order using the sample() function.

Random assignment will balance all covariates in expectation. This means that if you randomly assign (simple or complete) the sample infinitely many times, then people assigned to each condition will be identical on average. In practice, however, you just use one assignment for your experiment, so this guarantee is not so useful.

To make sure that a single assignment is balanced on particular covariates, you can to use **block random assignment**. Just like complete random assignment forces the number of 1s and 0s to be the same, block random assignment additionally forces the assignment to be balanced. Suppose you are assigning 100 learners and want to balance assignment on whether they are a transfer student. There are 40 transfer students and 60 non-transfer students. To implement block random assignment, you separately randomize the transfer students using complete random assignment with 20 1s and 20 0s, and the non-transfer students with 30 1s and 30 0s. To implement this in R, you use the same method as for complete random assignment, but you apply it once for the transfer students and then for the non-transfer students.

Below I simulate a simple, student-level dataset (dat) for you that has the variable transfer to block on. I save the data into a csv for you, so that everyone is working with the same data for the rest of the homework (note that if you run the code below, each time you get different values).

# Set sample size  
n = 500  
  
# Simulate data  
dat = data.frame(  
 sid = 1:n,  
 age = rnorm(n, mean = 20, sd = 2),  
 prev\_grade = runif(n, min = 0, max = 100),  
 transfer = rep(0:1, c(300, 200))  
)  
summary(dat)  
  
# write.csv(dat, file="info4100.inference.data.csv")

**Question 1:** Now it is your turn to implement the three types of random assignment: simple, complete, and blocked (on transfer status). For each one, save the assignment into a new column in the dataset.

dat = read.csv("info4100.inference.data.csv")  
  
###############################################   
####### BEGIN INPUT: Random Assignment ########  
###############################################  
n = nrow(dat)  
n

## [1] 500

# Simple random assignment  
dat$cond\_simple = rbinom(n, 1, 0.5)  
  
# Complete random assignment  
dat$cond\_complete = rep(c(1,0), each = n/2)  
dat$cond\_complete = sample(dat$cond\_complete, n)  
  
  
dat$cond\_blocked =c(sample(rep(c(1,0),150)),sample(rep(c(1,0),100)))  
  
# Blocked (on transfer status) random assignment  
dat\_1 = dat[which(dat$transfer == 1),]  
dat\_n\_1 = length(dat\_1$transfer)  
dat\_1$cond\_blocked = rep(c(1,0), dat\_n\_1/2)  
dat\_1$cond\_blocked = sample(dat\_1$cond\_blocked)  
  
dat\_0 = dat[which(dat$transfer == 0),]  
dat\_n\_0 = length(dat\_0$transfer)  
dat\_n\_0

## [1] 300

dat\_0$cond\_blocked = rep(c(1,0), dat\_n\_0/2)  
dat\_0$cond\_blocked = sample(dat\_0$cond\_blocked)  
dat = rbind(dat\_1, dat\_0)  
###############################################  
###############################################

# Part 2: Checking Balance

There are a few things to check after random assignment and ideally before deploying the experiment (so you can fix issues ahead of time). First, you want to check how many units you assigned to each condition. You can do this by tabulating the condition indicator.

Second, it is good practice to check if random assignment was successful in balancing covariates across groups. In the toy dataset you have three covariates: age, prev\_grade, and transfer status. To evaluate balance, you can compare the standardized difference between conditions. Standardization (or z-scoring) is a monotonic transformation that allows for comparing variables with different distributions.

For each variable, you want to compute the difference in means (treatment - control) and divide it by the pooled standard deviation (SD). For a continuous variable, you can compute the SD as follows: sqrt((variance0 + variance1) / 2), where sqrt() is the square-root, and the variances are computed using var() for participants in each condition. For a binary variable, the variance is computed in terms of the proportion of 1s for participants in each condition as follows: sqrt((p1 \* (1-p1) + p0 \* (1-p0))/2), where p1 is the proportion of 1s in the treatment group, and p0 in the control group. An absolute standardized difference below 0.1 is considered good.

**Question 2:** For each of the three types of random assignment you implemented above, check (1) how many units are assigned to each condition, and (2) how well the three covariates are balanced. Compute balance by hand below using the formula provided above. Show what you find. (Tip: you can write a short function to compute balance and reuse it.)

###############################################   
####### BEGIN INPUT: Check balance ############  
###############################################  
   
# Check assignment to condition  
table(dat$cond\_simple)

##   
## 0 1   
## 255 245

table(dat$cond\_complete)

##   
## 0 1   
## 250 250

table(dat$cond\_blocked)

##   
## 0 1   
## 250 250

# Check balance function  
checkBalance <- function(var\_i, var\_cond\_i) {  
 dat\_0 = dat[which(dat[var\_cond\_i] == 0),]  
 dat\_1 = dat[which(dat[var\_cond\_i] == 1),]  
   
 mean\_diff = abs(sum(dat\_1[var\_i])/count(dat\_1[var\_i]) - sum(dat\_0[var\_i])/count(dat\_0[var\_i]))  
   
 if(length(unique(dat[var\_i]))!=2) {  
 variance0 = var(dat\_0[var\_i])  
 variance1 = var(dat\_1[var\_i])  
 sd = sqrt((variance0 + variance1) / 2)  
 }else{  
 p1 = mean(dat\_1[var\_i])  
 p0 = mean(dat\_0[var\_i])  
 sd = sqrt((p1 \* (1-p1) + p0 \* (1-p0))/2)  
 }  
   
 return(mean\_diff/sd)  
}  
# Check balance: simple  
for (i in c('age', 'prev\_grade', 'transfer')){  
 ret\_c = checkBalance(i, 'cond\_simple')  
 print(ret\_c)  
}

## age  
## 1 0.003488511  
## prev\_grade  
## 1 0.07502379  
## transfer  
## 1 0.03260966

# Check balance: complete  
for (i in c('age', 'prev\_grade', 'transfer')){  
 ret\_c = checkBalance(i, 'cond\_complete')  
 print(ret\_c)  
}

## age  
## 1 0.07124228  
## prev\_grade  
## 1 0.1705068  
## transfer  
## 1 0.06522375

# Check balance: block  
for (i in c('age', 'prev\_grade', 'transfer')){  
 ret\_c = checkBalance(i, 'cond\_blocked')  
 print(ret\_c)  
}

## age  
## 1 0.05609594  
## prev\_grade  
## 1 0.0301448  
## transfer  
## 1 0

# From results, overall, good balance below 0.1. For transfer, the balance for simple and complete are slightly above 0.1.   
###############################################  
###############################################

# Part 3: Analysis

The simplest way to analyze an experiment is using the difference in means estimator, which is the difference in the mean outcome in treatment and control. You can do this in R with a t-test using t.test(). For a binary outcome, this is a test of proportions, which you can do in R using prop.test(). Note that prop.test() does NOT use the formula syntax like t.test() or lm(). Check the help by running ?prop.test() to see how to use it. In either case, you want to focus on the p-value. If the p-value is below 0.05 then the difference is statistically significant, otherwise it is not.

If you have a covariate that is predictive of the outcome, you can use it to increase the precision of your estimator. To do this, you can fit a linear regression like this lm(outcome ~ cond + good\_predictor) and inspect the results using summary(). Focus on the p-value for the coefficient on the condition indicator variable in the regression.

If your outcome variable is very skewed and even multi-modal, you can use a non-parametric test like the Wilcox test. This test considers the relative rank of values rather than their value itself. In R, you can use wilcox.test() to implement it.

No matter which test you use, the output will provide a p-value. The conditional probability of observing an outcome this extreme or more extreme given that there is no effect. Thus, if the p-value is small it means that it is unlikely for a difference between treatment and control to be due to chance. By convention, we reject the null hypothesis that a difference is just due to chance if the p-value is below 0.05 (or 5%).

Below I simulate student grade (grade) as an outcome variable to use in the analysis. That is, suppose you ran the experiment and got back this outcome data. Previous grades are a good predictor of future grades. This will be true here too so you can use prev\_grade as a covariate in the regression model to improve precision. Note that I am using the cond\_simple assignment, not the others. So you can try out the test of proportion, I am adding a binary outcome pass (getting a passing grade).

# grade = prev\_grade + noise + treatment effect  
dat$grade = dat$prev\_grade + rnorm(nrow(dat), 0, 50) + dat$cond\_simple \* 20  
dat$pass = as.numeric(dat$grade > 0)

**Question 3:** Analyze the experiment using the methods described above. You are interested in the effect of the treatment (cond\_simple) on the outcome (grade or pass). Use the t.test(), prop.test(), lm(), and wilcox.test() functions to analyze the experiment.

###############################################   
####### BEGIN INPUT: Perform analysis #########  
###############################################  
dat\_0 = dat[which(dat['cond\_simple'] == 0),]  
dat\_1 = dat[which(dat['cond\_simple'] == 1),]  
length(dat\_1$pass)

## [1] 245

# t-test  
t.test(dat\_0$grade, dat\_1$grade)

##   
## Welch Two Sample t-test  
##   
## data: dat\_0$grade and dat\_1$grade  
## t = -4.6775, df = 497.76, p-value = 3.746e-06  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -34.01140 -13.89071  
## sample estimates:  
## mean of x mean of y   
## 48.21229 72.16335

# subset = dat[c("pass", "cond\_simple")]  
# table = table(subset)  
# prop.test(table)  
  
# Test of proportion  
prop.test(x = c(sum(dat\_0$pass), sum(dat\_1$pass)), n = c(length(dat\_0$pass),length(dat\_1$pass)))

##   
## 2-sample test for equality of proportions with continuity correction  
##   
## data: c(sum(dat\_0$pass), sum(dat\_1$pass)) out of c(length(dat\_0$pass), length(dat\_1$pass))  
## X-squared = 2.9004, df = 1, p-value = 0.08856  
## alternative hypothesis: two.sided  
## 95 percent confidence interval:  
## -0.123955962 0.007749479  
## sample estimates:  
## prop 1 prop 2   
## 0.8235294 0.8816327

# Regression adjusting for prev grade  
m = lm(dat$grade ~ dat$cond\_simple + dat$prev\_grade)  
summary(m)

##   
## Call:  
## lm(formula = dat$grade ~ dat$cond\_simple + dat$prev\_grade)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -156.945 -35.622 1.736 33.122 152.812   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 10.97291 5.26404 2.085 0.0376 \*   
## dat$cond\_simple 22.32490 4.74907 4.701 3.36e-06 \*\*\*  
## dat$prev\_grade 0.77037 0.08447 9.120 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 53.05 on 497 degrees of freedom  
## Multiple R-squared: 0.1794, Adjusted R-squared: 0.1761   
## F-statistic: 54.32 on 2 and 497 DF, p-value: < 2.2e-16

# Wilcox test  
wilcox.test(dat\_0$grade, dat\_1$grade)

##   
## Wilcoxon rank sum test with continuity correction  
##   
## data: dat\_0$grade and dat\_1$grade  
## W = 24096, p-value = 9.798e-06  
## alternative hypothesis: true location shift is not equal to 0

###############################################  
###############################################

# Part 4: Reporting

After you have analyzed the experiment, you need to report its results. Most commonly, you report the effect size with the p-value. There are many measures of effect size. The two most common ones are Cohen’s d and percentage change.

1. Percentage change: by what percentage is the mean outcome in the treatment higher/lower than in the control
2. Cohen’s d: how many standard deviations difference is there between the treatment and control (this is the same method we used for checking covariate balance above)

**Question 4:** Compute percentage change and Cohen’s d for the experiment and summarize the result of the experiment in one sentence (citing percentage increase, d, and the p-value). Focus here only on the grade outcome with cond\_simple assignment.

###############################################   
####### BEGIN INPUT: Stats for Report #########  
###############################################  
  
# Percentage change  
dat\_0\_g = mean(dat\_0$grade)  
dat\_1\_g = mean(dat\_1$grade)  
  
pct\_chg = (dat\_1\_g - dat\_0\_g)/dat\_0\_g  
print(pct\_chg)

## [1] 0.4967833

# Cohen's d  
ret\_c = checkBalance('grade', 'cond\_simple')  
print(ret\_c)

## grade  
## 1 0.4183801

# Summary sentence  
# The p-value of effect size is 3.746e-06, which is much smaller than 0.05. Therefore the effect is significant. The mean grade of treatment group is 49.67% higher than control group. Standard deviations difference between treatment and control group is 0.418.  
  
  
###############################################  
###############################################

# Self-reflection

**Briefly summarize your experience on this homework. What was easy, what was hard, what did you learn?**

* The instructions are a bit hard to follow in this homework. It took me a while to understand the instructions. I wish in discussion session, a more structured introduction of the methods could be given.
* Once understanding the instruction, coding was easy.
* I’m also confused about how much text we need to write out to answer the question beside script.

# Submit Homework

This is the end of the homework. Please **Knit a Doc(x) report** that shows both the R code and R output and upload it on the EdX platform.