Analyzing Longitudinal Data: Robust Variance and Clustered Bootstrap

Ashley I Naimi

Fall 2024

Contents

1 Classical versus Complex Longitudinal Data

In this course, you have already encountered causal inference via potential outcomes when exposure under study is measured once (i.e., time fixed). In this lecture, we will focus on longitudinal, and complex longitudinal data, and the complications that may arise when dealing with such data. For clarity, let's define complex longitudinal data. We will be dealing with data from a cohort study, individuals sampled from a well-defined target population, and clear study start and stop times (i.e., closed cohort). Data from such a cohort are **longitudinal** when they are measured repeatedly over time.[1]

[1] Another such form is when data are measured repeatedly within clusters defined by geographic space (e.g., census tracts) or some other grouping (e.g., hospitals). We will not be dealing with these data here, though the methods to handle such data are very similar and in some cases identical.

Different scenarios can lead to longitudinal data:

1. exposure and covariates do not vary over time, but the study outcome is measured repeatedly in the same individual over follow up
2. exposure and covariates vary over time and are measured repeatedly in the same individual over follow up, but the study outcome can only occur (and/or is measured) only once
3. exposure and covariates vary over time and are measured repeatedly in the same individual over follow up, and the study outcome can occur more than once, and is measured repeatedly in the same individual over follow up.

Scenario 1 is the classical situation that one might refer to as "longitudinal" or correlated (outcomes) data. In this scenario, researchers often use mixed effects models or generalized estimating equations to deal with these data, but one can sometimes use simpler methods depending on the problem's context.

Repeated exposure, covariate, and (possibly) outcome measurement also leads to "longitudinal" data. But these data can result in something fundamentally different, which we refer to here as complex longitudinal data.

Repeated measurement over time creates the opportunity for us to capture complex causal relations between past and future covariates. Suppose we measure an exposure twice over follow-up, a covariate once, and the outcome at the end of follow-up (Figure 1). If we can assume that past exposure/covariate values do not affect future exposure/covariate values (usually a very risky assumption), we might not consider these data "complex," because we can use many standard methods to obtain correct results.

On the other hand, if past exposure/covariates affect future exposure/covariates in such a way that prior exposures or covariates confound future exposures

Figure 1: Longitudinal data that might not be considered 'complex' because there is no feedback between exposure and covariates.

(Figure 2), more advanced analytic techniques are needed.

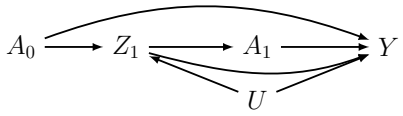


Figure 2: Causal diagram representing the relation between anti-retroviral treatment at time 0 (A_0), HIV viral load just prior to the second round of treatment (Z_1), anti-retroviral treatment status at time 1 (A_1), the CD4 count measured at the end of follow-up (Y), and an unmeasured common cause (U) of HIV viral load and CD4.

Here, we will learn why this distinction is important, and we'll cover a suite of methods that can be used to analyse data from each of these three scenarios.

2 Example 1: Simple Methods for Correlated Data

Standard regression models typically rely on the assumption that data are independent and identically distributed.

Suppose you had data on the BMI of 20 individuals. Let's say that these 20 individuals are picked randomly from the adult population in the United States.

Let's say one of the BMI values is 19.8 kg/m^2. Could you use this information to tell me anything about the other BMI values in the data?

Because of how these data were sampled, the answer to this question should be "no", you could not use this BMI to say anything about other BMI values in the data.

However, if I change the story a little, and told you that 10 of these data points were randomly selected women from the US Olympic Weightlifting Team (which included the 19.8 kg/m^2 data point), and the remaining ten were randomly selected women from Greene County, Alabama (the county with presumably the highest BMIs in the nation). On the basis of this information, you now know something more about what these data look like. You know that the BMI values from the Olympic Weightlifting Team will be closer to each other, and lower, than those from Greene County, Alabama. In effect, the BMI

values in each cluster of ten individuals are correlated.

But what do we really mean when we say that outcomes are correlated? To help make some concrete points, let's simulate 20 individuals' BMI based on the scenario above (10 from the Olympic Team, and 10 from Green County). We can do this in R:

```
set.seed(123)

bmi_data <- tibble(ID = 1:15,
                   BMI=c(rnorm(5,mean=20,sd=1),
                         rnorm(5,mean=34,sd=4),
                         rnorm(5,mean=12,sd=2)),
                   cluster=factor(c(rep(1,5),
                                    rep(2,5),
                                    rep(3,5))))

bmi_data %>% print(n=3)
```

```
## # A tibble: 15 x 3
##      ID   BMI cluster
##   <int> <dbl> <fct>
## 1     1  19.4 1
## 2     2  19.8 1
## 3     3  21.6 1
## # i 12 more rows
```

The code above simulates two groups of 100 BMI values. The first are generated from a normal distribution with a mean of 20 and standard deviation of 1 (the US Olympic Team). The second are generated from a normal distribution with a mean of 34 and a standard deviation of 4 (Greene County). The histogram in Figure 1 shows the distribution of BMI in these data. There's clearly an important separation between the BMI's from the two groups (by design). And while this simple example may not be very realistic, it will help us show precisely what we mean by the terms "correlated data", "clustered data" and the like.

So how can we evaluate clustering in our data? Typically, we use the **intra-**

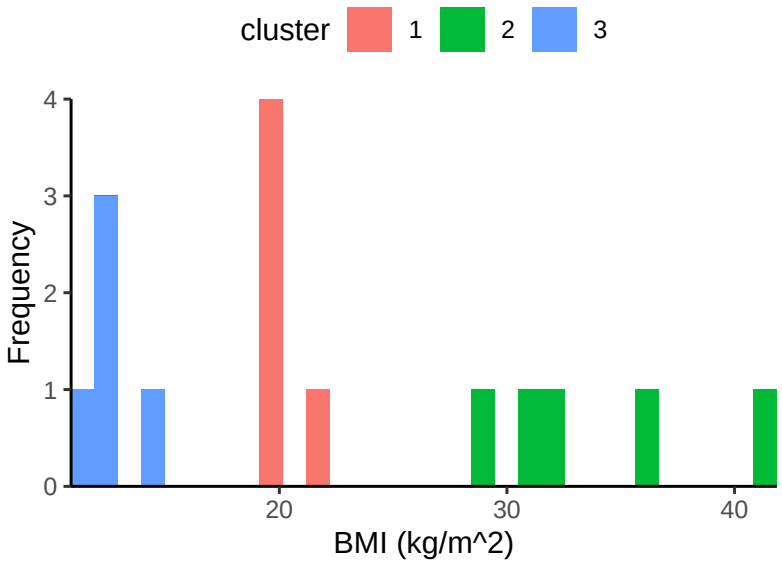


Figure 3: Histogram of Twenty BMI Values from Two Simulated Clusters

cluster correlation coefficient (ICC) to measure how correlated the data are in each cluster. Consider our hypothetical BMI data:

```
print(bmi_data, n = 1)
```

```
## # A tibble: 15 x 3
##      ID   BMI cluster
##   <int> <dbl> <fct>
## 1     1  19.4 1
## # i 14 more rows
```

These data have a total of 20 observations equally divided into two clusters: cluster 1 and cluster 2. This means that we can, for example, compute at least three means:

- the mean BMI in all 20 individuals
- the mean BMI in the 10 individuals in cluster 1
- the mean BMI in the 10 individuals in cluster 2

```
bmi_data$overall_mean <- mean(bmi_data$BMI)

bmi_data$cluster_mean <- c(rep(mean(bmi_data[bmi_data$cluster == 1,]$BMI), 5),
```

```
                          rep(mean(bmi_data[bmi_data$cluster == 2,]$BMI), 5),
                          rep(mean(bmi_data[bmi_data$cluster == 3,]$BMI), 5))

bmi_data$squares_within <- (bmi_data$BMI - bmi_data$overall_mean)^2

bmi_data$squares_between <- (bmi_data$BMI - bmi_data$cluster_mean)^2

kable(bmi_data)
```

ID	BMI	cluster	overall_mean	cluster_mean	squares_within	squares_between
1	19.43952	1	22.2107	20.19357	7.6794105	0.5685852
2	19.76982	1	22.2107	20.19357	5.9578794	0.1795622
3	21.55871	1	22.2107	20.19357	0.4250922	1.8636019
4	20.07051	1	22.2107	20.19357	4.5804170	0.0151442
5	20.12929	1	22.2107	20.19357	4.3322740	0.0041322
6	40.86026	2	22.2107	33.82272	347.8061132	49.5269107
7	35.84366	2	22.2107	33.82272	185.8577496	4.0842014
8	28.93976	2	22.2107	33.82272	45.2801917	23.8433867
9	31.25259	2	22.2107	33.82272	81.7557624	6.6055966
10	32.21735	2	22.2107	33.82272	100.1331011	2.5772192
11	14.44816	3	22.2107	12.61580	60.2569602	3.3575437
12	12.71963	3	22.2107	12.61580	90.0804406	0.0107795
13	12.80154	3	22.2107	12.61580	88.5322237	0.0344991
14	12.22137	3	22.2107	12.61580	99.7867907	0.1555814
15	10.88832	3	22.2107	12.61580	128.1963239	2.9842070

```
sum(bmi_data$squares_within)
```

```
## [1] 1250.661
```

```
sum(bmi_data$squares_between)
```

```
## [1] 95.81095
```

With these three means, we can also compute three measures of variability:

```
## the sum of squares overall
overall_variability <- sum((bmi_data$BMI - mean(bmi_data$BMI))^2)

overall_variability
```

```
## [1] 1250.661
```

```
## the sum of squares in cluster 1
cluster1_variability <- sum((bmi_data[bmi_data$cluster == 1,]$BMI -
                               mean(bmi_data[bmi_data$cluster == 1,]$BMI))^2)

cluster1_variability
```

```
## [1] 2.631026
```

```
## the sum of squares in cluster 2
cluster2_variability <- sum((bmi_data[bmi_data$cluster == 2,]$BMI -
                               mean(bmi_data[bmi_data$cluster == 2,]$BMI))^2)

cluster2_variability
```

```
## [1] 86.63731
```

```
## the sum of squares in cluster 3
cluster3_variability <- sum((bmi_data[bmi_data$cluster == 3,]$BMI -
                               mean(bmi_data[bmi_data$cluster == 3,]$BMI))^2)

cluster3_variability
```

```
## [1] 6.542611
```

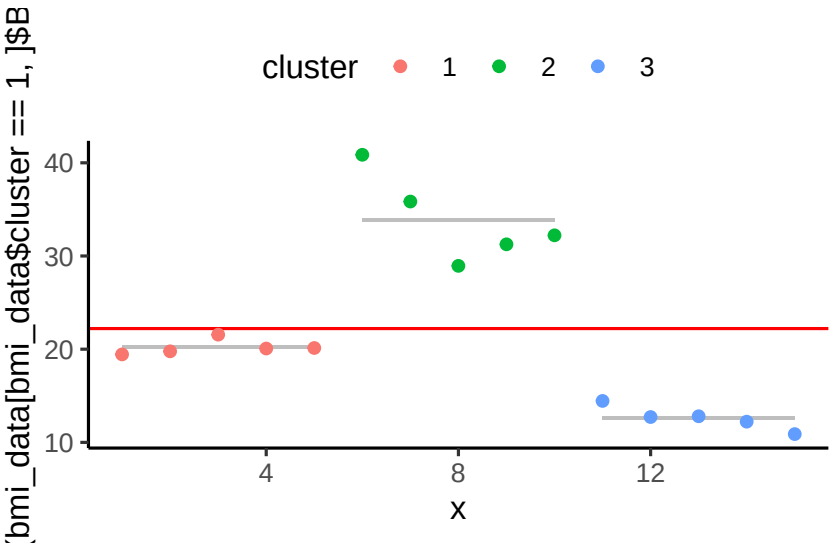
```
cluster1_variability + cluster2_variability + cluster3_variability
```

```
## [1] 95.81095
```

```
overall_variability / (overall_variability + cluster1_variability + cluster2_variability + cluster3_vari
```

```
## [1] 0.928843
```

```
ggplot2::ggplot(bmi_data) +
  geom_hline(yintercept = mean(bmi_data$BMI), color = "red") +
  geom_segment(aes(x = 1, xend = 5,
                   y = mean(bmi_data[bmi_data$cluster==1,]$BMI),
                   yend = mean(bmi_data[bmi_data$cluster==1,]$BMI)),
               color = "gray") +
  geom_segment(aes(x = 6, xend = 10,
                   y = mean(bmi_data[bmi_data$cluster==2,]$BMI),
                   yend = mean(bmi_data[bmi_data$cluster==2,]$BMI)),
               color = "gray") +
  geom_segment(aes(x = 11, xend = 15,
                   y = mean(bmi_data[bmi_data$cluster==3,]$BMI),
                   yend = mean(bmi_data[bmi_data$cluster==3,]$BMI)),
               color = "gray") +
  geom_point(aes(y = BMI,
                 x = ID,
                 group = cluster,
                 color = cluster))
```



For a continuous outcome variable like BMI, the ICC can be obtained using ANOVA, which is easy in R:

```
bmi_summary <- summary(aov(BMI ~ cluster,data=bmi_data))

bmi_summary
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## cluster      2 1154.8   577.4   72.32 2.02e-07 ***
## Residuals   12   95.8     8.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
bmi_summary[[1]][1,2]
```

```
## [1] 1154.85
```

```
bmi_summary[[1]][,2]
```

```
## [1] 1154.84978   95.81095
```

```
icc <- bmi_summary[[1]][1,2]/sum(bmi_summary[[1]][,2])

icc
```

```
## [1] 0.9233917
```

This tells us that roughly 92% of the variation in these data is occurring between clusters. This implies that within clusters, the magnitude of the variation is not as large. If, of all the variability in the data, most of it is occurring between clusters, then there's not much left over to occur within clusters. Subsequently, higher ICC values suggest more **similarity** within clusters than between clusters. In other words, individuals in these data look more similar to one another within each cluster, and are quite different from each other across clusters. This is what we would have expected given how we simulated the data.

In contrast, look at what happens if we simulate a different dataset with the same type of individuals in each cluster (i.e., no clustering):

```
set.seed(123)

bmi_data <- tibble(BMI = rnorm(20,mean=25,sd=5),
                   cluster=factor(c(rep(1,10),
                                    rep(2,10))))

head(bmi_data)
```

```
## # A tibble: 6 x 2
##     BMI cluster
##   <dbl> <fct>
## 1  22.2 1
## 2  23.8 1
## 3  32.8 1
## 4  25.4 1
## 5  25.6 1
## 6  33.6 1
```

```
tail(bmi_data)
```

```
## # A tibble: 6 x 2
##     BMI cluster
##   <dbl> <fct>
## 1  22.2 2
## 2  33.9 2
## 3  27.5 2
## 4  15.2 2
## 5  28.5 2
## 6  22.6 2
```

```
bmi_summary <- summary(aov(BMI ~ cluster,data=bmi_data))
```

```
icc <- bmi_summary[[1]][1,2]/sum(bmi_summary[[1]][,2])

icc
```

```
## [1] 0.004994307
```

In the above code, we simulated all 20 values from the same normal distribution with a mean of 25 and a standard deviation of 5. In other words, there is no (statistical) difference in the individuals across clusters. In this case, we obtain an ICC value of 0%. Finally, here's what happens if the cluster perfectly predicts BMI values:

```
set.seed(123)

bmi_data <- tibble(BMI=c(rep(25,10),rep(30,10)),
                   cluster=factor(c(rep(1,10),
                                    rep(2,10))))

head(bmi_data)
```

```
## # A tibble: 6 x 2
##     BMI cluster
##   <dbl> <fct>
## 1    25 1
## 2    25 1
## 3    25 1
## 4    25 1
## 5    25 1
## 6    25 1
```

```
tail(bmi_data)
```

```
## # A tibble: 6 x 2
##     BMI cluster
##   <dbl> <fct>
```

```
## 1    30 2
## 2    30 2
## 3    30 2
## 4    30 2
## 5    30 2
## 6    30 2
```

```
bmi_summary <- summary(aov(BMI ~ cluster,data=bmi_data))

icc <- bmi_summary[[1]][1,2]/sum(bmi_summary[[1]][,2])

icc
```

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

In the above code, we generated a BMI value of exactly 25 for all ten individuals in the first cluster, and a BMI value of exactly 30 for all ten individuals in the second cluster. Thus, cluster can perfectly predict the BMI value for everyone (i.e., there is no random variation), and we thus get an ICC value of 100%.

2.1 Are Correlated Data a Problem?

The answer to the question of whether correlated data are a problem depends entirely on the research question, and we are going to discuss the issues next. First, it's important to note that when we say "correlated data", what we typically refer too is correlated *outcome* data.

Specifically, suppose that your exposure of interest or your confounder adjustment set was highly correlated across several clusters, but the outcome you are studying is not correlated across these or any other clusters. If this is the case, you need not worry about correlated data. The problems with correlated data arise when the *outcome* under study is correlated. The specific "location" where this problem arises is in the process of trying to quantify the parameters of a model that we wish to fit. Take, for example, the following linear model

$$E(Y \mid X, C) = \beta_0 + \beta_1 X + \beta_2 C$$

Let's assume that in this example, Y represents BMI, X is some measure of diet (e.g., eat your vegetables versus don't eat your vegetables), and C is a confounder, and that we wanted to fit this model to the made up data in Table 1 (with only three observations, for simplicity):

ID	BMI (Y)	Vegetables (X)	Confounder (C)
1	21.0	0	1
2	32.7	1	0
3	25.8	1	1

Table 1: Some made up data for our likelihood function example

Typically, the objective here would be to get an estimate β_1, which we could interpret as a difference in BMI averages among those who eat their vegetables versus those who don't. An important statistical consideration is HOW we get these estimates. Many approaches exist, with one very common approach being maximum likelihood estimation.

Deeper Dive:

In introductory probability, you may have learned that the joint probability of two *independent* events [often denoted $P(A, B)$, and read "the probability of A and B] is equal to the product of their individual probabilities:

$$P(A, B) = P(A) \times P(B)$$

If, however, A and B are correlated, the above equation is no longer true. Instead, we'd have to use a more complicated form:

$$P(A, B) = P(A \mid B) \times P(B)$$

After choosing a distribution and link function, maximum likelihood estimation proceeds by specifying a likelihood for each person in the data, and multiplying all of these individual likelihoods together:

$$\underbrace{L(y;\beta)}_{\text{joint likelihood}} = \overbrace{L(21.0;\beta_0,\beta_2) \times L(32.7;\beta_0,\beta_1) \times L(25.8;\beta_0,\beta_1,\beta_2)}^{\text{product of individual likelihoods}}$$

What your computer software program (i.e., SAS, Stata, R, other) does is find values for β_1, β_2, and β_3 in the product of likelihoods that make joint likelihood as large as it can be with the data we have.

Though likelihoods are not probabilities, the two do share some properties (Pawitan, 2001). Specifically, if the outcomes are correlated, you cannot break up the joint likelihood into the product of individual likelihoods as in the equation above.

In the next sections, we're going to discuss some of the more practical implications of the problem that result from correlated outcomes. Using real data, we going to look at some different ways we can address the problems that arise.

2.2 Example Data with Correlated Outcomes

Here, we'll introduce the datasets we'll be using to illustrate some methods for dealing with correlated data.

The first example dataset is from a cluster randomized trial example in which 10 practices were randomly assigned to two treatment groups (patient centered care and normal care). Body mass index (kg/m^2) measured at year 1 of follow-up was the outcome. These data are available and described in Campbell (2006), but I obtained them from Mansournia et al. (2020):

```
cluster_trial <- read_csv(here("data", "cluster_trial_data_bmi.csv"))

cluster_trial %>%
    print(n = 5)
```

```
## # A tibble: 20 x 4
##      ID   BMI treatment practice
##   <dbl> <dbl>     <dbl>    <dbl>
## 1     1  26.2         1        1
## 2     2  27.1         1        1
## 3     3  25           1        2
## 4     4  28.3         1        2
## 5     5  30.5         1        3
## # i 15 more rows
```

The second example dataset is from a longitudinal (repeated outcome measure) study of the effect of a lead chelating agent (succimer) on blood

lead levels in children aged 12-33 months at enrollment. The data represent a random subset of 100 children from the original sample. Children were randomized at baseline to succimer or placebo, and blood lead levels were measured at weeks 0 (baseline), 1, 4, and 6. These data are available online,[2] and are described in Fitzmaurice et al. (2004):

[2] https://content.sph.harvard.edu/fitzmaur/ala/tlc.txt

```
lead_trial <- read_csv(here("data", "longitudinal_lead_data.csv"))

lead_trial <- gather(lead_trial, week, lead_value, L0:L6, factor_key = TRUE) %>%
    mutate(week = as.numeric(gsub("L", "", week))) %>%
    arrange(ID, week)

lead_trial %>%
    print(n = 8)
```

```
## # A tibble: 400 x 4
##      ID Treatment  week lead_value
##   <dbl> <chr>     <dbl>      <dbl>
## 1     1 P             0       30.8
## 2     1 P             1       26.9
## 3     1 P             4       25.8
## 4     1 P             6       23.8
## 5     2 A             0       26.5
## 6     2 A             1       14.8
## 7     2 A             4       19.5
## 8     2 A             6       21
## # i 392 more rows
```

We'll exclusively rely on the BMI data in this lecture (we won't have time to demonstrate with the longitudinal data). However, everything that I show you here can apply equivalently to either the BMI data or the lead data. If you are particularly interested, I'd encourage you to try to do the same analyses we present below with the longitudinal data.

2.3 Handling Correlated Outcome Data

We're going to focus today primarily on the BMI data. As a starting point, let's estimate the ICC to evaluate how correlated these outcomes are:

```
cluster_trial
```

```
## # A tibble: 20 x 4
##       ID   BMI treatment practice
##    <dbl> <dbl>     <dbl>    <dbl>
##  1     1  26.2         1        1
##  2     2  27.1         1        1
##  3     3  25           1        2
##  4     4  28.3         1        2
##  5     5  30.5         1        3
##  6     6  28.8         1        4
##  7     7  31           1        4
##  8     8  32.1         1        4
##  9     9  28.2         1        5
## 10    10  30.9         1        5
## 11    11  37           0        6
## 12    12  38.1         0        6
## 13    13  22.1         0        7
## 14    14  23           0        7
## 15    15  23.2         0        8
## 16    16  25.7         0        8
## 17    17  27.8         0        9
## 18    18  28           0        9
## 19    19  28           0       10
## 20    20  31           0       10
```

```
bmi_summary <- summary(aov(BMI ~ as.factor(practice),
                           data=cluster_trial)) # type II SS

icc <- bmi_summary[[1]][1,2]/sum(bmi_summary[[1]][,2])
```

```
icc
```

```
## [1] 0.9272896
```

With the BMI data, we get an intracluster correlation coefficient estimate of 93 indicating high levels of clustering in each practice.

So, with this high level clustering, the question is **what should we do about it?**

In the next sections, we'll explore what happens when we **ignore** clustered data, and how our results/conclusions compare when we use different methods to account for the clustering in the BMI trial data. These methods will include:

- Robust Standard Errors and Bootstrapping
- Generalized Estimating Equations
- Mixed Effects Models

The order of the techniques presented here is important. First, robust standard errors and (sometimes) the bootstrap are the **easiest** methods to implement when needing to deal with correlated outcomes. Generalized estimating equations are more complicated, and mixed effects models are most complicated. Second, the **assumptions** required for each of these methods to be valid generally increase in scope as we move down the list: robust standard errors and bootstrapping require generally fewer assumptions, while mixed effects models require the strongest set of assumptions.

Let's proceed with the clustered BMI data analysis. Let's conduct an analysis where we simply ignore the fact that the outcomes are correlated. We can fit a linear regression model, regressing BMI against the treatment. In R, we can do this using the `lm()` or `glm` functions. We can also use the `coefci` function from the `lmtest` package to easily get confidence intervals.

```
library(lmtest)

mod1 <- glm(BMI ~ treatment,
            data=cluster_trial,
```

```
            family=gaussian(link = "identity"))

coeftest(mod1)
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  28.3900     1.3466 21.0828   <2e-16 ***
## treatment     0.4200     1.9044  0.2205   0.8254
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coefci(mod1, level = 0.95)
```

```
##                 2.5 %    97.5 %
## (Intercept) 25.750719 31.029281
## treatment   -3.312507  4.152507
```

The above analysis tells us that the difference in average BMI between the patient centered care and the normal care groups is 0.42 kg/m^2. The standard error for this estimate is 1.9, which results in a p-value of 0.83 and 95% normal-interval (Wald) confidence intervals of -3.31, 4.15. These results are what we obtain when we ignore the clustering.

2.4 Robust Standard Errors

The above analysis just ignores the clustering of BMI across practices in the data. The easiest way to account for correlated outcomes, in this case due to clustering across practices, is to use robust or sandwich standard errors. There are several ways to do this in R. We'll use the `sandwich` package to implement these standard errors, and the `lmtest` package to get confidence intervals that can be modified to account for clustering.

```
library(lmtest)
library(sandwich)

mod1 <- glm(BMI ~ treatment,
            data = cluster_trial,
            family = gaussian(link = "identity"))

coeftest(mod1, vcov=vcovCL(mod1,
                           type = "HCO",
                           cadjust = F,
                           cluster = cluster_trial$practice))
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  28.3900     2.3239 12.2168   <2e-16 ***
## treatment     0.4200     2.4745  0.1697   0.8652
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coefci(mod1, vcov=vcovCL(mod1,
                         type = "HCO",
                         cadjust = F,
                         cluster = cluster_trial$practice),
       level = 0.95)
```

```
##                 2.5 %    97.5 %
## (Intercept) 23.835337 32.944663
## treatment   -4.429951  5.269951
```

What we see when we use methods to account for clustering is that the it is really the standard errors that are adjusted. This is particularly true when we use the robust variance estimator, or (as we will see) the bootstrap. It's also true with generalized estimating equations (GEE), but in the section on GEE, we'll also discuss how it's a bit more complicated.

It's important to discuss what can go wrong when we use the robust variance estimator. In particular, the most important threat to the validity of the robust variance estimator is small sample sizes. In our case, it's probably not a good idea to use the robust variance estimator we did (HC3). In R, small sample adjustments are implemented by default when using the `vcovCL()` function (Zei). Unfortunately, all of the methods we discuss here (robust variance, bootstrap, GEE, and mixed effects models) require "large" samples to get good performance.[3]

[3] This is almost universally true about any estimator in any setting. Ultimately, it depends on the complexity of the model/question that you have.

There are also many different variations of the robust variance estimator. These variations are typically referred to as HC, HC1, HC2, ... HC5. If you're interested, there is a paper by Mansournia et al in the IJE that explains these differences (Mansournia et al., 2020), and some more general features and properties of the robust variance estimator.

2.5 Clustered Bootstrap

Instead of using the robust standard error, a slightly more technical option is to use the **clustered** bootstrap. Essentially, the simplest clustered bootstrap approach proceeds in 4 steps:

1. Resample the data, with replacement, at the cluster level
2. Estimate the parameter of interest (in our case, the mean difference in BMI) and save the estimate
3. Repeat steps 1 and 2, 200 times
4. Take the standard deviation of all 200 estimates as the standard error of the estimate in the original (unsampled) data

In R, implementing the clustered bootstrap requires user written code to obtain the resampled data.[4] Here is some code that I've written in R to do a clustered bootstrap analysis:

[4] The `boot` package in R provides a range of boostrap estimators, but not for clustered data.

```
mod1$coefficients
```

```
## (Intercept)   treatment
##       28.39        0.42
```

```r
seed <- 123
set.seed(seed)

boot_func <- function(boot_num){

  clusters <- as.numeric(names(table(cluster_trial$practice)))
  index <- sample(1:length(clusters), length(clusters), replace=TRUE)
  bb <- table(clusters[index])
  boot <- NULL

  for(zzz in 1:max(bb)){
      cc <- cluster_trial[cluster_trial$practice %in% names(bb[bb %in% c(zzz:max(bb))]),]
      cc$b_practice<-paste0(cc$practice,zzz)
      boot <- rbind(boot, cc)
  }

  mod1 <- glm(BMI ~ treatment, data=boot,family=gaussian(link = "identity"))
  res <- cbind(boot_num,coef(mod1)[2])
  return(res)
}

boot_res <- lapply(1:750, function(x) boot_func(x))
boot_res <- do.call(rbind,boot_res)

head(boot_res)
```

```
##           boot_num
## treatment        1 -2.922222
## treatment        2  2.925000
## treatment        3  2.966667
## treatment        4  1.408333
## treatment        5 -2.761111
## treatment        6 -4.193636
```

```
tail(boot_res)
```

```
##           boot_num
## treatment      745  1.7333333
## treatment      746  0.1488095
## treatment      747 -2.3227273
## treatment      748  1.3000000
## treatment      749 -0.5800000
## treatment      750  1.0450000
```

```
mean(boot_res[,2])
```

```
## [1] 0.4690001
```

```
sd(boot_res[,2]) ## standard error of the treatment estimate
```

```
## [1] 2.677977
```

We can then use the standard normal-interval (Wald) estimator to get 95% confidence intervals with this bootstrapped standard error:

```
LCL <- mod1$coefficient[2] - 1.96*sd(boot_res[,2])
UCL <- mod1$coefficient[2] + 1.96*sd(boot_res[,2])

mod1$coefficient[2]
```

```
## treatment
##      0.42
```

```
LCL
```

```
## treatment
## -4.828835
```

```
UCL
```

```
## treatment
##  5.668835
```

What the above bootstrap code does is select, with replacement, practices in the `cluster_trial` data. Specifically, as we saw above, here's what the `cluster_trial` data look like:

```
head(cluster_trial)
```

```
## # A tibble: 6 x 4
##      ID   BMI treatment practice
##   <dbl> <dbl>     <dbl>    <dbl>
## 1     1  26.2         1        1
## 2     2  27.1         1        1
## 3     3  25           1        2
## 4     4  28.3         1        2
## 5     5  30.5         1        3
## 6     6  28.8         1        4
```

```
table(cluster_trial$practice)
```

```
##
##  1  2  3  4  5  6  7  8  9 10
##  2  2  1  3  2  2  2  2  2  2
```

The bootstrap code above randomly selects 10 practices from these data to create a "bootstrap resample". This bootstrap resample will contain 10 practices, but in the resample, some of the original practices may not be present, while others may be in the resample more than once. For example:

```
clusters <- as.numeric(names(table(cluster_trial$practice)))

index <- sample(1:length(clusters), length(clusters), replace=TRUE)
```

```
bb <- table(clusters[index])
boot <- NULL

for(zzz in 1:max(bb)){
    cc <- cluster_trial[cluster_trial$practice %in% names(bb[bb %in% c(zzz:max(bb))]),]
    cc$b_practice<-paste0(cc$practice,zzz)
    boot <- rbind(boot, cc)
}

head(boot)
```

```
## # A tibble: 6 x 5
##      ID   BMI treatment practice b_practice
##   <dbl> <dbl>     <dbl>    <dbl> <chr>
## 1     1  26.2         1        1 11
## 2     2  27.1         1        1 11
## 3     3  25           1        2 21
## 4     4  28.3         1        2 21
## 5     5  30.5         1        3 31
## 6     9  28.2         1        5 51
```

```
table(boot$b_practice)
```

```
##
## 101  11  21  31  32  33  51  61  62  91
##   2   2   2   1   1   1   2   2   2   2
```

By resampling this way, the "within-practice" correlation structure is respected, and we are thus able to obtain standard errors that appropriately account for clustering.

Again, it's important to discuss what can go wrong when we use a clustered bootstrap estimator. In my experience, many applied researchers are under the impression that the bootstrap does not require large samples to be valid. While there is simulation evidence that shows performance of the bootstrap is certainly better than the robust variance estimator (e.g., Cameron et al.,

2008), the theoretical validity of the bootstrap still rests on large sample (i.e., asymptotic) arguments. Nevertheless, in applied settings similar to what we encountered in the `cluster_trial` data (specifically, when there are fewer then 50 clusters), my preference would be to use the clustered bootstrap.

Similar to the robust variance estimator, there are also many different variations of the bootstrap variance estimator. Among the most important versions of these is the bias-corrected bootstrap, and the bias-corrected and accelerated bootstrap (Davison and Hinkley, 1997). These two variations have been shown to perform better than the normal interval boostrap (or the percentile bootstrap) in a range of settings. Unfortunately, these are much more complicated to code by hand. Thus, for the time being, I almost always rely on the normal-interval bootstrap when dealing with clustered data.

2.6 Summary of Results So Far

```
res <- data.frame(
  Version = c("Uncorrected", "Cluster Robust", "Cluster Bootstrap"),
  Estimate = c(coeftest(mod1)[2,1],
               coeftest(mod1)[2,1],
               coeftest(mod1)[2,1]),
  Std.Err = c(coeftest(mod1)[2,2],
              coeftest(mod1, vcov=vcovCL(mod1,type = "HC0",
                                         cadjust = F,
                                         cluster = cluster_trial$practice))[2,2],
              sd(boot_res[,2])),
  LCL = c(coefci(mod1, level = 0.95)[2,1],
          coefci(mod1, vcov=vcovCL(mod1,type = "HC0",
                                   cadjust = F,
                                   cluster = cluster_trial$practice), level = 0.95)[2,1],
          LCL),
  UCL = c(coefci(mod1, level = 0.95)[2,2],
          coefci(mod1, vcov=vcovCL(mod1,type = "HC0",
                                   cadjust = F,
                                   cluster = cluster_trial$practice), level = 0.95)[2,2],
```

```
          UCL)
)


knitr::kable(res, digits = 2)
```

Version	Estimate	Std.Err	LCL	UCL
Uncorrected	0.42	1.90	-3.31	4.15
Cluster Robust	0.42	2.47	-4.43	5.27
Cluster Bootstrap	0.42	2.68	-4.83	5.67

References

Various Versatile Variances: An Object-Oriented Implementation of Clustered Covariances in R. CRAN.

A. Colin Cameron, Jonah B. Gelbach, and Douglas L. Miller. Bootstrap-based improvements for inference with clustered errors. *The Review of Economics and Statistics*, 90(3):414–427, 2008.

Michael J Campbell. *Statistics at square two: understanding modern statistical applications in medicine*. Blackwell, 2006.

AC Davison and DV Hinkley. *Bootstrap methods and their application*. Cambridge University Press, Cambridge, NY, 1997.

Garrett M. Fitzmaurice, Nan M. Laird, and James H. Ware. *Applied longitudinal analysis*. Wiley series in probability and statistics. Wiley-Interscience, Hoboken, N.J., 2004.

Mohammad Ali Mansournia, Maryam Nazemipour, Ashley I Naimi, Gary S Collins, and Michael J Campbell. Demystifying robust standard errors for epidemiologists. *International Journal of Epidemiology*, Under Review, 2020.

Yudi. Pawitan. *In all likelihood : statistical modelling and inference using likelihood*. Clarendon Press ; Oxford University Press, Oxford; New York, 2001.