Assignment 1

Yue Han

2/15/2021

PART 1

Exercise 1 Missing data

Number of students

```
data$datstu %>% nrow()
```

```
## [1] 340823
```

Number of schools

1. the number of schools in school dataset

```
data$datsss %>% select(schoolcode) %>% distinct() %>% nrow()
```

```
## [1] 898
```

2. the number of schools in student dataset (schools that students applied for)

```
choices_school <- data$datstu %>%
  select(X,schoolcode1:schoolcode6,rankplace, score,jssdistrict) %>%
  pivot_longer(schoolcode1:schoolcode6, names_to = "choice", values_to = "school") %>%
  mutate(choice = str_extract(choice,"[0-9]"))
choices_school %>% select(school) %>% distinct() %>% drop_na() %>% nrow()
```

```
## [1] 640
```

Number of programs

```
choices_program <- data$datstu %>%
  select(X,choicepgm1:choicepgm6) %>%
  pivot_longer(choicepgm1:choicepgm6, names_to = "choice", values_to = "program") %>%
  mutate(choice = str_extract(choice,"[0-9]"))
choices_program %>% select(program) %>% distinct() %>% filter(program != "") %>% nrow()
```

```
## [1] 32
```

Number of choices (school,program)

```
choices <- merge(choices_program, choices_school, by=c("X", "choice"))
choices %>% select(program,school) %>% distinct() %>% drop_na() %>% filter(program != "") %>% nrow()
```

```
## [1] 2773
```

I exclude three types of invalid records:

1. program is NA

```
choices %>% select(program,school) %>% distinct() %>% filter(program == "") %>% nrow()
```

```
## [1] 308
```

2. school is NA

```
choices %>% select(program,school) %>% distinct() %>% filter(is.na(school)) %>% nrow()
```

```
## [1] 6
```

3. both program and school is NA

```
choices %>% select(program,school) %>% distinct() %>% filter(program == "", is.na(school)) %>% nrow()
```

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

Missing test score

```
data$datstu %>% filter(is.na(score)) %>% nrow()
```

```
## [1] 179887
```

Apply to the same school (different programs)

```
choices_school %>%
  filter(!is.na(school)) %>%
  group_by(X) %>%
  dplyr::summarise(n=n(),n_d=n_distinct(school),.groups="drop") %>%
  filter(n!=n_d) %>%
  nrow()
```

```
## [1] 120071
```

Apply to less than 6 choices

```
choices %>%
  mutate(invalid_choice = case_when(
    is.na(school) ~ 1,
    program == "" ~ 1,
    TRUE ~ 0
  )) %>%
  filter(invalid_choice == 1) %>%
  distinct(X) %>%
  nrow()
```

```
## [1] 21001
```

Exercise 2 Data

```
schools <- data$datsss %>%
  select(-c(X,schoolname)) %>%
  distinct() %>%
  drop_na()

admitted <- choices %>%
  drop_na() %>%
  filter(rankplace == choice) %>%
  merge(schools,by.x="school",by.y="schoolcode", all.x=TRUE) %>%
  select(X,school, program, rankplace,score, sssdistrict, ssslong, ssslat, jssdistrict)

admitted_summary <- admitted %>%
  select(school, program, sssdistrict, ssslong, ssslat, score) %>%
```

```
  group_by(school, program, sssdistrict, ssslong, ssslat) %>%
  dplyr::summarise(cutoff = min(score),
            quality = mean(score),
            size = n(),.groups="drop")
```

```
admitted_summary %>% head(20)
```

```
## # A tibble: 20 x 8
##    school program         sssdistrict        ssslong ssslat cutoff quality  size
##     <int> <chr>           <chr>                <dbl>  <dbl>  <int>   <dbl> <int>
##  1  10101 Agriculture     Accra Metropolitan  -0.197   5.61    288    310.    49
##  2  10101 Business        Accra Metropolitan  -0.197   5.61    305    325.   100
##  3  10101 General Arts    Accra Metropolitan  -0.197   5.61    316    330.   100
##  4  10101 General Science Accra Metropolitan  -0.197   5.61    299    329.    50
##  5  10101 Home Economics  Accra Metropolitan  -0.197   5.61    284    301.    49
##  6  10101 Visual Arts     Accra Metropolitan  -0.197   5.61    296    312.    50
##  7  10102 General Arts    Accra Metropolitan  -0.197   5.61    388    405.    88
##  8  10102 General Science Accra Metropolitan  -0.197   5.61    389    406.    70
##  9  10102 Home Economics  Accra Metropolitan  -0.197   5.61    363    377.    45
## 10  10102 Visual Arts     Accra Metropolitan  -0.197   5.61    343    371.    45
## 11  10103 Agriculture     Accra Metropolitan  -0.197   5.61    316    333.    38
## 12  10103 Business        Accra Metropolitan  -0.197   5.61    341    358.   119
## 13  10103 General Arts    Accra Metropolitan  -0.197   5.61    349    363.   117
## 14  10103 General Science Accra Metropolitan  -0.197   5.61    335    354.    80
## 15  10103 Home Economics  Accra Metropolitan  -0.197   5.61    320    336.    49
## 16  10103 Visual Arts     Accra Metropolitan  -0.197   5.61    343    358.    40
## 17  10104 General Arts    Accra Metropolitan  -0.197   5.61    302    320.    55
## 18  10104 General Science Accra Metropolitan  -0.197   5.61    245    283.    55
## 19  10104 Home Economics  Accra Metropolitan  -0.197   5.61    264    286.    55
## 20  10104 Visual Arts     Accra Metropolitan  -0.197   5.61    273    298.    55
```

Exercise 3 Distance

```
dis <- function(ssslong,ssslat,jsslong,jsslat){
  d <- sqrt((69.172*(ssslong-jsslong)*cos(jsslat/57.3))^2 + (69.172*(ssslat-jsslat))^2)
  return(d)
}
```

```
school_distance <- admitted %>%
  merge(select(data$datjss,-X), by="jssdistrict", all.x = TRUE) %>%
  mutate(distance = dis(ssslong, ssslat, point_x, point_y))
```

```
school_distance %>% select(X,jssdistrict,sssdistrict,distance) %>% head(20)
```

```
##         X                          jssdistrict
## 1  207962 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 2  182053 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 3  258932 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 4  182726 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 5  218509 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 6  220174 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 7  207814 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 8  181868 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 9  220123 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 10 181590 Abura/Asebu/Kwamankese (Abura Dunkwa)
```

```
## 11 181972 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 12 181382 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 13 258995 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 14 181867 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 15 259082 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 16 181589 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 17 181388 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 18 181433 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 19 283485 Abura/Asebu/Kwamankese (Abura Dunkwa)
## 20 181390 Abura/Asebu/Kwamankese (Abura Dunkwa)
##                               sssdistrict  distance
## 1                Assin North (Assin Fosu) 46.461827
## 2   Abura/Asebu/Kwamankese (Abura Dunkwa)  0.000000
## 3                   Mfantsiman (Saltpond) 14.034819
## 4                    Cape Coast Municipal  7.719788
## 5    Shama/Ahanta/East (Sekondi/Takoradi) 29.582286
## 6  Asikuma/Odoben/Brakwa (Breman Asikuma) 37.280041
## 7                Assin North (Assin Fosu) 46.461827
## 8   Abura/Asebu/Kwamankese (Abura Dunkwa)  0.000000
## 9  Asikuma/Odoben/Brakwa (Breman Asikuma) 37.280041
## 10  Abura/Asebu/Kwamankese (Abura Dunkwa)  0.000000
## 11  Abura/Asebu/Kwamankese (Abura Dunkwa)  0.000000
## 12  Abura/Asebu/Kwamankese (Abura Dunkwa)  0.000000
## 13                  Mfantsiman (Saltpond) 14.034819
## 14  Abura/Asebu/Kwamankese (Abura Dunkwa)  0.000000
## 15                  Mfantsiman (Saltpond) 14.034819
## 16  Abura/Asebu/Kwamankese (Abura Dunkwa)  0.000000
## 17  Abura/Asebu/Kwamankese (Abura Dunkwa)  0.000000
## 18  Abura/Asebu/Kwamankese (Abura Dunkwa)  0.000000
## 19               Assin North (Assin Fosu) 46.461827
## 20  Abura/Asebu/Kwamankese (Abura Dunkwa)  0.000000
```

Exercise 4 Descriptive Characteristics

```
ad_summary <- function(n,q = 0) {
  ad <- school_distance  %>%
    filter(rankplace == n)
  if (q!=0) {
    q_sc <- quantile(ad$score)
    ad <- ad %>% filter(score>=q_sc[q],score<=q_sc[q+1])
  }
   ad %>%
    group_by(school, program) %>%
    dplyr:: summarise(cutoff = min(score),
                      quality = mean(score),
                      distance = mean(distance),
                      .groups="drop") %>%
     dplyr:: summarise(cutoff_mean = min(cutoff),
                      cutoff_sd = sd(cutoff),
                      quality_mean = mean(quality),
                      quality_sd = sd(quality),
                      distance_mean = mean(distance,na.rm=TRUE),
                      distance_sd = sd(distance,na.rm=TRUE))
}
```

row number represents the rank of choice

```
map_df(1:6, ad_summary)
```

```
## # A tibble: 6 x 6
##   cutoff_mean cutoff_sd quality_mean quality_sd distance_mean distance_sd
##         <int>     <dbl>        <dbl>      <dbl>         <dbl>       <dbl>
## 1         165      47.4         289.       44.5          33.1        36.0
## 2         173      47.5         284.       45.8          33.2        32.3
## 3         190      46.0         281.       43.1          31.7        32.8
## 4         185      43.7         276.       39.8          28.9        33.5
## 5         198      21.8         247.       22.3          32.4        26.5
## 6         158      23.5         249.       22.4          32.5        24.3
```

row number represents the rank of choice

the first table represents the first quartile test score, and so on

```
map(1:4, function(y) map_df(1:6, function(x) ad_summary(n=x,q=y)))
```

```
## [[1]]
## # A tibble: 6 x 6
##   cutoff_mean cutoff_sd quality_mean quality_sd distance_mean distance_sd
##         <int>     <dbl>        <dbl>      <dbl>         <dbl>       <dbl>
## 1         165     17.9          246.      12.5           30.1        41.1
## 2         173     16.0          241.      12.0           29.0        34.8
## 3         190     14.4          237.      10.1           26.4        32.9
## 4         185     12.3          232.       8.29          26.2        35.6
## 5         198      7.22         221.       5.45          30.6        28.0
## 6         158      8.59         220.       6.26          31.4        24.0
##
## [[2]]
## # A tibble: 6 x 6
##   cutoff_mean cutoff_sd quality_mean quality_sd distance_mean distance_sd
##         <int>     <dbl>        <dbl>      <dbl>         <dbl>       <dbl>
## 1         269      9.80         286.       7.91          32.0        41.9
## 2         263     10.1          280.       8.47          34.9        46.5
## 3         255      8.31         269.       6.71          31.8        37.7
## 4         247      7.29         260.       5.83          25.0        32.0
## 5         230      4.41         237.       3.92          29.7        26.2
## 6         231      4.08         238.       3.51          31.5        35.4
##
## [[3]]
## # A tibble: 6 x 6
##   cutoff_mean cutoff_sd quality_mean quality_sd distance_mean distance_sd
##         <int>     <dbl>        <dbl>      <dbl>         <dbl>       <dbl>
## 1         305     12.9          325.      11.7           37.0        46.0
## 2         298     11.2          316.      10.2           38.0        41.2
## 3         284      9.67         300.       8.55          34.4        41.3
## 4         273      8.46         287.       7.18          28.6        39.9
## 5         246      6.23         256.       5.25          32.9        26.5
## 6         245      6.14         255.       5.29          31.1        31.5
##
## [[4]]
## # A tibble: 6 x 6
##   cutoff_mean cutoff_sd quality_mean quality_sd distance_mean distance_sd
```

```
##         <int>     <dbl>        <dbl>      <dbl>         <dbl>       <dbl>
## 1         356      13.1         375.       14.7          42.0        48.1
## 2         341      15.6         362.       16.6          45.6        49.5
## 3         321      17.2         343.       17.2          36.2        40.3
## 4         304      18.3         327.       17.8          29.9        37.5
## 5         269      17.5         291.       17.5          35.3        31.0
## 6         266      18.4         290.       17.7          34.4        25.0
```

PART 2

Exercise 5 Data creation

```
set.seed(111)
n <- 10000
x1 <- runif(n,1,3)
x2 <- rgamma(n,shape=3,scale=2)
x3 <- rbernoulli(n,0.3)*1
e <- rnorm(n,2,1)
y <- 0.5+ 1.2*x1 - 0.9*x2 +0.1*x3 + e
y_bar <- mean(y)
ydum <- as.integer(y > y_bar)
```

Exercise 6 OLS

```
cor(y,x1)
```

```
## [1] 0.2121754
```

the correlation between y and x1 is significantly different from 1.2

```
x <- rbind(rep(1,n),x1,x2,x3)
b <- solve(x%*%t(x))%*%x%*%y
rownames(b)[1] <- 'intercept'
colnames(b) <- 'est_beta'
```

I calculate the sd of coefficients based on both true sigma and estimated sigma of error term.

```
sig2 <- 1
sig2_hat <- sum((y - t(x)%*%b)^2)/(n-4)
b_sd <- as.data.frame(sqrt(diag(solve(x%*%t(x))*sig2))) # use true sig2
rownames(b_sd)[1] <- 'intercept'
colnames(b_sd) <- 'beta_sd_true_sig2'
b_sd_ <-  as.data.frame(sqrt(diag(solve(x%*%t(x))*sig2_hat))) # use estimate sig2
rownames(b_sd_)[1] <- 'intercept'
colnames(b_sd_) <- 'beta_sd_est_sig2'
```

```
cbind(b,b_sd,b_sd_)
```

```
##              est_beta beta_sd_true_sig2 beta_sd_est_sig2
## intercept  2.49459072       0.040446639      0.040805043
## x1         1.21280452       0.017260165      0.017413110
## x2        -0.90006970       0.002886612      0.002912191
## x3         0.07894914       0.021766832      0.021959712
```

Exercise 7 Discrete choice

Since the probability must between 0 and 1, the linear probability model's likelihood function is not continuous (set $p = x\beta$, when $x\beta < 0$, let $p = 0$, when $x\beta > 1$, let $p = 1$). As a result, we cannot calculate the hessian matrix for linear probability model.

```
logit <- function(x,b) {
  xb <- t(x) %*% b
  return(exp(xb)/(1+exp(xb)))
}
probit <- function(x,b) {
  xb <- t(x) %*% b
  return(pnorm(xb))
}
linear <- function(x,b) {
  return(t(x) %*% b)
}
log_lik <- function(f,y,x,b) {
  p <- f(x,b)
  p[p < 1e-8] <- 1e-8 # prevent log(0) situation
  p[p > 1-1e-8] <- 1-1e-8 # prevent log(0) situation
  likelihood <- (p^y)*((1-p)^(1-y))
  return(sum(log(likelihood)))
}
```

if we ignore the fact that $p = x\beta$ must between 0 and 1, then y is not a Bernoulli distribution but a normal distribution given ϵ is a normal distribution. we can just use ols to estimate the parameters and parameters' standard deviation

```
set.seed(111)
start <- rnorm(4)
# logit
fn1 <- function(b){
  return(-log_lik(f=logit,y=ydum,x=x,b))
}
lg <- optim(par=start,fn=fn1,method="BFGS",hessian = TRUE)
# probit
fn2 <- function(b){
  return(-log_lik(f=probit,y=ydum,x=x,b))
}
pb <- optim(par=start,fn=fn2,method="BFGS",hessian = TRUE)
# linear prob
fn3 <- function(b){
  return(-log_lik(f=linear,y=ydum,x=x,b))
}
lp <- optim(par=start,fn=fn3,method="BFGS",hessian = TRUE)
# linear
ln <- NULL
ln$par <- solve(x%*%t(x))%*%x%*%ydum
```

```
# show results
binary_reg_beta <- cbind(pb$par,lg$par,lp$par, ln$par)
rownames(binary_reg_beta) <- c("intercept","x1","x2","x3")
colnames(binary_reg_beta) <- c("probit","logit","linear prob", "linear")
binary_reg_beta
```

```
##                probit      logit linear prob      linear
## intercept  2.80528539  5.0564868   0.2352207  0.88845551
## x1         1.18070600  2.1087610  -0.3307359  0.14575671
## x2        -0.86674262 -1.5566615  -0.3116238 -0.10403279
## x3         0.09306401  0.1797667  -2.3023457  0.01458232
```

above table shows the estimated coefficient of logit, probit and linear model

```
lg$sd <- sqrt(diag(solve(lg$hessian)))
lg_coef <- cbind(lg$par,lg$sd)
rownames(lg_coef) <- c("intercept","x1","x2","x3")
colnames(lg_coef) <- c("coef","sd")

pb$sd <- sqrt(diag(solve(pb$hessian)))
pb_coef <- cbind(pb$par,pb$sd)
rownames(pb_coef) <- c("intercept","x1","x2","x3")
colnames(pb_coef) <- c("coef","sd")

sig2_hat_ <- sum((ydum - t(x)%*%ln$par)^2)/(n-4)
ln$sd <- as.data.frame(sqrt(diag(solve(x%*%t(x))*sig2_hat_)))
ln_coef <-  cbind(ln$par,ln$sd)
rownames(ln_coef)[1] <- 'intercept'
colnames(ln_coef) <- c("coef","sd")
```

```
lg_coef
```

```
##                coef         sd
## intercept  5.0564868 0.17955319
## x1         2.1087610 0.07931358
## x2        -1.5566615 0.03492548
## x3         0.1797667 0.08342599
```

we can see that all of the coefficients are significantly different from 0. for intercept, x1 and x2, the significant level is 1%, while for x3, the significant level is 5%. this means that under 5% significant level, we can say that intercept, x1 and x3 have positive impact on the probability of y = 1, while x2 has negative impact on the probability of y = 1

```
pb_coef
```

```
##                coef         sd
## intercept  2.80528539 0.09576049
## x1         1.18070600 0.04291246
## x2        -0.86674262 0.01749796
## x3         0.09306401 0.04637470
```

```
ln_coef
```

```
##                coef           sd
## intercept  0.88845551 0.0133746794
## x1         0.14575671 0.0057074996
## x2        -0.10403279 0.0009545296
## x3         0.01458232 0.0071977404
```

the probit model and linear model shows similar results

Exercise 8 Marginal Effects

```
logit_gradient <- function(x,b) {
  xb <- t(x) %*% b
  return(exp(xb)/(1+exp(xb))^2)
}
probit_gradient <- function(x,b) {
  xb <- t(x) %*% b
  return(dnorm(xb))
}

lg_par <- lg$par[-1]
names(lg_par) <- c("x1","x2","x3")
lg_me <- map_dbl(map(lg_par,function(a) logit_gradient(x,lg$par)*a), mean)

pb_par <- pb$par[-1]
names(pb_par) <- c("x1","x2","x3")
pb_me <- map_dbl(map(pb_par,function(a) probit_gradient(x,pb$par)*a), mean)
```

```
lg_me
```

```
##          x1          x2          x3
##  0.14400985 -0.10630630  0.01227649
```

```
pb_me
```

```
##          x1          x2          x3
##  0.14522737 -0.10660973  0.01144691
```

we can see that the marginal effect of logit and probit model are almost the same

```
bootstrap <- function(f,g,iter) {
  sample_me <- NULL
  for (i in 1:iter) {
    set.seed(i)
    start <- runif(4,0,1)
    sample_idx <- sample.int(n,size=n,replace=TRUE)
    sample <- x[,sample_idx]
    sample_fn <- function(b){
      return(-log_lik(f=f,y=ydum[sample_idx],x=sample,b))
    }
    sample_model <- optim(par=start,fn=sample_fn,method="BFGS")
    sample_model_par <- sample_model$par[-1]
    names(sample_model_par) <- c("x1","x2","x3")
    sample_me <- rbind(map_dbl(map(sample_model_par,function(a) g(x,sample_model$par)*a), mean), sample_
  }
  return(apply(sample_me,2,sd))
}
```

```
logit_me_sd <- bootstrap(logit,logit_gradient,100)
logit_me_sd
```

```
##           x1           x2           x3
## 0.0044993604 0.0004751811 0.0057577149
```

```
probit_me_sd <- bootstrap(probit,probit_gradient,100)
probit_me_sd
```

```
##          x1          x2          x3
## 0.08185605 0.02948421 0.01443960
```

using bootstrap, we can calculate the standard deviation of marginal effect of logit and probit models. it seems that standard deviation of marginal effect of two models are approximately the same.