Stat 344 Group Project

Samuel Leung: 6559 2651 Cuong Andrew Luu: 16723686 Xuan Chen: 15734643
Eleanor Yi: 91187922 Yi Chen Kevin Ding: 87327789

2023-11-10

```
data <- read.csv("data/cardio_data_processed.csv")
head(data)
```

```
##   id   age gender height weight ap_hi ap_lo cholesterol gluc smoke alco active
## 1  0 18393      2    168     62   110    80           1    1     0    0      1
## 2  1 20228      1    156     85   140    90           3    1     0    0      1
## 3  2 18857      1    165     64   130    70           3    1     0    0      0
## 4  3 17623      2    169     82   150   100           1    1     0    0      1
## 5  4 17474      1    156     56   100    60           1    1     0    0      0
## 6  8 21914      1    151     67   120    80           2    2     0    0      0
##   cardio age_years      bmi          bp_category  bp_category_encoded
## 1      0        50 21.96712 Hypertension Stage 1 Hypertension Stage 1
## 2      1        55 34.92768 Hypertension Stage 2 Hypertension Stage 2
## 3      1        51 23.50781 Hypertension Stage 1 Hypertension Stage 1
## 4      1        48 28.71048 Hypertension Stage 2 Hypertension Stage 2
## 5      0        47 23.01118               Normal               Normal
## 6      0        60 29.38468 Hypertension Stage 1 Hypertension Stage 1
```

```
nrow(data)
```

```
## [1] 68205
```

Goal

- To estimate the total number of people with cardiovascular disease: $\hat{t}_{\text{cardio}}$
- To estimate the average blood pressure of the population: $\bar{u}_{\text{ap\_hi}}$, $\bar{u}_{\text{ap\_lo}}$

Sample Size Calculations:

We can't find the maximum variance from a continuous variable, we go through sample size calculations with the maximum variance possible from a binary variable: the total number of people with cardiovascular disease. We want the margin of error δ of our estimate to be within $\pm$ 1000 people, 19 times out of 20. Given the size of our population is $\sim$ 68000, we believe that an estimate within this margin of error range is reasonable.

Mathematical steps

Let X be the binary random variable that models the number of people with cardiovascular disease, with mean p and variance $p(1-p)$. Let $\bar{y}_S = \hat{p}$ be the estimate of the average number people with cardiovascular disease. We want to estimate the total number of people with cardiovascular disease, $\hat{t}_{p,\text{cardio}} = N\bar{y}_s$. The variance of this estimator will be $var(\hat{t}_{p,\text{cardio}}) = N^2\hat{p}(1-\hat{p})$. Going through our sample size calculations, we want to maximize variance, so we take $\hat{p} = 0.5$. Therefore, ignoring the finite population correction factor for now, we want to solve the following:

$$\begin{aligned} 1000 &\geq 1.96\sqrt{0.25N^2/n_0} \\ \implies \left(\frac{1000}{1.96}\right)^2 &\geq \frac{N^2}{4n_0} \\ \implies n_0 &\geq \left(\frac{N}{2} \cdot \frac{1.96}{1000}\right)^2 \end{aligned}$$

Converting this to code:

```
# add reasoning for why we chose MOE to be 1000

N <- nrow(data)
target <- 1000
n_0 <- (1.96 * N / (2*target))^2
print(paste0("sample size ignoring FPC: ", n_0))
```

```
## [1] "sample size ignoring FPC: 4467.70591281"
```

```
print(paste0("n_0 / N: ", n_0 / N))
```

```
## [1] "n_0 / N: 0.065504082"
```

Comparing the sample size calculated above with the total population size, we see that $\frac{n_0}{N} = 0.066 \geq 0.05$. So we cannot ignore the finite population correction factor in our sample size calculations. Factoring in the FPC, we would want to solve the following:

$$n = \frac{n_0}{1 + n_0/N}$$

Doing the calculations again with the FPC in mind with code:

```
n <- ceiling(n_0 / (1 + n_0 / N))
print(paste0("Sample Size with FPC: ", n))
```

```
## [1] "Sample Size with FPC: 4194"
```

So we use a sample size of 4194.

```r
# set seed for reproduceable results
set.seed(124)

# generate random sample
srs <- data[sample(1:nrow(data), n), ]

# double check size of sample
dim(srs)
```

```
## [1] 4194   17
```

```r
fpc <- 1 - (n / N)
target_lst <- c("ap_hi", "ap_lo", "cardio")

ap_hi.est <- round(mean(srs$ap_hi), 3)
ap_hi.var <- fpc*var(srs$ap_hi) / n
ap_hi.se <- sqrt(ap_hi.var)
ap_hi.moe <- round(1.96*sqrt(ap_hi.var), 3)

ap_lo.est <- round(mean(srs$ap_lo), 3)
ap_lo.var <- fpc*var(srs$ap_lo) / n
ap_lo.se <- sqrt(ap_lo.var)
ap_lo.moe <- round(1.96*sqrt(ap_lo.var), 3)

cardio.p <- mean(srs$cardio)
cardio.est <- round(N*cardio.p, 3)
cardio.var <- fpc*(cardio.p)*(1-cardio.p) / n
cardio.se <- N * sqrt(cardio.var)
cardio.moe <- round(1.96*sqrt(cardio.var), 3)

vanilla.summary <- round(rbind(
  c(ap_hi.est, ap_hi.se, ap_hi.est - ap_hi.moe, ap_hi.est + ap_hi.moe),
  c(ap_lo.est, ap_lo.se, ap_lo.est - ap_lo.moe, ap_lo.est + ap_lo.moe),
  c(cardio.est, cardio.se, cardio.est - cardio.moe, cardio.est + cardio.moe)
), 3)

vanilla.summary <- cbind(paste("vanilla.", target_lst, sep=""), vanilla.summary)
colnames(vanilla.summary) <- c("method", "est", "se", "lower", "upper")

vanilla.summary
```

```
##      method           est         se        lower       upper
## [1,] "vanilla.ap_hi"  "126.314"   "0.237"   "125.85"    "126.778"
## [2,] "vanilla.ap_lo"  "81.373"    "0.136"   "81.107"    "81.639"
## [3,] "vanilla.cardio" "33630.887" "510.093" "33630.872" "33630.902"
```

Regression and Ratio Estimates

To calculate regression and ratio estimates, we want to use an auxiliary variable with strong correlation. We do so by first calculating the correlation values for each of our target variables against the other numeric columns in our data set:

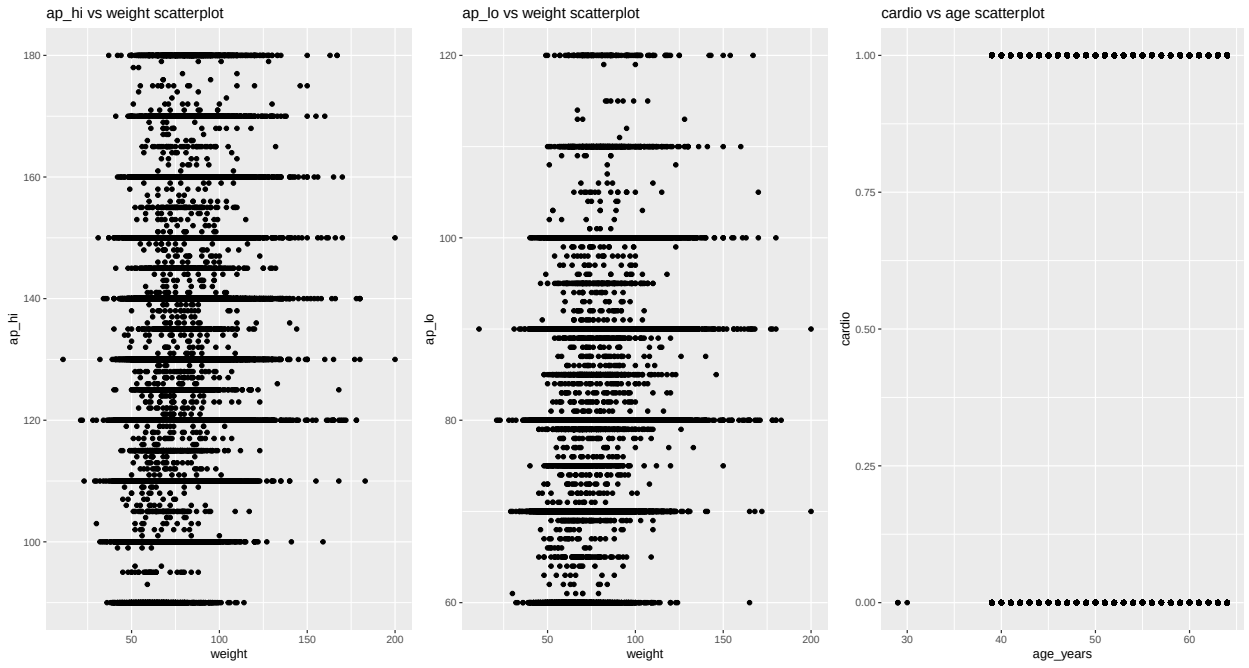
```
pop.cor <- cor(data[,2:15])
pop.cor[target_lst, !(colnames(pop.cor)%in%target_lst)]
```

```
##              age      gender      height    weight cholesterol       gluc
## ap_hi   0.2116064 0.060721542  0.01854418 0.2682888   0.1953296 0.09315051
## ap_lo   0.1559992 0.066125957  0.03554988 0.2501841   0.1616365 0.07331896
## cardio 0.2390321 0.006097527 -0.01127589 0.1778293   0.2207782 0.08890543
##              smoke         alco       active age_years       bmi
## ap_hi   0.02603188  0.032535902 -0.001409008 0.2113136 0.2301632
## ap_lo   0.02383578  0.036211509 -0.001233588 0.1557770 0.2069436
## cardio -0.01656671 -0.009037713 -0.037943619 0.2388754 0.1628798
```

The highest positive-correlating variable for `ap_hi` and `ap_lo` are `weight`. The highest positively-correlating variable for `cardio` is `age`, however it is difficult to interpret the value in this column, as the age is not counted by year. So we opt to use the `age_years` column, which has similar correlation values to `age`. We note that for all these correlations, none of them seem particularly strong, which suggests that the ratio and regression estimates may not be that useful in prediction. To verify their usefulness, we make scatter plots for each target variable against the proposed auxiliary variables above:

```
aphi_vs_weight <- ggplot(data = data, aes(x=weight, y=ap_hi)) +
  geom_point() +
  labs(title = "ap_hi vs weight scatterplot")
aplo_vs_weight <- ggplot(data = data, aes(x=weight, y=ap_lo)) +
  geom_point() +
  labs(title = "ap_lo vs weight scatterplot")
cardio_vs_age <- ggplot(data = data, aes(x=age_years, y=cardio)) +
  geom_point() +
  labs(title = "cardio vs age scatterplot")

plot_grid(aphi_vs_weight, aplo_vs_weight, cardio_vs_age, ncol = 3)
```



For the scatter plots, we can visually confirm that there is little to no linear correlation for all plots. In the case of the `cardio` vs `age_years` plot, an estimate based on linear regression does not seem appropriate given the `cardio` variable is binary.

```
# ratio estimate:
ratio.ap_hi.ratio <- mean(srs$ap_hi) / mean(srs$weight)
ratio.ap_lo.ratio <- mean(srs$ap_lo) / mean(srs$weight)
ratio.cardio.ratio <- mean(srs$cardio) / mean(srs$age_years)

ratio.ap_hi.se <- sqrt(fpc*sum(var(srs$ap_hi-ratio.ap_hi.ratio*srs$weight))/n)
ratio.ap_lo.se <- sqrt(fpc*sum(var(srs$ap_lo-ratio.ap_lo.ratio*srs$weight))/n)
ratio.cardio.se <- N*sqrt(fpc*sum(var(srs$cardio-ratio.cardio.ratio*srs$age_years))/n)

ratio.ap_hi.est <- ratio.ap_hi.ratio * mean(data$weight)
ratio.ap_lo.est <- ratio.ap_lo.ratio * mean(data$weight)
ratio.cardio.est <- ratio.cardio.ratio * mean(data$age_years) * N

ratio.summary <- round(rbind(
  c(ratio.ap_hi.est, ratio.ap_hi.se,
    ratio.ap_hi.est - 1.96 * ratio.ap_hi.se,
    ratio.ap_hi.est + 1.96 * ratio.ap_hi.se),
  c(ratio.ap_lo.est, ratio.ap_lo.se,
    ratio.ap_lo.est - 1.96 * ratio.ap_lo.se,
    ratio.ap_lo.est - 1.96 * ratio.ap_lo.se),
  c(ratio.cardio.est, ratio.cardio.se,
    ratio.cardio.est - 1.96 * ratio.cardio.se,
    ratio.cardio.est + 1.96 * ratio.cardio.se)
), 3)
ratio.summary <- cbind(paste("ratio.", target_lst, sep=""), ratio.summary)
colnames(ratio.summary) <- c("method", "est", "se", "lower", "upper")
# rownames(ratio.summary) <- paste("ratio.", target_lst, sep="")
ratio.summary
```

```
##      method         est         se        lower       upper
## [1,] "ratio.ap_hi"  "126.56"    "0.381"   "125.814"   "127.306"
## [2,] "ratio.ap_lo"  "81.531"    "0.24"    "81.061"    "81.061"
## [3,] "ratio.cardio" "33582.429" "499.536" "32603.338" "34561.52"
```

```
# regression estimate:

# helper function for regression estimate. X, y should be matrices of proper size. Don't ask me why i'm
regression.est_and_se <- function(X, y, x.pop) {
  X <- as.matrix(X); y <- as.matrix(y)
  X <- cbind(rep(1, dim(X)[1]), X)
  w <- solve(t(X)%*%X)%*%t(X)%*%y
  y.hat <- X%*%w
  se <- sqrt(fpc*sum(var(y - y.hat))/nrow(X))
  est <- cbind(1, x.pop) %*% w

  ret <- c()
  ret$est <- est; ret$se <- se
  ret$lower <- est - 1.96*se
  ret$upper <- est + 1.96*se
```

```
  return(ret)
}

reg.ap_hi <- regression.est_and_se(srs$weight, srs$ap_hi, mean(data$weight))
reg.ap_lo <- regression.est_and_se(srs$weight, srs$ap_lo, mean(data$weight))
reg.cardio <- regression.est_and_se(srs$age_years, srs$cardio, mean(data$age_years))

# reg.cardio estimates population mean, so we need to multiply estimate and SE by N
reg.cardio$est <- N*reg.cardio$est
reg.cardio$se <- N*reg.cardio$se

reg.summary <- rbind(
  reg.ap_hi, reg.ap_lo, reg.cardio
)

reg.summary <- cbind(rownames(reg.summary), reg.summary)
colnames(reg.summary)[1] <- "method"
rownames(reg.summary) <- NULL
reg.summary
```

```
##      method       est      se        lower     upper
## [1,] "reg.ap_hi"  126.353  0.2296917 125.9028  126.8032
## [2,] "reg.ap_lo"  81.39437 0.1321236 81.13541  81.65333
## [3,] "reg.cardio" 33543.01 496.8257  0.4775197 0.5060741
```

Comparing the performance of each estimator in one table:

```
rbind(vanilla.summary, ratio.summary, reg.summary)
```

```
##       method           est         se        lower       upper
##  [1,] "vanilla.ap_hi"  "126.314"   "0.237"   "125.85"    "126.778"
##  [2,] "vanilla.ap_lo"  "81.373"    "0.136"   "81.107"    "81.639"
##  [3,] "vanilla.cardio" "33630.887" "510.093" "33630.872" "33630.902"
##  [4,] "ratio.ap_hi"    "126.56"    "0.381"   "125.814"   "127.306"
##  [5,] "ratio.ap_lo"    "81.531"    "0.24"    "81.061"    "81.061"
##  [6,] "ratio.cardio"   "33582.429" "499.536" "32603.338" "34561.52"
##  [7,] "reg.ap_hi"      126.353     0.2296917 125.9028    126.8032
##  [8,] "reg.ap_lo"      81.39437    0.1321236 81.13541    81.65333
##  [9,] "reg.cardio"     33543.01    496.8257  0.4775197   0.5060741
```

We observe that the estimated values for each target variable across all estimating methods are roughly the same. When comparing the standard errors of each, we see that ratio estimator has the largest SE out of all three methods. This is expected as from the correlation table and scatter plots above, there is little to no positive correlation between the target and response variable, making the ratio estimator a poor choice. The standard errors for `ap_hi` and `ap_lo` are similar for vanilla and regression estimates, with the vanilla estimate having slightly lower standard errors. Given the low correlation values shown above, the regression line formed will probably be similar to the constant prediction line that the vanilla estimate has, resulting in similar estimates and standard error. However, for the `cardio` response variable, given that the response is binary, linear regression would not be a suitable method to model the data, which is why we see such a higher standard error using the regression method when compared to the vanilla estimate.

Stratification

- We assume that we know nothing about our population, aside from population size
- We also know strata sizes, N_h/N
-

We want to stratify our population based on the variable that will create the greatest distinction between stratas. From our intuition, there are a few candidate variables that fulfill this requirement, such as `gender`, `smoke`, and `alco`. Taking sample allocation into account, we would also want to choose a stratifying variable which can help us assume equal variances of our target variables post sampling and stratification, as we don't have a best guess for the strata variances. Out of the 3 candidate stratifiers listed above, our intuition suggests that `gender` would have the least effect on cardiovascular disease presence, while retaining distinct separation between stratas, so we choose `gender` to stratify on. Assuming equal variances between these stratas, we sample using proportional allocation:

$$\frac{n_h}{n} = \frac{N_h}{N}$$

Calculating the stratified estimate and standard error under proportional allocation:

```
# generate stratified sample
set.seed(124)
n.gender <- round(prop.table(table(data$gender)) * n, 0)
pop.gender1 <- data[data$gender==1,]; pop.gender2 <- data[data$gender==2,]
strata.sample <- rbind(pop.gender1[sample(1:nrow(pop.gender1), n.gender[1]),],
                       pop.gender2[sample(1:nrow(pop.gender2), n.gender[2]), ])


# given population data, sample and calculate the stratified estimate + SE
stratify.est_and_se <- function(pop, sample, var_str, target_str, total=FALSE) {
  pop.var.count <- table(data[,var_str])
  sample.var.count <- table(sample[,var_str])
  # population proportion for each strata
  pop.prop <- prop.table(pop.var.count)

  sample.means <- tapply(sample[,target_str], sample[,var_str], mean)
  sample.vars <- tapply(sample[,target_str], sample[,var_str], var) / sample.var.count
  fpc <- 1 - sample.var.count / pop.var.count

  est <- round(sum(pop.prop * sample.means), 3)
  se <- round(sqrt(sum(pop.prop^2 * fpc * sample.vars)), 3)
  if (total==TRUE) {est <- N*est; se <- N*se}
  # print(paste0("Strata Estimate for ", target_str, ": ", est, " +/- ", 1.96*se))
  ret <- c(est = est, se = se, lower = est - 1.96*se, upper = est + 1.96*se)
  # ret$est <- est
  # ret$se <- se
  return(ret)
}

strat.ap_hi <- stratify.est_and_se(data, strata.sample,
                                   var_str = "gender", target_str = "ap_hi")
strat.ap_lo <- stratify.est_and_se(data, strata.sample,
                                   var_str = "gender", target_str = "ap_lo")
strat.cardio <- stratify.est_and_se(data, strata.sample, var_str = "gender",
```

```
                    target_str = "cardio", total=TRUE)
strat.summary <- rbind(strat.ap_hi, strat.ap_lo, strat.cardio)
strat.summary <- cbind(rownames(strat.summary), strat.summary)
colnames(strat.summary)[1] <- "method"
rownames(strat.summary) <- NULL
strat.summary
```

```
##      method         est         se        lower        upper
## [1,] "strat.ap_hi"  "126.276"   "0.236"   "125.81344"  "126.73856"
## [2,] "strat.ap_lo"  "81.169"    "0.136"   "80.90244"   "81.43556"
## [3,] "strat.cardio" "33761.475" "477.435" "32825.7024" "34697.2476"
```

Comparing all estimates and standard errors from all methods of estimation:

```
rbind(vanilla.summary, ratio.summary, reg.summary, strat.summary)
```

```
##       method           est         se        lower        upper
##  [1,] "vanilla.ap_hi"  "126.314"   "0.237"   "125.85"     "126.778"
##  [2,] "vanilla.ap_lo"  "81.373"    "0.136"   "81.107"     "81.639"
##  [3,] "vanilla.cardio" "33630.887" "510.093" "33630.872"  "33630.902"
##  [4,] "ratio.ap_hi"    "126.56"    "0.381"   "125.814"    "127.306"
##  [5,] "ratio.ap_lo"    "81.531"    "0.24"    "81.061"     "81.061"
##  [6,] "ratio.cardio"   "33582.429" "499.536" "32603.338"  "34561.52"
##  [7,] "reg.ap_hi"      126.353     0.2296917 125.9028     126.8032
##  [8,] "reg.ap_lo"      81.39437    0.1321236 81.13541     81.65333
##  [9,] "reg.cardio"     33543.01    496.8257  0.4775197    0.5060741
## [10,] "strat.ap_hi"    "126.276"   "0.236"   "125.81344"  "126.73856"
## [11,] "strat.ap_lo"    "81.169"    "0.136"   "80.90244"   "81.43556"
## [12,] "strat.cardio"   "33761.475" "477.435" "32825.7024" "34697.2476"
```

Stratified estimate SE now the lowest.