Michael Ellsworth - Assignment 1

30101253

Further explore the dataset “api”

The dataset “api” in package “survey” contains the academic performance index for year 1999 (api99) and 2000 (api00). To import the data, you need to attach the package “survey”.

Now treat the new constructed dataset “MYDATA” as the “population” source. **The variable** we are interested is **“api00”**.

Goal: **estimate the population average as well as the standard deviation of the estimate**.

Population level

Question 1

Calculate the population mean

```
pop_mean <- mean(MYDATA$api00)
pop_mean
```

```
## [1] 664.6981
```

Equal-size cluster sampling

Question 2

Apply one-stage cluster sampling to estimate the population mean (**n=3**), calculate its estimated standard deviation (refer to the formula in lecture notes), **using L1∼L10 as clusters**.

```
# Define number of clusters sampled, number of clusters total and size of clusters
n_cluster <- 3
N_cluster <- 10
M_cluster <- nrow(MYDATA)/N_cluster

# Create a vector of cluster names to sample from
clusters <- paste0("L", seq(1, 10))

# Sample from the cluster name vectors n times
cluster_sample <- sample(clusters, n_cluster)

# Filter the MYDATA table to include only the clusters from cluster_sample
MYDATA_cluster_sample <- MYDATA %>%
  filter(cluster_name %in% cluster_sample)

# Calculate the totals from each of the clusters
results <- MYDATA_cluster_sample %>%
  group_by(cluster_name) %>%
  summarise(totals = sum(api00), count = n())

# Estimate the population mean
mean(MYDATA_cluster_sample$api00)
```

```
## [1] 667.5934
```

```
# Estimate the standard deviation
((1 - n_cluster / N_cluster) * var(results$totals) / (n_cluster * M_cluster**2))**0.5
```

```
## [1] 5.962857
```

Question 3

Please repeat Q2 10 times which gives you at least 10 population mean estimates (**e.g. using "for" loop**), compute the average of these 10 estimates as well as their standard deviation.

```
# Initialize a vector of population mean estimates
pop_mean_estimates_cluster <- c()

# Create a for loop to append pop_mean_estimates
for(i in 1:10){
  # Sample from the cluster name vectors n times
  cluster_sample <- sample(clusters, n_cluster)

  # Filter the MYDATA table to include only the clusters from cluster_sample
  MYDATA_cluster_sample <- MYDATA %>%
    filter(cluster_name %in% cluster_sample)

  # Estimate the population mean
  pop_mean_estimates_cluster[i] <- mean(MYDATA_cluster_sample$api00)
}
# Ten estimates of the population mean
pop_mean_estimates_cluster
```

```
##  [1] 683.9192 679.1955 673.2940 668.3156 680.2079 672.7146 636.4949
##  [8] 680.3506 668.0539 678.6791
```

```
# Calculate the mean of the pop_mean_estimates_cluster
mean_pop_estimates_cluster <- mean(pop_mean_estimates_cluster)
mean_pop_estimates_cluster
```

```
## [1] 672.1225
```

```
# Calculate the standard deviation of the pop_mean_estimates_cluster
sd_pop_estimates_cluster <- sd(pop_mean_estimates_cluster)
sd_pop_estimates_cluster
```

```
## [1] 13.61106
```

Question 4

Take an SRS (simple random sample) of the same sample size (3*619) to estimate the population mean, calculate its estimated standard deviation (refer to the formula in lecture notes).

```
# Define the secondary sampling unit sample size
n_SRS <- 619 * 3
N_SRS <- 619 * 10

# Filter the MYDATA table to include only the clusters from cluster_sample
MYDATA_SRS <- sample_n(MYDATA, n_SRS)

# Estimate the population mean
mean(MYDATA_SRS$api00)
```

```
## [1] 665.8002
```

```
# Estimate the standard deviation
sd(MYDATA_SRS$api00)
```

```
## [1] 127.4208
```

Question 5

Repeat Q4 for at least 10 times which gives you at least 10 population mean estimates (**e.g. using "for" loop**), compute the average of these 10 estimates as well as their standard deviation.

```
# Initialize a vector of population mean estimates
pop_mean_estimates_SRS <- c()

# Create a for loop to append pop_mean_estimates
for(i in 1:10){
  # Filter the MYDATA table to include only the clusters from cluster_sample
  MYDATA_SRS <- sample_n(MYDATA, n_SRS)

  # Estimate the population mean
  pop_mean_estimates_SRS[i] <- mean(MYDATA_SRS$api00)
}
# Ten population estimates
pop_mean_estimates_SRS
```

```
##  [1] 667.3581 669.3543 665.1912 668.1788 662.7954 666.8288 662.9230
##  [8] 660.4254 665.3414 667.2639
```

```
# Calculate the mean of the pop_mean_estimates_SRS
mean_pop_estimates_SRS <- mean(pop_mean_estimates_SRS)
mean_pop_estimates_SRS
```

```
## [1] 665.566
```

```
# Calculate the standard deviation of the pop_mean_estimates_SRS
sd_pop_estimates_SRS <- sd(pop_mean_estimates_SRS)
sd_pop_estimates_SRS
```

```
## [1] 2.791763
```

Question 6

Compare your estimates (cluster-sampling and SRS) with the population mean, **use ANOVA table (calculate SSB and SSW)** to explain why your cluster-sampling-based estimate is good or bad.

```
# Find population means in each cluster
SSB_data <- MYDATA %>%
  group_by(cluster_name) %>%
  summarise(mean = mean(api00)) %>%
  mutate(square_diff = (mean - pop_mean)^2)

# Sum the squares (SSB) between psus
SSB <- sum(SSB_data$square_diff) * 619
SSB
```

```
## [1] 7237647
```

```
MSB <- SSB / (N_cluster - 1)

# Create a data set with cluster means and differences squared
SSW_data <- MYDATA %>%
  group_by(cluster_name) %>%
  mutate(cluster_mean = mean(api00)) %>%
  ungroup() %>%
  mutate(square_diff = (api00 - cluster_mean)^2)

# Sum the squares (SSW) within psus
SSW <- sum(SSW_data$square_diff)
SSW
```

```
## [1] 94595893
```

```
MSW <- SSW / (N_cluster * (M_cluster - 1))

# Total
SSTO <- SSW + SSB
S_squared <- SSTO / (N_cluster * M_cluster - 1)

# Anova table for cluster sampling
summary(aov(api00 ~ cluster_name, data = MYDATA))
```

```
##                Df   Sum Sq Mean Sq F value Pr(>F)
## cluster_name    9  7237647  804183   52.54 <2e-16 ***
## Residuals    6180 94595893   15307
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Is MSB greater than S^2?
MSB > S_squared
```

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

Since MSB is greater than S^2 , cluster sampling is more volatile than simple random sample.

Ratio estimation

Question 7

Use variable **grad.sch** as auxiliary variable, apply the ratio estimation to estimate the population mean (take SRS, get B_hat and then estimate).

```
# Take a simple random sample of MYDATA using sample size of n = 3 * 619
MYDATA_SRS_7 <- sample_n(MYDATA, n_SRS)

# Calculate means of the sample of y (api00) and x (grad.sch)
sample_mean_api00 <- mean(MYDATA_SRS_7$api00)
sample_mean_grad.sch <- mean(MYDATA_SRS_7$grad.sch)

# Calculate the ratio estimate (B_hat)
B_hat <- sample_mean_api00 / sample_mean_grad.sch

# Calculate the ratio estimate of the population mean of y (api00) using the known popul
ation mean of x (grad.sch)
mean_api00_ratio_estimate <- B_hat * mean(MYDATA$grad.sch)
mean_api00_ratio_estimate
```

```
## [1] 650.9081
```

Question 8

Repeat the above procedure for at least 10 times (**e.g. using "for" loop**), get at least 10 estimates, compute their average as well as standard deviation.

```
mean_api00_ratio_estimate_vector_8 <- c()

for(i in 1:10){
  # Take a simple random sample of MYDATA using sample size of n = 3 * 619
  MYDATA_SRS_8 <- sample_n(MYDATA, n_SRS)

  # Calculate means of the sample of y (api00) and x (grad.sch)
  sample_mean_api00 <- mean(MYDATA_SRS_8$api00)
  sample_mean_grad.sch <- mean(MYDATA_SRS_8$grad.sch)

  # Calculate the ratio estimate (B_hat)
  B_hat <- sample_mean_api00 / sample_mean_grad.sch

  # Calculate the ratio estimates of the population mean of y (api00) using the known po
pulation mean of x (grad.sch)
  mean_api00_ratio_estimate_vector_8[i] <- B_hat * mean(MYDATA$grad.sch)
}
# Ten estimates of the population mean
mean_api00_ratio_estimate_vector_8
```

```
##  [1] 653.6863 690.2807 659.5344 703.6170 648.2762 651.1128 677.8918
##  [8] 660.9949 689.5044 643.6001
```

```
# Mean of the ten estimates of the population mean
mean(mean_api00_ratio_estimate_vector_8)
```

```
## [1] 667.8499
```

```
# Standard deviation of the ten estimates of the population mean
sd(mean_api00_ratio_estimate_vector_8)
```

```
## [1] 20.87063
```

Question 9

Use variable **api99** as auxiliary variable, apply the ratio estimation to estimate the population mean (take SRS, get B_hat and then estimate).

```
# Take a simple random sample of MYDATA using sample size of n = 3 * 619
MYDATA_SRS_9 <- sample_n(MYDATA, n_SRS)

# Calculate means of the sample of y (api00) and x (api99)
sample_mean_api00 <- mean(MYDATA_SRS_9$api00)
sample_mean_api99 <- mean(MYDATA_SRS_9$api99)

# Calculate the ratio estimate (B_hat)
B_hat <- sample_mean_api00 / sample_mean_api99

# Calculate the ratio estimate of the population mean of y (api00) using the known popul
ation mean of x (grad.sch)
mean_api00_ratio_estimate <- B_hat * mean(MYDATA$api99)
mean_api00_ratio_estimate
```

```
## [1] 664.7178
```

Question 10

Repeat the above procedure for at least 10 times (**e.g. using “for” loop**), get at least 10 estimates, compute their average as well as standard deviation.

```
mean_api00_ratio_estimate_vector_10 <- c()

for(i in 1:10){
  # Take a simple random sample of MYDATA using sample size of n = 3 * 619
  MYDATA_SRS_10 <- sample_n(MYDATA, n_SRS)

  # Calculate means of the sample of y (api00) and x (grad.sch)
  sample_mean_api00 <- mean(MYDATA_SRS_10$api00)
  sample_mean_api99 <- mean(MYDATA_SRS_10$api99)

  # Calculate the ratio estimate (B_hat)
  B_hat <- sample_mean_api00 / sample_mean_api99

  # Calculate the ratio estimates of the population mean of y (api00) using the known po
pulation mean of x (api99)
  mean_api00_ratio_estimate_vector_10[i] <- B_hat * mean(MYDATA$api99)
}
# Ten estimates of the population mean
mean_api00_ratio_estimate_vector_10
```

```
##  [1] 664.2974 663.8925 664.7344 665.0150 664.4110 665.6214 665.0126
##  [8] 664.4587 665.4797 663.2884
```

```
# Mean of the ten estimates of the population mean
mean(mean_api00_ratio_estimate_vector_10)
```

```
## [1] 664.6211
```

```
# Standard deviation of the ten estimates of the population mean
sd(mean_api00_ratio_estimate_vector_10)
```

```
## [1] 0.7103513
```

Question 11

Compare the estimation results above, which auxiliary variable gives better estimation? Explain why (do significance test, calculate correlation coefficient).

The ratio estimate of the population mean of api00 is better estimated using the auxiliary variable api00 than grad.sch. This is evident from the standard deviation of each of the population mean estimates calculated in Question 8 and Question 10. The standard deviation of the population mean estimates using grad.sch is ~ `{r} sd(mean_api00_ratio_estimate_vector_8)` which is much greater than the standard deviation of the population mean estimates using api99, which is ~ `{r} sd(mean_api00_ratio_estimate_vector_10)`.

This can also be explained using the correlation coefficient.

```
# Calculate the correlation coefficient of api00 ~ grad.sch
cor(x = MYDATA$grad.sch, y = MYDATA$api00)
```

```
## [1] 0.6680752
```

```
# Calculate the correlation coefficient of api00 ~ api.99
cor(x = MYDATA$api99, y = MYDATA$api00)
```

```
## [1] 0.975085
```

Since the correlation coefficient of `api00 ~ api.99` is closer to 1 than `api00 ~ grad.sch`, the relationship is more linear and therefore a better estimator.

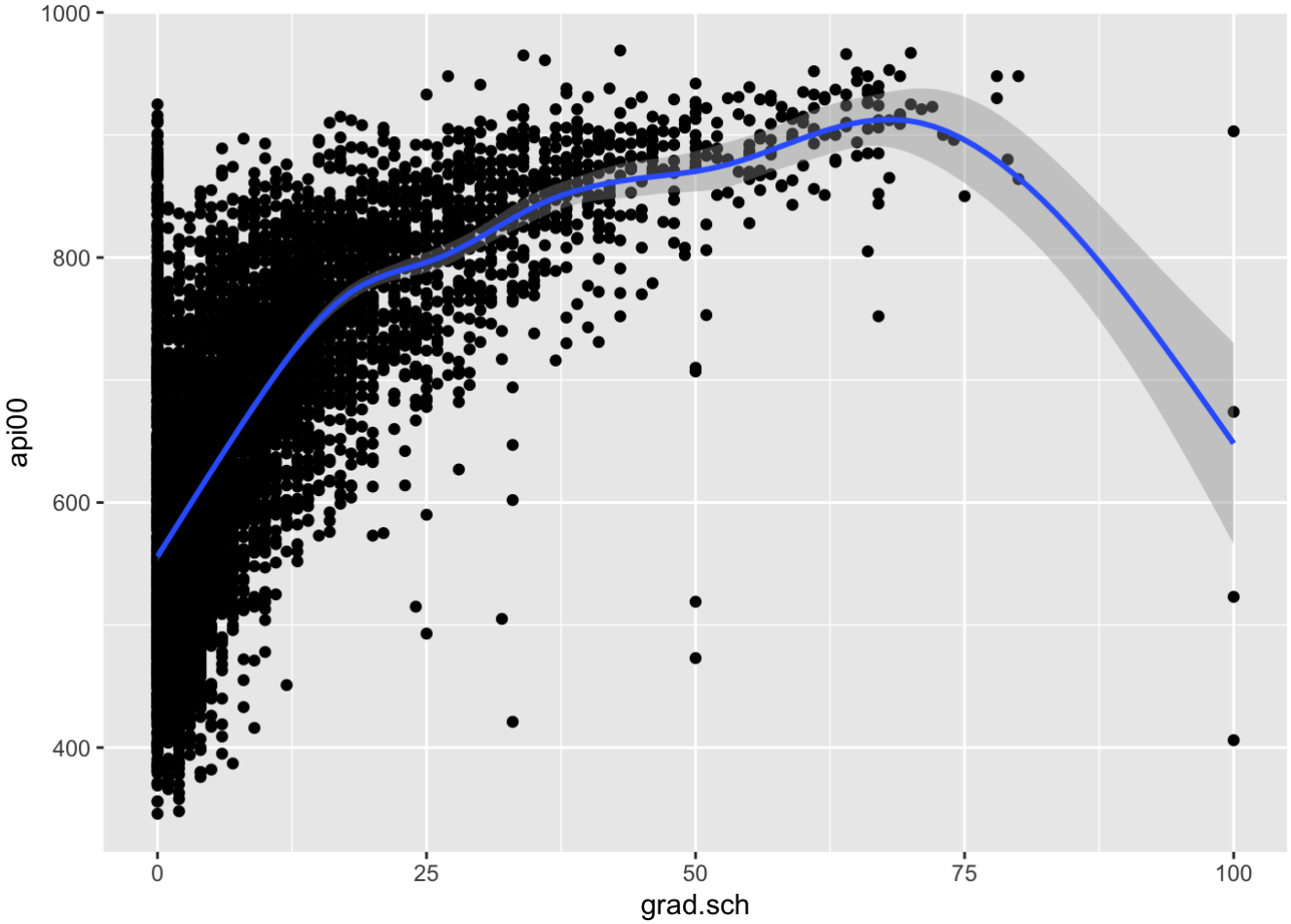
Question 12

Please use function “scatter.smooth” to visualize your explanations in Q11. **hint: help(scatter.smooth)**

The differences in the correlation coefficient between `api00 ~ grad.sch` and `api00 ~ api.99` can also be expressed visually using scatter plots.

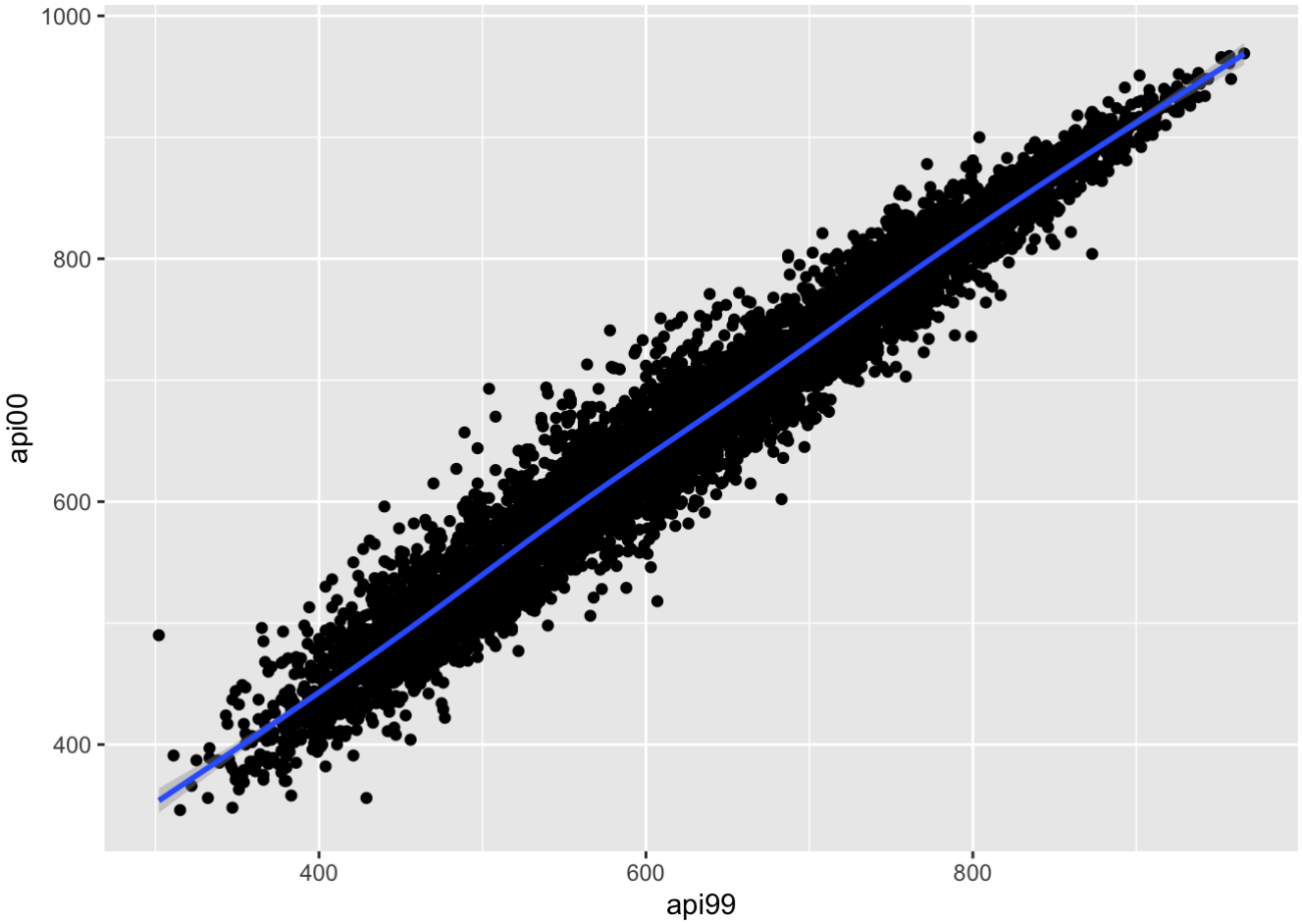
```
# Plot the relationship between grad.sch and api00 to visualize how linear the relations
hip is
MYDATA %>% ggplot(aes(x = grad.sch, y = api00)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
# Plot the relationship between api99 and api00 to visualize how linear the relationship is
MYDATA %>% ggplot(aes(x = api99, y = api00)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



As shown by the above plots, the relationship between `api00 ~ api99` is very linear whereas the relationship between `api00 ~ grad.sch` is less linear.