STAT 344 Group Project (Part I)

Lucas Qin, Frederick Wang, Isabel Wilson

2023-10-31

Reading in Data

```
all_data_files = list.files("data")
all_df = read.csv(paste0("data/", all_data_files[1]))

for (i in 2 : length(all_data_files)) {
  all_df = rbind(all_df, read.csv(paste0("data/", all_data_files[i])))
}

filtered_df = all_df %>% filter(Section == "OVERALL") %>%
  mutate(CourseNum = ifelse(
    is.na(Detail), Course, paste0(Course, Detail)
  )) %>%
  select(-Campus, -Year, -Session, -Section, -Professor, -Course, -Detail)

filtered_df %>% slice(1:10)
```

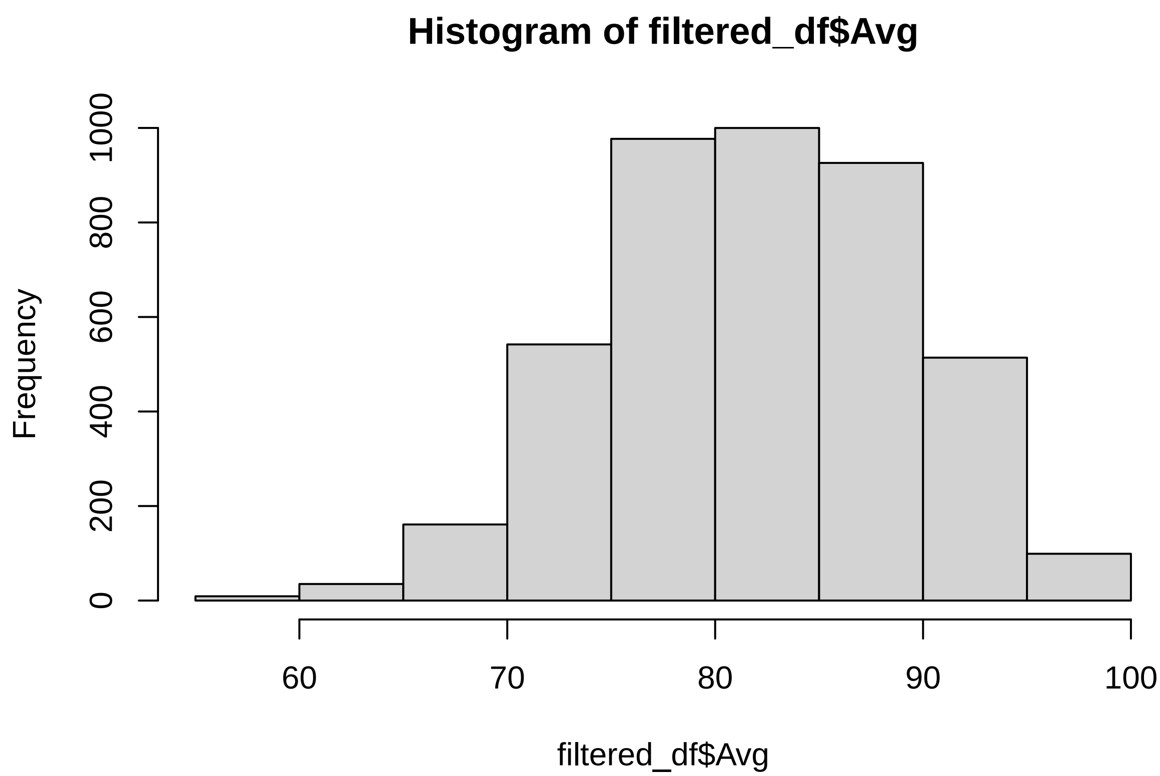
```
##    Subject                                                  Title Enrolled
## 1     AANB                                Topics in Animal Welfare       11
## 2     ACAM                        Asian Canadians in Popular Culture       50
## 3     ACAM                              Dis/Orienting Asian Canada       48
## 4     ACAM                          Selected Topics in ACAM Studies       53
## 5     ACAM                          Selected Topics in ACAM Studies       12
## 6     ACAM                      Asian Canadian Community-Based Media       18
## 7     ACAM Directed Studies in Asian Canadian and Asian Migration        9
## 8     ADHE                                          Teaching Adults      164
## 9     ADHE                          Institutions of Adult Education      142
## 10    ADHE        Developing Short Courses, Workshops and Seminars      166
##         Avg   Std.dev High Low X.50 X50.54 X55.59 X60.63 X64.67 X68.71 X72.75
## 1  92.72727  2.611165   95  90    0      0      0      0      0      0      0
## 2  77.96000 10.604119   93  46    1      2      1      1      2      2      5
## 3  77.60417 17.594414   91   5    3      1      0      0      1      1      3
## 4  80.24528 13.027937   92   8    1      1      1      0      0      4      5
## 5  80.91667  3.553701   86  75    0      0      0      0      0      0      2
## 6  84.94444  3.621378   90  77    0      0      0      0      0      0      0
## 7  94.33333  2.958040   98  90    0      0      0      0      0      0      0
## 8  81.92073 12.560521   97   0    2      2      3      0      4      9     10
## 9  84.06338 11.298161   98   0    2      0      0      7      0      2      9
## 10 83.60241 12.079474   98   0    2      0      0      1      6      5      8
```

```
##    X76.79 X80.84 X85.89 X90.100 CourseNum
## 1       0      0      0      11       550
## 2       3     17     12       4       250
## 3      10      8     16       5       300
## 4       2     14     18       7      320B
## 5       3      5      2       0      320D
## 6       1      6      9       2       350
## 7       0      0      0       9      447C
## 8      12     40     47      35       327
## 9       4     33     35      50       328
## 10     18     34     38      54       329
```

```r
N = nrow(filtered_df)
N
```

```
## [1] 4263
```

```r
histogram = hist(filtered_df$Avg)
```



Since we don't have access to the population standard deviation to guess the standard error of the sample we will be taking, we decided to use our personal transcript standard deviations to estimate the sample standard error. We found our transcript standard deviations to be around 7 percent, so we decided to guess the sample standard error to be 7 as well.

```r
margin_of_error = 1 # desired width is 2%
sample_stdev_guess = 7 # intuitive guess since we don't have previous studies
n1 = (1/(margin_of_error^2/(qnorm(0.975)^2*sample_stdev_guess^2) + 1/N)) %>% ceiling()
n1 # minimum sample size for the mean
```

```
## [1] 181
```

```r
margin_of_error = 0.05 # 2% width
conservative_squared_se = 0.5 * (1 - 0.5)
n2 = (1/(margin_of_error^2/(qnorm(0.975)^2*conservative_squared_se) + 1/N)) %>% ceiling()
n2 # minimum sample size for the proportion
```

```
## [1] 353
```

```r
n = max(n1, n2) # final decided sample size
n
```

```
## [1] 353
```

```r
set.seed(1)
srs = sample(filtered_df$Avg, n)
srs
```

```
##   [1] 74.56250 71.29213 91.22581 82.38462 75.32812 88.62500 90.33333 77.70563
##   [9] 93.63636 74.67647 78.22605 75.90698 82.68966 83.72727 83.57576 83.21176
##  [17] 75.77686 76.36039 86.11111 91.98361 62.62740 86.45455 76.51220 69.29897
##  [25] 84.25000 83.87619 81.00000 80.87143 93.33333 90.90000 75.72131 90.20833
##  [33] 78.67857 72.79200 83.06897 81.03704 75.95000 78.44578 66.02703 80.45238
##  [41] 81.35294 81.75000 85.73620 73.94118 87.06780 85.92683 86.07143 80.51515
##  [49] 91.38462 95.75000 79.85714 75.74747 84.56250 75.88889 91.58333 86.65000
##  [57] 76.47059 92.33333 83.11111 90.69565 71.75000 84.98291 79.82143 87.87500
##  [65] 79.30000 82.53191 82.73256 87.57143 89.92683 80.47619 95.19048 75.70968
##  [73] 88.13228 92.00000 74.70992 71.26263 82.66667 92.50000 77.78571 76.13793
##  [81] 87.58333 79.35435 86.69444 74.53982 93.10000 72.71951 93.66667 90.72727
##  [89] 80.63158 90.87500 69.23427 76.96667 69.82143 87.30000 68.88806 90.55556
##  [97] 80.44928 76.21818 72.16667 85.70588 85.88889 83.40441 93.02128 87.37500
## [105] 88.07317 85.54545 79.70513 81.17290 78.73077 80.45000 71.55556 79.50000
## [113] 77.26667 72.25989 84.60000 60.23077 91.27273 83.31250 87.26316 84.89062
## [121] 83.45238 88.54545 95.61538 88.64474 80.14545 81.80952 73.81690 78.90625
## [129] 74.73585 73.81767 67.89908 92.25000 81.10526 85.45161 88.30000 78.05882
## [137] 74.00000 75.92000 95.50000 94.14286 81.72308 73.29412 85.64286 84.80645
## [145] 88.00000 84.33333 88.55556 80.25000 88.65714 72.52941 69.92453 84.71429
## [153] 87.23077 79.77778 84.55556 89.83333 88.20000 78.57143 86.46667 72.50000
## [161] 82.13043 86.65217 76.41917 78.73077 78.76471 78.38095 82.10169 89.16667
## [169] 71.21429 75.18919 88.64706 72.67188 80.32967 76.21164 84.00000 77.33333
## [177] 90.36735 89.94737 91.66667 84.67188 74.38596 81.08537 78.59633 81.59259
## [185] 79.02386 84.95652 79.47917 87.65049 90.37037 87.00000 76.80000 93.50000
## [193] 84.66667 68.99180 73.65000 78.85106 74.17391 69.49717 69.74555 90.64815
## [201] 89.15385 82.28571 82.15686 81.13136 88.71429 84.28889 76.22222 89.04545
## [209] 71.17391 95.85417 88.80645 96.71429 78.74336 72.97500 92.88889 87.33333
## [217] 79.41463 84.81818 85.87097 76.30078 71.94118 81.04167 85.58621 71.67347
```

```
## [225] 92.20000 74.98361 96.75000 82.19444 85.10000 70.04762 92.64286 70.05714
## [233] 72.53333 81.22526 90.57143 81.58000 75.14815 74.10968 89.37037 87.14286
## [241] 84.40000 70.92500 83.96364 75.02899 89.50000 84.60000 66.71111 90.00000
## [249] 81.44444 89.18349 88.40000 84.57143 75.34545 86.24561 91.27500 80.83721
## [257] 80.72047 70.31959 90.20000 85.41176 72.18182 76.77778 73.60934 81.29249
## [265] 79.63804 80.84615 92.00000 85.37037 83.36559 82.62500 70.70492 81.16667
## [273] 71.16522 85.52632 66.55882 90.71429 93.35714 76.09091 85.66667 96.17647
## [281] 79.58824 88.09677 72.07143 89.00000 81.34783 94.00000 77.15385 85.56250
## [289] 79.92857 86.01852 83.00000 74.04167 72.73050 75.93220 80.81250 92.08696
## [297] 70.98392 88.60000 78.57642 72.31925 79.53333 78.87160 83.54902 73.92308
## [305] 75.66935 75.36538 95.75000 88.02353 86.66667 92.25000 91.00000 78.37500
## [313] 83.00000 67.23684 76.49123 82.36364 87.28571 84.84848 71.00000 77.00388
## [321] 86.85507 87.68182 77.37500 84.41667 84.33333 81.49287 88.93548 90.10000
## [329] 82.16667 58.03279 91.14815 71.92000 82.66667 75.73973 79.37838 77.55422
## [337] 88.00000 75.29787 67.70588 86.53333 94.26667 78.47619 82.44186 80.50000
## [345] 77.81720 76.87912 84.96392 85.30508 85.00000 80.04639 75.21212 94.65000
## [353] 91.93960
```

SRS for mean:

```
sample_mean = mean(srs); sample_mean
```

```
## [1] 81.89962
```

```
sample_se = sqrt((1-n/N)*sd(srs)^2/n); sample_se
```

```
## [1] 0.3723216
```

```
ci_lb = sample_mean - qnorm(0.975)*sample_se
ci_ub = sample_mean + qnorm(0.975)*sample_se

conf_int = c(ci_lb, ci_ub)
c("Confidence Interval for Average UBC Grades Across All Classes in 2021",
  "Winter Using Simple Random Sample",
  conf_int)
```

```
## [1] "Confidence Interval for Average UBC Grades Across All Classes in 2021"
## [2] "Winter Using Simple Random Sample"
## [3] "81.169882922767"
## [4] "82.6293567986551"
```

SRS for proportion of grades above 90%:

```
p_hat = length(srs[srs >= 90]) / n; p_hat
```

```
## [1] 0.1558074
```

```
se_p_hat = sqrt((1 - n / N) * (p_hat * (1 - p_hat)) / n); se_p_hat
```

```
## [1] 0.01848665
```

```r
conf_int_p_hat = p_hat + c(-1, 1) * qnorm(0.975) * se_p_hat
c("Confidence Interval for Proportion of UBC Grades",
  "above 90% Across All Classes in 2021",
  "Winter Using Simple Random Sample",
  conf_int_p_hat)
```

```
## [1] "Confidence Interval for Proportion of UBC Grades"
## [2] "above 90% Across All Classes in 2021"
## [3] "Winter Using Simple Random Sample"
## [4] "0.119574204578259"
## [5] "0.192040526299928"
```

Preprocessing before stratifying:

```r
process_faculty = function(faculty) {
  if (faculty == "Faculty of Arts" |
      faculty == "Faculty of Education") {
    return ("arts")
  } else if (faculty == "Faculty of Science" |
             faculty == "Faculty of Medicine") {
    return ("science")
  } else if (faculty == "Faculty of Applied Science") {
    return ("engineering")
  } else if (faculty == "Faculty of Comm and Bus Admin" |
             faculty == "Vancouver School of Economics") {
    return ("business")
  } else {
    return ("other")
  }
}
```

```r
code2faculty = read.csv("summary.csv") %>%
  mutate(Faculty = Vectorize(process_faculty)(FacultyRaw)) %>%
  select(-Description, -FacultyRaw)
filtered_df = merge(code2faculty, filtered_df, by.x = "Subject")
```

```r
counts = filtered_df %>% group_by(Faculty) %>%
  summarize(counts = length(Faculty)) %>% arrange(desc(counts))
```

Stratified sampling (proportional allocation):

```r
stratas = counts$Faculty %>% as.vector()
num_stratas = length(stratas)
Nh = counts$counts
weights = Nh / N
nh = round(weights * n)

means = rep(0, num_stratas)
sd = rep(0, num_stratas)
props = rep(0, num_stratas)
```

```r
for (i in 1 : num_stratas) {
  subpopulation = filtered_df %>% filter(Faculty == stratas[i])
  sample = sample(subpopulation$Avg, nh[i])
  means[i] = mean(sample)
  sd[i] = sd(sample)
  props[i] = length(sample[sample >= 90]) / nh[i]
}

mean_average_str = sum(weights * means)
prop_str = sum(weights * props)

se = sd / sqrt(nh) * sqrt(1 - nh / Nh)
se_mean_average_str = sqrt(sum(weights^2 * se^2))

se_props_squared = props * (1 - props) / nh * (1 - nh / Nh)
se_prop_str = sqrt(sum(weights^2 * se_props_squared))

ci_mean_str =
  mean_average_str + c(-1, 1) * qnorm(0.975) * se_mean_average_str
ci_prop_str =
  prop_str + c(-1, 1) * qnorm(0.975) * se_prop_str
```