Dan Seol - 260677676 MATH 525 Assignment 2

#3.24 Show that the variance is minimized for a fixed cost with the cost function in (3.12) when $n_h \propto \frac{N_h S_h}{\sqrt{c_h}}$
We want to minimize

$$C = c_0 + \sum_{h=1}^{H} c_h n_h$$

for a fixed variance

$$V(\hat{t}_{str}) = \sum_{h=1}^{H}(1 - \frac{n_h}{N_h})N_h^2 \frac{S_h^2}{n_h}$$

We use Lagrange multipliers.

First, for a given strata, with samples $\vec{n} = (n_1, n_2, ..., n_H)$ construct a function $L(\vec{n})$ such that

$$L(\vec{n}) = c_0 + \sum_{h=1}^{H} c_h n_h - \lambda\{\sum_{h=1}^{H}(1 - \frac{n_h}{N_h})(\frac{N_h}{N})^2(\frac{S_h^2}{n_h})\}$$

such that

$$\nabla L(\vec{n}) = (\frac{\partial L}{\partial n_1}, \frac{\partial L}{\partial n_2}, ..., \frac{\partial L}{\partial n_h}) = \vec{0}$$

Remark $\forall i \in \{1, ..., H\}$

$$\frac{\partial L}{\partial n_i} = c_i - \lambda \frac{\partial}{\partial n_i}\{(1 - \frac{n_i}{N_i})(\frac{N_i}{N})^2 \frac{S_i^2}{n_i}\} = 0$$

$$\implies c_i - \lambda(\frac{N_i}{N})^2 \frac{\partial}{\partial n_i}\{\frac{S_i^2}{n_i} - \frac{S_i^2}{N_i}\} = c_i n_i - \lambda(\frac{N_i}{N})^2 \frac{\partial}{\partial n_i}(\frac{S_i^2}{n_i}) = 0$$

$$\implies c_i + \lambda(\frac{N_i}{N})^2 \frac{S_i^2}{n_i^2} = 0$$

$$\implies n_i^2 c_i = -\lambda \frac{N_i^2}{N^2} S_i^2 = \frac{-\lambda}{N^2} N_i^2 S_i^2 \implies$$

$$n_i^2 = \frac{-\lambda}{N^2}\frac{N_i^2 S_i^2}{c_i} \implies n_i = \sqrt{\frac{-\lambda}{N^2}}\frac{N_i S_i}{\sqrt{c_i}} = r\frac{N_i S_i}{\sqrt{c_i}}$$

where

$$r := \sqrt{\frac{-\lambda}{N^2}}$$

Showing

$\forall h \ n_h \propto \frac{N_i S_i}{\sqrt{c_i}}$

#3.37

```r
#Import necessary packages
library(tidyverse)
```

```
## -- Attaching packages --------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.3.0
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0

## -- Conflicts ---------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(survey)
```

```
## Loading required package: grid

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:tidyr':
##
##     expand

## Loading required package: survival

##
## Attaching package: 'survey'

## The following object is masked from 'package:graphics':
##
##     dotchart
```

```
library(srvyr)
```

```
##
## Attaching package: 'srvyr'

## The following object is masked from 'package:stats':
##
##     filter
```

```
library(knitr)
library(kableExtra)

#Discovered a function: strata?
#svydesign turned out to be a better function since it allows a complex survey design
```

##(a)

Using one or more following variables: $age, sex, race$ or $marstat$, divide the population into strata. Explain how you decided upon your stratification variable and how you chose the number of strata to us. (Note: It is NOT FAIR to use the values of inctot in the population to choose your strata! However, you may draw a pilot sample of size 200 using an SRS to aid you in constructing your strata.

```
ipums <- read_csv('ipums.csv')
```

```
## Parsed with column specification:
## cols(
##   Stratum = col_double(),
##   Psu = col_double(),
##   Inctot = col_double(),
##   Age = col_double(),
```

```
##   Sex = col_double(),
##   Race = col_double(),
##   Hispanic = col_double(),
##   Marstat = col_double(),
##   Ownershg = col_double(),
##   Yrsusa = col_double(),
##   School = col_double(),
##   Educrec = col_double(),
##   Labforce = col_double(),
##   Occ = col_double(),
##   Classwk = col_double(),
##   VetStat = col_double()
## )
```

```r
head(ipums)
```

```
## # A tibble: 6 x 16
##   Stratum   Psu Inctot   Age   Sex  Race Hispanic Marstat Ownershg Yrsusa
##     <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>    <dbl>   <dbl>    <dbl>  <dbl>
## 1       1     1   4105    18     1     2        0       5        0      0
## 2       1     1   7795    20     1     1        0       5        2      0
## 3       1     1  16985    24     1     1        0       1        1      0
## 4       1     1   7045    21     1     1        0       1        2      0
## 5       1     1   2955    23     1     1        0       5        2      0
## 6       1     1      0    17     1     1        0       5        1      0
## # ... with 6 more variables: School <dbl>, Educrec <dbl>, Labforce <dbl>,
## #   Occ <dbl>, Classwk <dbl>, VetStat <dbl>
```

```r
dim(ipums)
```

```
## [1] 53461    16
```

```r
ipums_complete = ipums %>% filter(!is.na(Inctot))
#Verify that none of the data is missing
dim(ipums_complete)
```

```
## [1] 53461    16
```

```r
summary(ipums_complete)
```

```
##     Stratum           Psu            Inctot           Age
##  Min.   :1.000   Min.   : 1.00   Min.   :-9995   Min.   :15.00
##  1st Qu.:2.000   1st Qu.:19.00   1st Qu.: 1505   1st Qu.:25.00
##  Median :6.000   Median :51.00   Median : 6005   Median :37.00
##  Mean   :4.977   Mean   :45.21   Mean   : 9194   Mean   :41.17
##  3rd Qu.:7.000   3rd Qu.:68.00   3rd Qu.:13260   3rd Qu.:56.00
##  Max.   :9.000   Max.   :90.00   Max.   :75000   Max.   :90.00
##       Sex             Race          Hispanic          Marstat
##  Min.   :1.000   Min.   :1.000   Min.   :0.00000   Min.   :1.000
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:0.00000   1st Qu.:1.000
##  Median :2.000   Median :1.000   Median :0.00000   Median :1.000
##  Mean   :1.522   Mean   :1.186   Mean   :0.05425   Mean   :2.402
##  3rd Qu.:2.000   3rd Qu.:1.000   3rd Qu.:0.00000   3rd Qu.:5.000
##  Max.   :2.000   Max.   :5.000   Max.   :1.00000   Max.   :5.000
##     Ownershg         Yrsusa           School         Educrec
##  Min.   :0.000   Min.   :0.0000   Min.   :1.000   Min.   :1.000
##  1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:5.000
```

```
##  Median :1.000   Median :0.0000   Median :1.000   Median :7.000
##  Mean   :1.256   Mean   :0.2548   Mean   :1.148   Mean   :6.343
##  3rd Qu.:2.000   3rd Qu.:0.0000   3rd Qu.:1.000   3rd Qu.:8.000
##  Max.   :2.000   Max.   :5.0000   Max.   :2.000   Max.   :9.000
##     Labforce          Occ           Classwk         VetStat
##  Min.   :0.000   Min.   : 0.00   Min.   : 0.00   Min.   :0.000
##  1st Qu.:1.000   1st Qu.: 0.00   1st Qu.: 0.00   1st Qu.:1.000
##  Median :2.000   Median :18.00   Median :22.00   Median :1.000
##  Mean   :1.588   Mean   :28.77   Mean   :16.34   Mean   :1.132
##  3rd Qu.:2.000   3rd Qu.:48.00   3rd Qu.:22.00   3rd Qu.:1.000
##  Max.   :2.000   Max.   :96.00   Max.   :29.00   Max.   :2.000
```

```
#Pick age, sex, race or Marstat
#It seems that everything besides age is semantically a factor variable
#Make sure R recognizes those columns as factors

lapply(list(ipums_complete$Marstat, ipums_complete$Age, ipums_complete$Race, ipums_complete$Sex), is.fa
```

```
## [[1]]
## [1] FALSE
##
## [[2]]
## [1] FALSE
##
## [[3]]
## [1] FALSE
##
## [[4]]
## [1] FALSE
```

```
ipums_complete$Marstat = as.factor(ipums_complete$Marstat)
ipums_complete$Race = as.factor(ipums_complete$Race)
ipums_complete$Sex = as.factor(ipums_complete$Sex)

#Let's discretize ipums_complete$Age

ipums_complete$Agecat<-cut(ipums_complete$Age, c(0, 20, 40, 60, 80, 100),labels= c("S1","S2","S3","S4",

is.factor(ipums_complete$Agecat)
```

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

```
counts1 = ipums_complete %>% count(Race)%>% mutate(prop=n/sum(n),
                                                   prop_alloc = round(prop*660))
counts2 = ipums_complete %>% count(Sex)%>% mutate(prop=n/sum(n),
                                                  prop_alloc = round(prop*660))
counts3 = ipums_complete %>% count(Marstat)%>% mutate(prop=n/sum(n),
                                                      prop_alloc = round(prop*660))
countsAge = ipums_complete %>% count(Agecat)%>% mutate(prop=n/sum(n),
                                                       prop_alloc = round(prop*660))
```

```
#for part b
counts1
```

```
## # A tibble: 5 x 4
##   Race      n    prop prop_alloc
##   <fct> <int>   <dbl>      <dbl>
```

```
## 1 1      46186 0.864          570
## 2 2       5874 0.110           73
## 3 3        325 0.00608          4
## 4 4        886 0.0166          11
## 5 5        190 0.00355          2
```

```r
counts2
```

```
## # A tibble: 2 x 4
##   Sex       n  prop prop_alloc
##   <fct> <int> <dbl>      <dbl>
## 1 1     25538 0.478        315
## 2 2     27923 0.522        345
```

```r
counts3
```

```
## # A tibble: 5 x 4
##   Marstat     n   prop prop_alloc
##   <fct>   <int>  <dbl>      <dbl>
## 1 1       31160 0.583         385
## 2 2        1186 0.0222         15
## 3 3        3298 0.0617         41
## 4 4        4109 0.0769         51
## 5 5       13708 0.256         169
```

```r
countsAge
```

```
## # A tibble: 5 x 4
##   Agecat     n   prop prop_alloc
##   <fct>  <int>  <dbl>      <dbl>
## 1 S1      7536 0.141          93
## 2 S2     21860 0.409         270
## 3 S3     13950 0.261         172
## 4 S4      8752 0.164         108
## 5 S5      1363 0.0255         17
```

```r
set.seed(329)
srs_200=ipums_complete %>% slice(sample(1:nrow(ipums_complete),size=200,replace=F))
dim(srs_200)
```

```
## [1] 200  17
```

```r
srs_200 %>%
  summarise(SampleMean=mean(Inctot),
                        SampleVar = var(Inctot),
                        SampleSD = sd(Inctot))  %>%
  gather(stat,val) %>%
  kable(.,format="latex",digits=0) %>%
  kable_styling(.)
```

stat	val
SampleMean	8605
SampleVar	88327430
SampleSD	9398

```r
#Pilot SRS of 200
sampleRace = srs_200 %>% count(Race)%>% mutate(prop=n/sum(n),
```

```r
                                            prop_alloc = round(prop*200))
sampleSex = srs_200 %>% count(Sex)%>% mutate(prop=n/sum(n),
                                            prop_alloc = round(prop*200))
sampleMarstat = srs_200 %>% count(Marstat)%>% mutate(prop=n/sum(n),
                                            prop_alloc = round(prop*200))
sampleAge = srs_200 %>% count(Agecat)%>% mutate(prop=n/sum(n),
                                            prop_alloc = round(prop*200))
sampleRace
```

```
## # A tibble: 5 x 4
##   Race      n  prop prop_alloc
##   <fct> <int> <dbl>      <dbl>
## 1 1       176 0.88         176
## 2 2        19 0.095         19
## 3 3         1 0.005          1
## 4 4         3 0.015          3
## 5 5         1 0.005          1
```

```r
sampleSex
```

```
## # A tibble: 2 x 4
##   Sex       n  prop prop_alloc
##   <fct> <int> <dbl>      <dbl>
## 1 1        91 0.455         91
## 2 2       109 0.545        109
```

```r
sampleMarstat
```

```
## # A tibble: 5 x 4
##   Marstat     n  prop prop_alloc
##   <fct>   <int> <dbl>      <dbl>
## 1 1         121 0.605        121
## 2 2           4 0.02           4
## 3 3           9 0.045          9
## 4 4          16 0.08          16
## 5 5          50 0.25          50
```

```r
sampleAge
```

```
## # A tibble: 5 x 4
##   Agecat     n  prop prop_alloc
##   <fct>  <int> <dbl>      <dbl>
## 1 S1        26  0.13         26
## 2 S2        80  0.4          80
## 3 S3        48  0.24         48
## 4 S4        38  0.19         38
## 5 S5         8  0.04          8
```

```r
#Comparing pilot sample vs. sample in part b
sampleRace
```

```
## # A tibble: 5 x 4
##   Race      n  prop prop_alloc
##   <fct> <int> <dbl>      <dbl>
## 1 1       176 0.88         176
## 2 2        19 0.095         19
## 3 3         1 0.005          1
```

```
## 4 4          3 0.015           3
## 5 5          1 0.005           1
```

```
counts1
```

```
## # A tibble: 5 x 4
##   Race      n    prop prop_alloc
##   <fct> <int>   <dbl>      <dbl>
## 1 1     46186 0.864          570
## 2 2      5874 0.110           73
## 3 3       325 0.00608          4
## 4 4       886 0.0166          11
## 5 5       190 0.00355          2
```

```
sampleSex
```

```
## # A tibble: 2 x 4
##   Sex       n  prop prop_alloc
##   <fct> <int> <dbl>      <dbl>
## 1 1        91 0.455         91
## 2 2       109 0.545        109
```

```
counts2
```

```
## # A tibble: 2 x 4
##   Sex       n  prop prop_alloc
##   <fct> <int> <dbl>      <dbl>
## 1 1     25538 0.478        315
## 2 2     27923 0.522        345
```

```
sampleMarstat
```

```
## # A tibble: 5 x 4
##   Marstat     n  prop prop_alloc
##   <fct>   <int> <dbl>      <dbl>
## 1 1         121 0.605        121
## 2 2           4 0.02           4
## 3 3           9 0.045          9
## 4 4          16 0.08          16
## 5 5          50 0.25          50
```

```
counts3
```

```
## # A tibble: 5 x 4
##   Marstat     n   prop prop_alloc
##   <fct>   <int>  <dbl>      <dbl>
## 1 1       31160 0.583         385
## 2 2        1186 0.0222         15
## 3 3        3298 0.0617         41
## 4 4        4109 0.0769         51
## 5 5       13708 0.256         169
```

```
sampleAge
```

```
## # A tibble: 5 x 4
##   Agecat     n  prop prop_alloc
##   <fct>  <int> <dbl>      <dbl>
## 1 S1        26  0.13         26
## 2 S2        80  0.4          80
```

```
## 3 S3        48  0.24         48
## 4 S4        38  0.19         38
## 5 S5         8  0.04          8
```

```r
countsAge
```

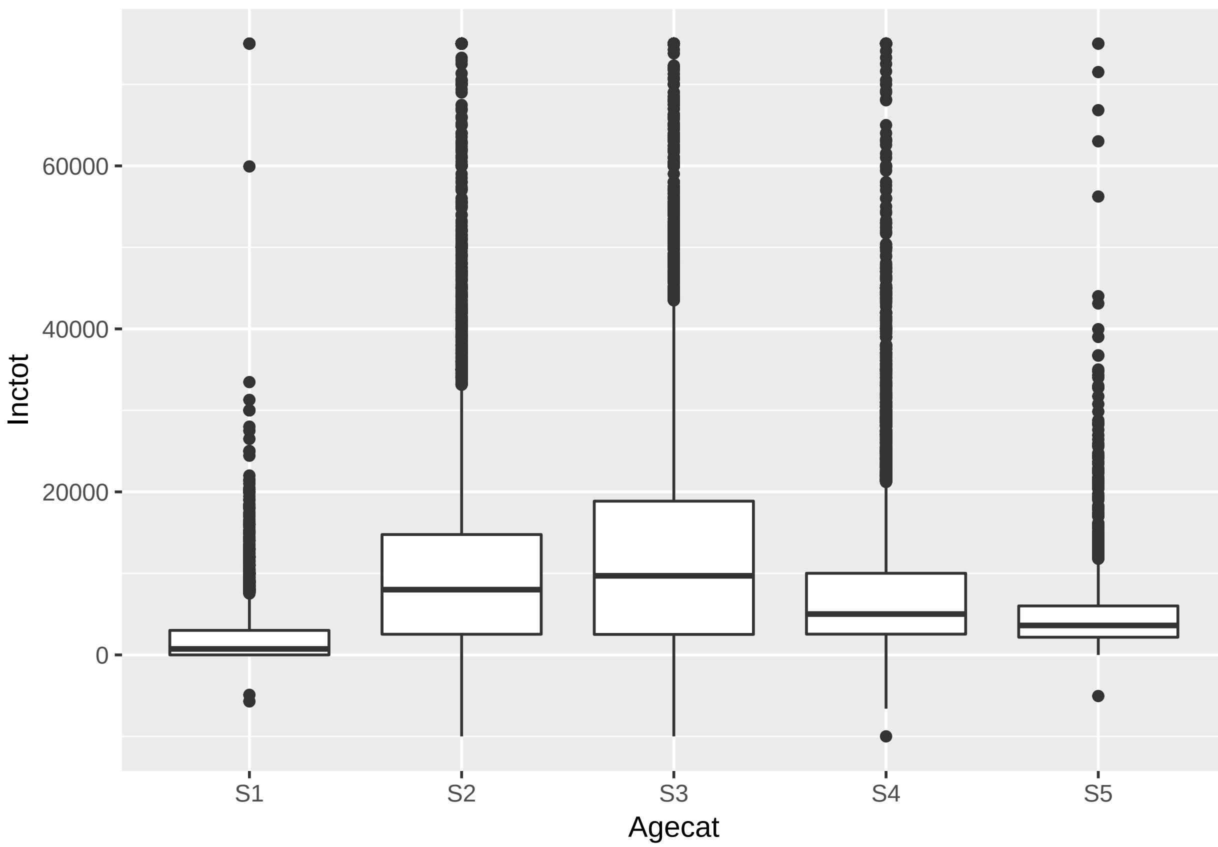
```
## # A tibble: 5 x 4
##   Agecat     n   prop prop_alloc
##   <fct>  <int>  <dbl>      <dbl>
## 1 S1      7536 0.141          93
## 2 S2     21860 0.409         270
## 3 S3     13950 0.261         172
## 4 S4      8752 0.164         108
## 5 S5      1363 0.0255         17
```

It seems that partitioning data by age is the most promising option here, since there is a seemingly valid explanation why age might affect income (The higher position one gets with respect to career, the more income one would earn).

As we have seen from the table output given by the R chunk above,I have constructed five strata, namely

$$S_1 := (0, 20] \quad S_2 := (20, 40] \quad S_3 := (40, 60] \quad S_4 := (60, 80] \quad S_5 := (80.100]$$

```r
ggplot(ipums_complete, aes(Agecat, Inctot))+geom_boxplot()
```



##(b)

Using the strata you constructed, draw a stratified random sample using proportional allocation. Use the same overall sample size you used for your SRS in Exercise 37 of Chapter 2. Explain how you calculated the sample size to be drawn from each stratum.

```r
#We have obtained 660 or 661 as optimal sample size in Assignment 1
set.seed(20171015)
strat_660 =  inner_join(ipums_complete,countsAge,by="Agecat") %>% group_by(Agecat) %>% slice(sample(1:n
```

```
## Warning in 1:n: numerical expression has 7536 elements: only the first used

## Warning in 1:n: numerical expression has 21860 elements: only the first
## used

## Warning in 1:n: numerical expression has 13950 elements: only the first
## used

## Warning in 1:n: numerical expression has 8752 elements: only the first used

## Warning in 1:n: numerical expression has 1363 elements: only the first used
```

```r
dim(strat_660)
```

```
## [1] 660  20
```

```r
ipums.stratdesign = svydesign(~1,strata=~Agecat,data=strat_660,fpc=~n)
```

```r
#We find the sampling proportion

sampleProp <- 660/dim(ipums)[1]
sampleProp
```

```
## [1] 0.01234545
```

Since we do proportional allocation with $\frac{n}{N} = 0.0123 = 1.24\%$, we sample 1.23% of each population stratum.

##(c)

Using the stratified sample you selected with proportional allocation, estimate the total income for the population, along with 95% CI.

```r
svytotal(~Inctot,ipums.stratdesign)
```

```
##            total       SE
## Inctot 484751317 20326751
```

```r
confint(svytotal(~Inctot,ipums.stratdesign))
```

```
##            2.5 %    97.5 %
## Inctot 444911617 524591018
```

##(d)

Using the pilot sample of size 200 to estimate the within-stratum variances, use optimal allocation to determine sample stratum sizes. Use the same value of n as in part 37b, which is the same n from the SRS in Exercise 37 of Chapter 2. Draw a stratified random sample from the population along with a 95% CI.

Since we have no associated cost function with it, we will assume every sample unit has the same cost. Then the optimal allocation becomes Neyman allocation where for strata $h, = 1, ...H$

$$n_h \propto N_h S_h$$

$$n_h = n \frac{N_h S_h}{\sum_{i=1}^{N} N_i S_i}$$

```r
#strat_200
```

```r
S_i = inner_join(srs_200,sampleAge,by="Agecat") %>% group_by(Agecat) %>% summarise(SampleSD = sd(Inctot
S_i
```

```
## # A tibble: 5 x 2
##   Agecat SampleSD
##   <fct>     <dbl>
## 1 S1        3373.
## 2 S2        8101.
## 3 S3       12841.
## 4 S4        7892.
## 5 S5        4556.
```

```r
N_i <- as.numeric(unlist(countsAge[2]))
s_i <- as.numeric(unlist(S_i[2]))
np_i = N_i*s_i/sum(N_i*s_i)

NeymanAge = ipums_complete %>% count(Agecat)%>% mutate(size = N_i, si=s_i, Neyman=np_i, neyman_prop_all
NeymanAge
```

```
## # A tibble: 5 x 6
##   Agecat     n  size     si Neyman neyman_prop_alloc
##   <fct>  <int> <dbl>  <dbl>  <dbl>             <dbl>
## 1 S1      7536  7536  3373. 0.0556                37
## 2 S2     21860 21860  8101. 0.388                256
## 3 S3     13950 13950 12841. 0.392                259
## 4 S4      8752  8752  7892. 0.151                100
## 5 S5      1363  1363  4556. 0.0136                 9
```

```r
set.seed(20171015)
#Due to rounding we are taking one more population unit in our sample
neyman_661 <- inner_join(ipums_complete,NeymanAge,by="Agecat") %>% group_by(Agecat) %>% slice(sample(1:
```

```
## Warning in 1:n: numerical expression has 7536 elements: only the first used

## Warning in 1:n: numerical expression has 21860 elements: only the first
## used

## Warning in 1:n: numerical expression has 13950 elements: only the first
## used

## Warning in 1:n: numerical expression has 8752 elements: only the first used

## Warning in 1:n: numerical expression has 1363 elements: only the first used
```

```r
dim(neyman_661)
```

```
## [1] 661  22
```

```r
ipums.Neymandesign = svydesign(~1,strata=~Agecat,data=neyman_661,fpc=~n)
```

```r
svytotal(~Inctot,ipums.Neymandesign)
```

```
##            total       SE
## Inctot 493391266 19465182
```

```r
confint(svytotal(~Inctot,ipums.Neymandesign))
```

```
##            2.5 %    97.5 %
## Inctot 455240209 531542322
```

##(e) Under what conditions can optimal allocation be expected to perform much better than proportional allocation?

Do these conditions exist for this population? Comment on the relative performance you observed between these two allocations.

Optimal allocation (in our case it would be Neyman allocation) is guaranteed to perform better when all the standard deviations are known.

Yes, the conditions do exist since we can find population stratum-within variances:

```
S_perf <- inner_join(srs_200,sampleAge,by="Agecat") %>% group_by(Agecat) %>% summarise(PopulationSD = sc
S_perf
```

```
## # A tibble: 5 x 2
##   Agecat PopulationSD
##   <fct>         <dbl>
## 1 S1            3373.
## 2 S2            8101.
## 3 S3           12841.
## 4 S4            7892.
## 5 S5            4556.
```

```
set.seed(20171015)
s_perf <- as.numeric(unlist(S_perf[2]))
np_perf = N_i*s_perf/sum(N_i*s_i)

NeymanAge_known = ipums_complete %>% count(Agecat)%>% mutate(size = N_i, si=s_perf, Neyman=np_perf, neym
NeymanAge_known
```

```
## # A tibble: 5 x 6
##   Agecat     n  size     si Neyman neyman_prop_alloc
##   <fct>  <int> <dbl>  <dbl>  <dbl>             <dbl>
## 1 S1      7536  7536  3373. 0.0556                37
## 2 S2     21860 21860  8101. 0.388                256
## 3 S3     13950 13950 12841. 0.392                259
## 4 S4      8752  8752  7892. 0.151                100
## 5 S5      1363  1363  4556. 0.0136                 9
```

```
neyman_perf <- inner_join(ipums_complete,NeymanAge_known,by="Agecat") %>% group_by(Agecat) %>% slice(sam
```

```
## Warning in 1:n: numerical expression has 7536 elements: only the first used

## Warning in 1:n: numerical expression has 21860 elements: only the first
## used

## Warning in 1:n: numerical expression has 13950 elements: only the first
## used

## Warning in 1:n: numerical expression has 8752 elements: only the first used

## Warning in 1:n: numerical expression has 1363 elements: only the first used
```

```
dim(neyman_perf)
```

```
## [1] 661  22
```

```
ipums.Neymanperf = svydesign(~1,strata=~Agecat,data=neyman_perf,fpc=~n)
```

```
svytotal(~Inctot,ipums.Neymanperf)
```

```
##             total       SE
## Inctot 493391266 19465182
```

```r
confint(svytotal(~Inctot,ipums.Neymanperf))
```

```
##             2.5 %    97.5 %
## Inctot 455240209 531542322
```

Returning identical results with our estimated standard deviations with Neyman allocations, and doing signifacantly better than proportional allocations

$$SE(\text{Neyman}) = 19465182 << 20326751 = SE(\text{proportional})$$

##(f)

Overall, do you think your stratification was worthwhile for sampling from this population? How did your stratified estimates compare with the estimate from the SRS you took in Chapter 2? If you were to start over on the stratification, what would you do differently?

Let us compare the results:

For SRS,

```r
set.seed(20171015)
srs_661 = ipums_complete %>% slice(sample(1:nrow(ipums_complete),
                                          size=661, replace=F))
srs_design = survey::svydesign(id=~1,data=srs_661, fpc=rep(53461,661))

svytotal(~Inctot,srs_design)
```

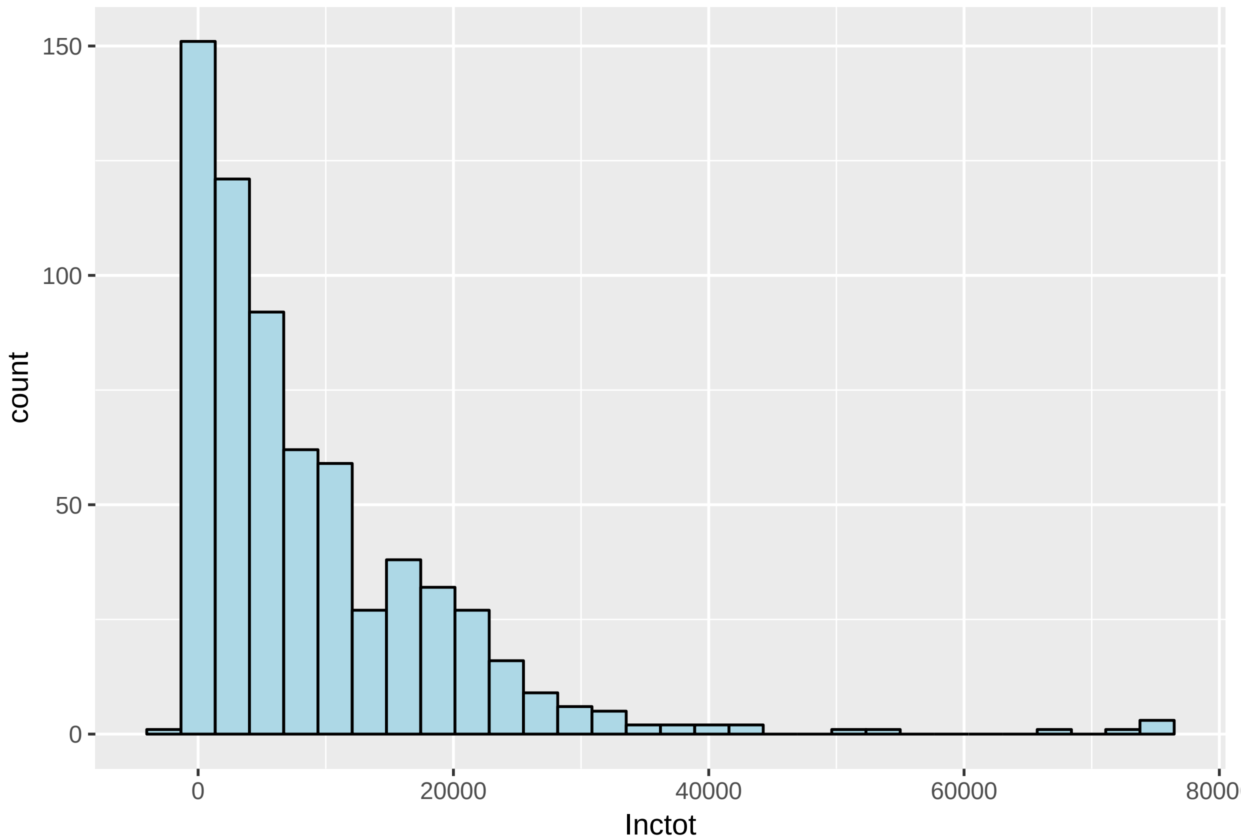
```
##            total       SE
## Inctot 4.75e+08 21475362
```

```r
confint(svytotal(~Inctot, srs_design))
```

```
##             2.5 %    97.5 %
## Inctot 432906005 517087877
```

```r
ggplot(srs_661,aes(x=Inctot)) + geom_histogram(fill="lightblue",col="black")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
svytotal(~Inctot,ipums.Neymanperf)
```

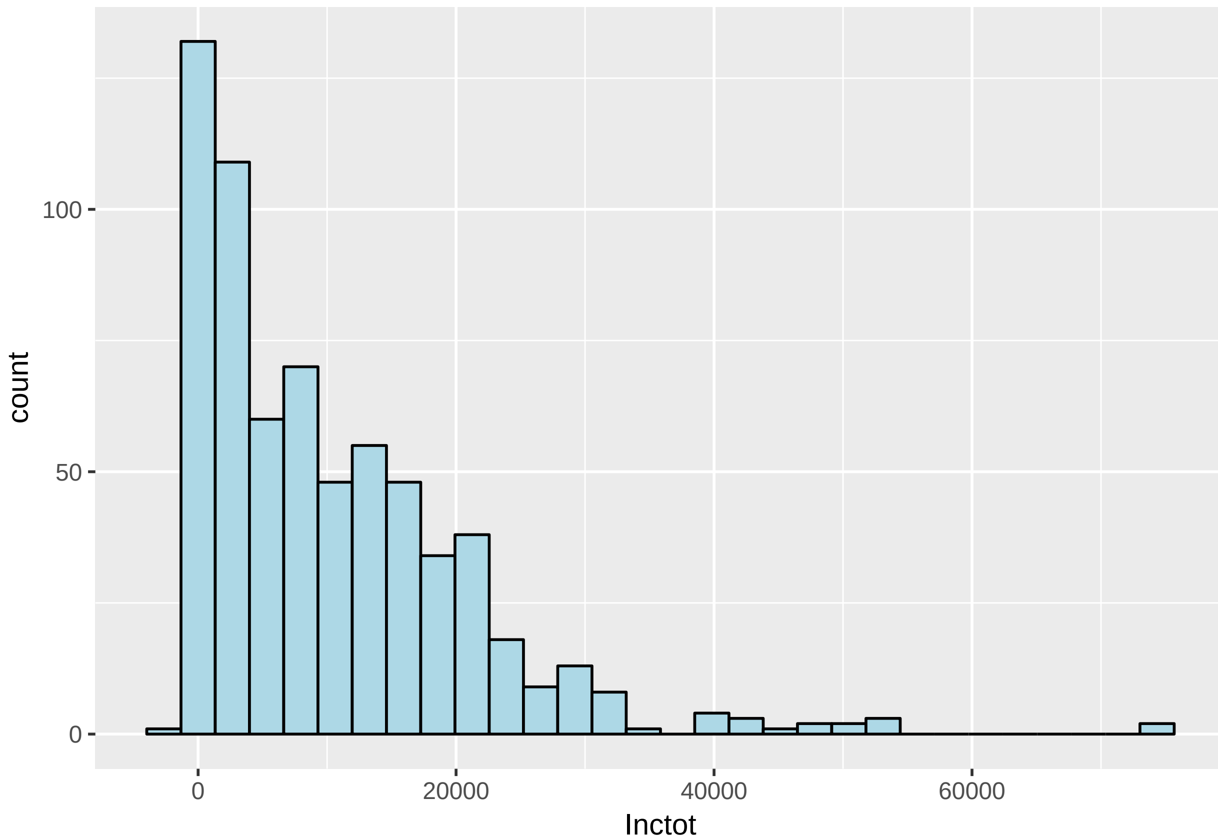
```
##            total       SE
## Inctot 493391266 19465182
```

```
confint(svytotal(~Inctot,ipums.Neymanperf))
```

```
##            2.5 %    97.5 %
## Inctot 455240209 531542322
```

```
ggplot(neyman_perf,aes(x=Inctot)) + geom_histogram(fill="lightblue",col="black")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Overall, I do think the stratification was worth done. Stratified sample returns a bit higher estimate of total population income compared to SRS

For options of alternative stratification I could have done might be:

- Having combinations of factors to construct strata; an example would be a combination of Sex and Age.
- I might have partitoned the strata with narrower intervals.