IPUMS Data Analysis

Yang

May 22, 2019

Read IPUMS Dataset

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.4.4
## -- 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 1.2.0
## v readr   1.3.1     v forcats 0.3.0
## Warning: package 'ggplot2' was built under R version 3.4.4
## Warning: package 'tibble' was built under R version 3.4.4
## Warning: package 'tidyr' was built under R version 3.4.4
## Warning: package 'readr' was built under R version 3.4.4
## Warning: package 'purrr' was built under R version 3.4.4
## Warning: package 'dplyr' was built under R version 3.4.4
## Warning: package 'forcats' was built under R version 3.4.4
## -- Conflicts ------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(survey)
```

```
## Warning: package 'survey' was built under R version 3.4.4
## 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
```

```r
library(srvyr)
```

```
## Warning: package 'srvyr' was built under R version 3.4.4
##
## Attaching package: 'srvyr'
## The following object is masked from 'package:stats':
##
##     filter
```

```r
ipums <- read.csv(file="C:/Program Files/RStudio/ipums.csv", header=TRUE, sep=",")
ipums_complete=ipums %>% filter(complete.cases(.))
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

```r
attach(ipums_complete)
```

Simple Random Sample Approach

Determine desired sample size of SRS

```r
set.seed(19961021)
IPUMS_SRS_50= ipums_complete %>% slice(sample(1:nrow(ipums_complete),size=50, replace=F))
Inctot=IPUMS_SRS_50$Inctot
abs_error=700
S_sq=var(IPUMS_SRS_50$Inctot)
Z=qnorm(0.975)
n=S_sq*(Z**2)/((abs_error**2)+S_sq*(Z**2)/nrow(ipums_complete))
round(n)
```

```
## [1] 599
```

```r
desired_sample_size=round(n)
```

The desired sample size would be 599 using given absolute error e=700.

Taking SRS of size 599

```r
IPUMS_SRS_599= ipums_complete %>% slice(sample(1:nrow(ipums_complete),size=desired_sample_size, replace
IPUMS_SRS_design =survey::svydesign(id=~1,data=IPUMS_SRS_599, fpc=rep(dim(ipums_complete)[1],desired_sam
svytotal(~Inctot,IPUMS_SRS_design)
```

```
##            total       SE
## Inctot 480414915 23446474
```

```r
svymean(~Inctot,IPUMS_SRS_design)
```

```
##          mean     SE
## Inctot 8986.3 438.57
```

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

```
##            2.5 %    97.5 %
## Inctot 434460670 526369160
```

The estimated population total inctot is 480414915. And We are 95% confident that [434460670, 526369160] traps the true population total inctot.

Stratified Random Sample Approach

calculate&compare SSW using different criterion

```
SSW=function(X){
  group_by_var=ipums_complete%>%group_by(vars=X)
  summary_var=group_by_var%>%summarise(mean=mean(Inctot),sum=sum(Inctot),var=var(Inctot))
  sum(summary_var$var)
}

df=data.frame(matrix(ncol = 2))
colnames(df)=c("group by","SSW")
for (var in colnames(ipums_complete)){
  if(var=="Age"||var=="Sex"||var=="Marstat"||var=="Race")
  df=rbind(df,list(var,SSW(eval(parse(text = var)))))
}
df_complete=df%>% filter(complete.cases(.))
df_complete[order(df_complete$SSW),]
```

```
##   group by        SSW
## 2      Sex  205089094
## 4  Marstat  423397534
## 3     Race  447908283
## 1      Age 7814113438
```

From the table above, we know that the strata divided by sex has the smallest SSW, therefore we divide our population according to their sex into two groups:

Divide population into strata using sex as criteria

```
counts_sex =ipums_complete%>%count(Sex)%>%mutate(prop_sex=n/sum(n))
Sex_1=inner_join(ipums_complete,counts_sex,by="Sex")%>%group_by(Sex)%>%subset(Sex==1)
Sex_2=inner_join(ipums_complete,counts_sex,by="Sex")%>%group_by(Sex)%>%subset(Sex==2)
```

Draw Sample Using Proportional Allocation

```
#determined sample size from SRS
set.seed(19961021)
desired_sample_size=599
counts_sex =ipums_complete %>%count(Sex)%>%mutate(prop_sex=n/sum(n),prop_alloc_sex=round(counts_sex$prop

#draw sample by from sex1 and sex2 proportionally(proportional allocation)
Str_sample_599=inner_join(ipums_complete,counts_sex,by="Sex")%>%group_by(Sex)%>%slice(sample(1:n,size=pr
```

```
## Warning in 1:n: numerical expression has 25538 elements: only the first
## used
```

```
## Warning in 1:n: numerical expression has 27923 elements: only the first
## used
```

```r
ipums_str_prop=svydesign(~1,strata=~Sex,data=Str_sample_599,fpc =~n)
svytotal(~Inctot,ipums_str_prop)
```

```
##             total       SE
## Inctot 521658518 21293355
```

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

```
##            2.5 %    97.5 %
## Inctot 479924309 563392727
```

Note: the sample size drawn from each sex group is determined by $desired\ sample\ size * \frac{number\ of\ individuals\ in\ sex\ group\ 1}{total\ number\ of\ individuals}$. The estimated population total inctot is 521658518. And We are 95% confident that [479924309, 563392727] traps the true population total inctot.

Draw Sample Using Optimal Allocation

```r
pilot_sample_size=200
Str_sample_200_sex=inner_join(ipums_complete,counts_sex,by="Sex")%>%group_by(Sex)%>%slice(sample(1:n,si
```

```
## Warning in 1:n: numerical expression has 25538 elements: only the first
## used

## Warning in 1:n: numerical expression has 27923 elements: only the first
## used
```

```r
Sex_1_pilot=Str_sample_200_sex%>%subset(Sex==1)
Sex_2_pilot=Str_sample_200_sex%>%subset(Sex==2)
#calculate sample size in each strata
std_sex1=sqrt(var(Sex_1_pilot$Inctot))
std_sex2=sqrt(var(Sex_2_pilot$Inctot))
v1=std_sex1**2
v2=std_sex2**2
temp1=std_sex1*counts_sex$n[1]
temp2=std_sex2*counts_sex$n[2]
temp_total=temp1+temp2
n1=round(temp1/temp_total*desired_sample_size)
n2=round(temp2/temp_total*desired_sample_size)

Str_optimal=inner_join(ipums_complete,counts_sex,by="Sex")%>%group_by(Sex)%>%slice(sample(1:n,size=rbind
```

```
## Warning in 1:n: numerical expression has 25538 elements: only the first
## used

## Warning in 1:n: numerical expression has 27923 elements: only the first
## used
```

```r
ipums_str_optimal=svydesign(~1,strata=~Sex,data=Str_optimal,fpc =~n)
svytotal(~Inctot,ipums_str_optimal)
```

```
##             total       SE
## Inctot 501677127 19519517
```

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

```
##             2.5 %    97.5 %
## Inctot 463419578 539934676
```

The estimated population total inctot is 501677127. And We are 95% confident that [469171924, 542675441] traps the true population total inctot.

Comparing Variances

```r
library(scales)
```

```
## Warning: package 'scales' was built under R version 3.4.4
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
percent(((v1-v2)/v2))
```

```
## [1] "514%"
```

Optimal allocation performs better than proportional allocation when variances of strata or costs of strata vary greatly. In our case, the relative difference between variance of group 1(Sex=1) over group 2(Sex=2) is 514%. Therefore, assuming that the costs of drawing sample from sex group 1 and sex group 2 are equal, it is reasonable for us to sample more heavily from group 1 to compensate.

Comparing Results from prop alloc and opt alloc

```r
##estimation using proportional allocation
svytotal(~Inctot,ipums_str_prop)
```

```
##            total       SE
## Inctot 521658518 21293355
```

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

```
##             2.5 %    97.5 %
## Inctot 479924309 563392727
```

```r
##estimation using optimal allocation
svytotal(~Inctot,ipums_str_optimal)
```

```
##            total       SE
## Inctot 501677127 19519517
```

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

```
##             2.5 %    97.5 %
## Inctot 463419578 539934676
```

```
##True population total
sum(ipums_complete$Inctot)
```

```
## [1] 491533095
```

Comparing the estimation and CI yielded from stratified random sampling(strata criteria="Sex"), using proportional allocation and optimal allocation, it's obvious that the sample drawn from optimal allocation has a smaller variance and thus we can infer better from its confidence interval. Besides, the estimated total Inctot from optimal allocation is closer to the true population total Inctot. Therefore, we can conclude that stratified random sampling(strata criteria="Sex") with optimal allocation is better than that with proportional allocation.

In all, the stratification using **Sex** does bring an increment of in the precison of estimated total than from the SRS taken in Chapter 2. Built upon that, optimal allocation further improve the performance than proportional allocation.

```
df2=data.frame(matrix(ncol = 2))
colnames(df2)=c("group by","SSW")
Vars=colnames(ipums_complete)
for (var in Vars[4:length(Vars)]){
  SSW(eval(parse(text = var)))
  list(var,SSW(eval(parse(text = var))))
  df2=rbind(df2,list(var,SSW(eval(parse(text = var)))))
}
df_complete2=df2%>% filter(complete.cases(.))
df_complete2[order(df_complete2$SSW),]
```

```
##    group by         SSW
## 8    School   165876334
## 10 Labforce   174498253
## 4  Hispanic   179930699
## 2       Sex   205089094
## 6  Ownershg   225166525
## 13  VetStat   310167041
## 5   Marstat   423397534
## 3      Race   447908283
## 7    Yrsusa   644996144
## 9   Educrec   758561621
## 12  Classwk  1256200657
## 1       Age  7814113438
## 11      Occ 11074714096
```

If the stratification could be done using different criterion, we would look into this using another variable, such as *School*(since it has the smallest SSW which maximize the SSB), to create strata.

Ratio Estimator

```
##take SRS sample of size 599
set.seed(19961021)
desired_sample_size=599
ipums_srs599=ipums_complete%>%slice(sample(1:nrow(ipums_complete), size=desired_sample_size, replace=F))
ipums_srs_design=svydesign(ids=~1,data=ipums_srs599,fpc=~fpc)
##ratio estimation
```

```r
ipums_totals = ipums_complete%>% summarise(T_inctot =sum(Inctot),T_age =sum(Age),B = T_inctot/T_age)
ipums_totals
```

```
##    T_inctot   T_age        B
## 1 491533095 2200842 223.3387
```

```r
r=svyratio(~Inctot,~Age,ipums_srs_design)
r
```

```
## Ratio estimator: svyratio.survey.design2(~Inctot, ~Age, ipums_srs_design)
## Ratios=
##             Age
## Inctot 208.4174
## SEs=
##             Age
## Inctot 10.07756
```

```r
##Confidence interval
confint(r)
```

```
##               2.5 %  97.5 %
## Inctot/Age 188.6657 228.169
```

```r
##predicted r
predicted_r=predict(r,total=ipums_totals%>%pull(T_age))
predicted_r
```

```
## $total
##              Age
## Inctot 458693667
##
## $se
##             Age
## Inctot 22179117
```

```r
#comparing result
##SRS result
svytotal(~Inctot,ipums_srs_design)
```

```
##            total       SE
## Inctot 461964177 21703031
```

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

```
##            2.5 %    97.5 %
## Inctot 419427018 504501335
```

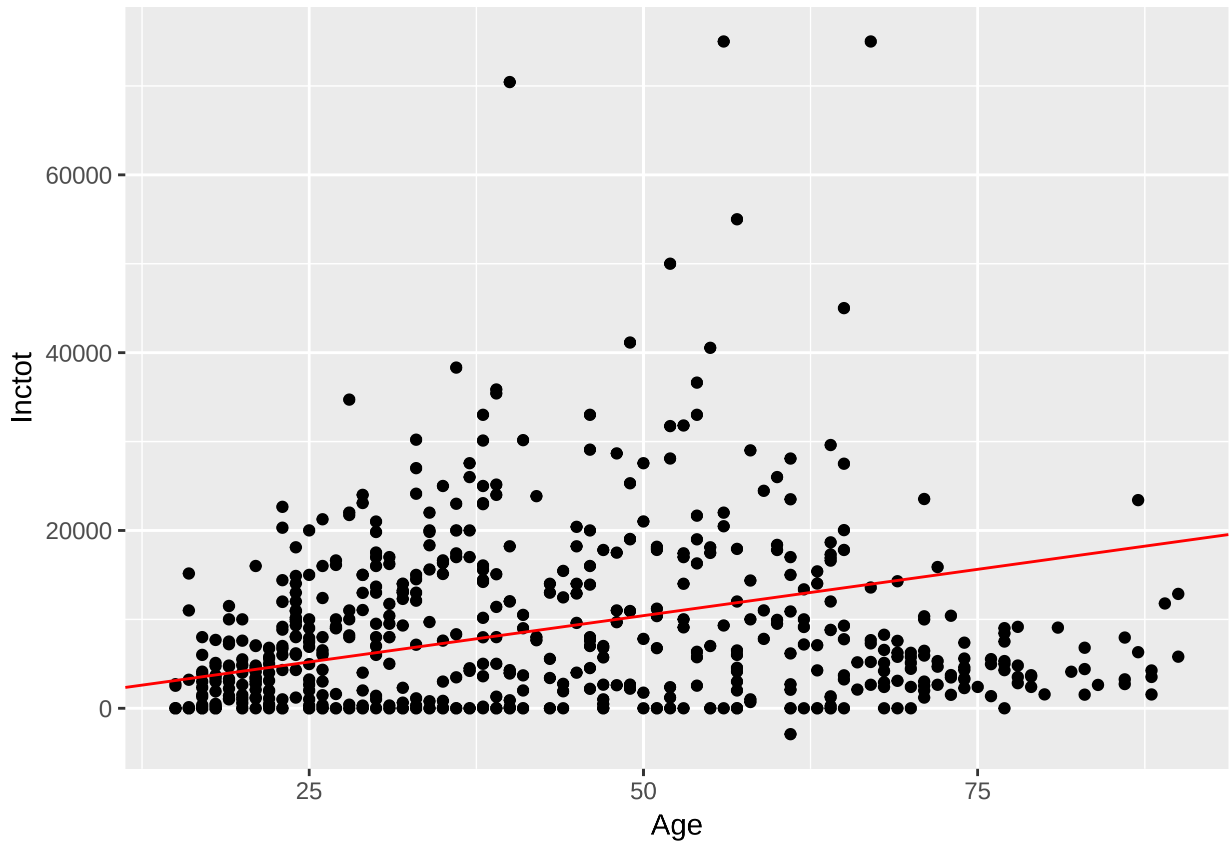
```r
##ratio estimator 95% CI
predicted_r$total+c(qnorm(0.025),qnorm(0.975))*predicted_r$se
```

```
## Warning in c(qnorm(0.025), qnorm(0.975)) * predicted_r$se: Recycling array of length 1 in vector-arra
##   Use c() or as.vector() instead.

## Warning in predicted_r$total + c(qnorm(0.025), qnorm(0.975)) * predicted_r$se: Recycling array of len
##   Use c() or as.vector() instead.

## [1] 415223397 502163938
```

```r
##True total
ipums_complete$Inctot%>%sum()
```

```
## [1] 491533095
```

```
##
ggplot(ipums_srs599,aes(x=Age,y=Inctot))+geom_point()+geom_abline(intercept=0,slope=r[[1]],color="red")
```



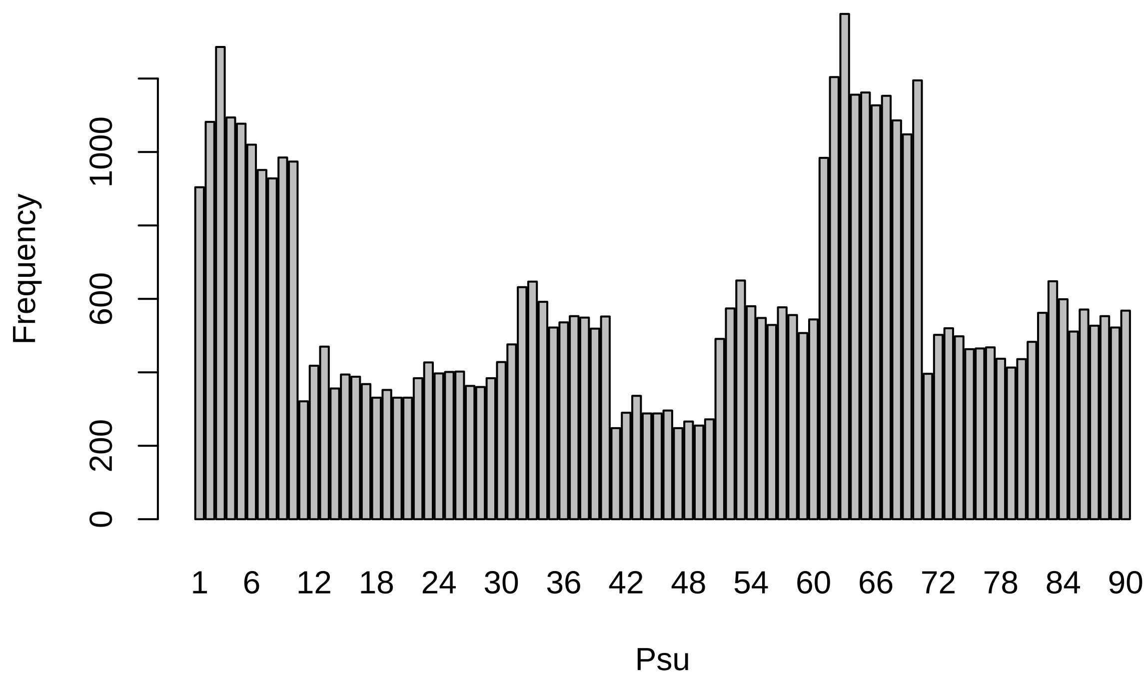
The estimated population total inctot is 461964177. And We are 95% confident that [419427018, 504501335] traps the true population total inctot.

Note that the standard error of ratio estimator(SE=21703031) is lower than that of SRS estimator(SE=23446474).

Cluster Sampling

Frequency table

```
t=table(ipums_complete$Psu)
barplot(t,xlab = "Psu",ylab = "Frequency")
```




```
num_psu = ipums_complete %>% summarise(Num_Clusters = n_distinct(Psu))
num_psu
```

```
##   Num_Clusters
## 1           90
```

```
onestagecluster_sample = ipums_complete %>% filter(Psu %in% sample(unique(Psu),size=10))
onestagecluster_sample %>% summarise(Num_Clusters = n_distinct(Psu))
```

```
##   Num_Clusters
## 1           10
```

```
#calculating total SSUs with same cost as SRS
totalcost=50*599
total_ssus=round((totalcost-10*100)/20)
#proportional allocation
counts_psu =onestagecluster_sample%>%count(Psu)%>%mutate(prop_psu=n/sum(n))
counts_psu=onestagecluster_sample%>%count(Psu)%>%mutate(prop_psu=n/sum(n),prop_alloc_psu=round(counts_ps
onestagecluster_sample = onestagecluster_sample %>% inner_join(counts_psu,by="Psu")%>% rename(fpc2=prop_

index=c(1:dim(onestagecluster_sample)[1])
onestagecluster_sample$id=index
onestagecluster_sample=onestagecluster_sample%>%mutate(fpc1 = num_psu%>%pull(Num_Clusters))
```

```
twostagecluster_sample= onestagecluster_sample %>% group_by(Psu)%>%slice(sample(1:n,size=fpc2,replace=F)
```

```
## Warning in 1:n: numerical expression has 368 elements: only the first used
```

```
## Warning in 1:n: numerical expression has 397 elements: only the first used
```

```
## Warning in 1:n: numerical expression has 428 elements: only the first used
## Warning in 1:n: numerical expression has 536 elements: only the first used
## Warning in 1:n: numerical expression has 266 elements: only the first used
## Warning in 1:n: numerical expression has 272 elements: only the first used
## Warning in 1:n: numerical expression has 520 elements: only the first used
## Warning in 1:n: numerical expression has 483 elements: only the first used
## Warning in 1:n: numerical expression has 599 elements: only the first used
## Warning in 1:n: numerical expression has 568 elements: only the first used
```

```
twostage_design = svydesign(id=~Psu+id,fpc=~fpc1+fpc2,data=twostagecluster_sample)
```

```
svymean(~Inctot,twostage_design)
```

```
##         mean     SE
## Inctot 10338 871.78
```

```
confint(svymean(~Inctot,twostage_design))
```

```
##           2.5 %   97.5 %
## Inctot 8629.848 12047.18
```

```
svytotal(~Inctot,twostage_design)
```

```
##            total       SE
## Inctot 134824545 14708715
```

```
confint(svytotal(~Inctot,twostage_design))
```

```
##            2.5 %    97.5 %
## Inctot 105995994 163653096
```

Result from SRS:Inctot=480414915,SE=23446474. Inctot_mean= 8986.3 SE=438.57. The cluster sampling yield both total inctot and average inctot with smaller SE than SRS.

Unequal Probability Sampling

unequal-probability sample of 10 psus(proportional to num of person) with Replacement

```
set.seed(19961021)
num_clusters = ipums_complete %>% summarise(Num_Clusters = n_distinct(Psu))
Mi_table = ipums %>% group_by(Psu) %>% summarise(Mi = n()) %>%
ungroup() %>% mutate(N = n())
Mi_table = Mi_table %>% mutate(psi_i = Mi/sum(Mi))
Mi_table
```

```
## # A tibble: 90 x 4
##      Psu    Mi     N  psi_i
##    <int> <int> <int>  <dbl>
##  1     1   904    90 0.0169
##  2     2  1082    90 0.0202
##  3     3  1286    90 0.0241
```

```
## 4      4  1094     90 0.0205
## 5      5  1077     90 0.0201
## 6      6  1020     90 0.0191
## 7      7   951     90 0.0178
## 8      8   928     90 0.0174
## 9      9   985     90 0.0184
## 10    10   974     90 0.0182
## # ... with 80 more rows
```

```
##one stage sampling psus
onestage_wr = Mi_table %>% sample_n(size=10, replace=T, weight=Mi)
onestage_wr = onestage_wr %>% group_by(Psu) %>% mutate(replication = 1:n())
onestage_wr %>%head()
```

```
## # A tibble: 6 x 5
## # Groups:   Psu [6]
##     Psu    Mi     N   psi_i replication
##   <int> <int> <int>   <dbl>       <int>
## 1    53   650    90 0.0122            1
## 2    69  1048    90 0.0196            1
## 3     4  1094    90 0.0205            1
## 4    26   402    90 0.00752           1
## 5    34   592    90 0.0111            1
## 6    65  1162    90 0.0217            1
```

```
onestage_sample_wr = inner_join(ipums_complete,onestage_wr,by="Psu") %>%mutate(weight_1 = 1/(10*psi_i))
onestage_sample_wr%>%head()
```

```
##   Stratum Psu Inctot Age Sex Race Hispanic Marstat Ownershg Yrsusa School
## 1       1   4   2510  17   1    1        0       5        1      0      2
## 2       1   4   1005  23   1    1        0       5        1      0      2
## 3       1   4   9505  21   1    1        0       5        1      0      1
## 4       1   4    185  17   1    2        0       5        2      0      1
## 5       1   4   7005  20   1    1        0       5        2      0      1
## 6       1   4   4120  20   1    1        0       5        0      0      2
##   Educrec Labforce Occ Classwk VetStat   Mi  N      psi_i replication
## 1       5        2  59      22       1 1094 90 0.02046352           1
## 2       8        1   8      22       1 1094 90 0.02046352           1
## 3       7        2  15      22       1 1094 90 0.02046352           1
## 4       5        1   8      22       1 1094 90 0.02046352           1
## 5       7        2  32      22       1 1094 90 0.02046352           1
## 6       8        2  84      22       1 1094 90 0.02046352           1
##   weight_1
## 1 4.886746
## 2 4.886746
## 3 4.886746
## 4 4.886746
## 5 4.886746
## 6 4.886746
```

```
Id=c(1:dim(onestage_sample_wr)[1])
onestage_sample_wr$Id=Id
```

Two-stage:subsample of 20 persons in each selected psus

```
twostage_sample_wr = onestage_sample_wr %>% group_by(Psu,replication) %>%
sample_n(size=20,replace=FALSE) %>% ungroup()
twostage_sample_wr = twostage_sample_wr %>% mutate(weight_2=Mi/20)
twostage_sample_wr%>%head()
```

```
## # A tibble: 6 x 23
##   Stratum   Psu Inctot   Age   Sex  Race Hispanic Marstat Ownershg Yrsusa
##     <int> <int>  <int> <int> <int> <int>    <int>   <int>    <int>  <int>
## 1       1     4   1805    17     1     1        0       5        1      0
## 2       1     4  15150    44     1     1        0       1        1      0
## 3       1     4   6765    44     2     1        0       1        1      0
## 4       1     4      0    29     2     2        0       1        1      0
## 5       1     4  16005    34     1     2        0       5        1      0
## 6       1     4      0    37     2     1        0       1        2      0
## # ... with 13 more variables: School <int>, Educrec <int>, Labforce <int>,
## #   Occ <int>, Classwk <int>, VetStat <int>, Mi <int>, N <int>,
## #   psi_i <dbl>, replication <int>, weight_1 <dbl>, Id <int>,
## #   weight_2 <dbl>
```

```
twostage_cluster_wr_design = svydesign(id=~Psu+Id, data = twostage_sample_wr,
weight = ~weight_1+weight_2)
svytotal(~Inctot,twostage_cluster_wr_design )
```

```
##            total       SE
## Inctot 506544312 54095496
```

```
svymean(~Inctot,twostage_cluster_wr_design)
```

```
##        mean     SE
## Inctot 9475 1011.9
```

```
confint(svytotal(~Inctot,twostage_cluster_wr_design ))
```

```
##            2.5 %    97.5 %
## Inctot 400519088 612569535
```

The estimated population total inctot is 506544312. And We are 95% confident that [400519088, 612569535] traps the true population total inctot.