**MSDS 6370**

**Project - Designing a stratified sampling plan**

**Ekaterina Pirogova, Hari Sanadhya**

### **Descriptive Statistics**

The data set contains 9762 observations on 3 variables – id, sales data and inventory data of some industry. Sales and inventory variables are extremely right skewed with heavy tail (high kurtosis). There are no missing values in a data set.



The sales variable has the lowest value of 124.3 and maximum value of about 52 million with the population total of about 1400 millions. Standard deviation of this variable is about 666 thousand.

The inventory variable has the range between 3.03 and 105 million with the population total of about 1755 million. Standard deviation of this variable is about 1.5 million which is about two times more than standard deviation of sales variable.

The non-normality of data can be seen on these histograms and QQ-plots, where we can see that most of the data (almost 100%) concentrated on the low-end values for inventory and sales data with several extremely high observations on the upper values end.

|  |  |
| --- | --- |
| **Sales** | **Inventory** |
|  |  |
|  |  |

### **Stratification Using Inventory**

We have chosen two upper-end observations for certainty stratum. These observations are about 5 times more than the third highest observation. These highest observations have the values of about 101 and 105 million. The next highest observation has a value of about 23 million. Because these two observations present such high values and have significant impact on the mean of the inventory variable we included them in the certainty stratum.

For stratification we excluded certainty stratum with two observations with highest inventory values and used remaining observations to form stratum boundaries.

We wanted to achieve the minimum variance of the estimated mean by minimizing For this reason we implemented equal WhSh method with different number of stratums, trying to achieve equal and minimum WhSh across all stratums. We tried to implement cumulative method and LH method, but these methods produce high discrepancy for some stratums in WhSh values. For example, LH method with number of non-certainty stratums 7 gave WhSh in a range of 3500 – 4500 for 6 stratums and 61590 for the last stratum.

Below is a table of the best WhSh values we were able to achieve with different number of stratums. This method is highly dependable on number of stratums and initial number of observations in each stratum. We used LH method for initial stratum size calculations, because we found this initialization method produces the best results for finding equal WhSh. In the bold letters shown the highest and lowest values.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Stratum | WhSh | WhSh | WhSh | WhSh | WhSh | WhSh |
| 1 | **14,004.3** | **4,385.3** | 9,862.8 | **6,825.1** | 2,349.5 | 2,268.3 |
| 2 | 13,895.6 | 9,952.9 | 4,061.7 | **920.84** | 2,459.6 | **1,640.2** |
| 3 | **13,822.5** | 11,451.3 | 7,655.6 | 5,099.1 | **2,238.7** | 1,944.5 |
| 4 | 13,894.9 | **15,594.1** | 9,823.6 | 3,408.6 | 2,402.9 | 2,532.2 |
| 5 | 13,838.8 | 7,409.5 | **12,361.3** | 1,773.6 | 2,358.3 | 2,095.7 |
| 6 |  | 6,682.1 | 4,896.5 | 2,933.1 | 2,350.6 | 2,221.1 |
| 7 |  |  | **3,564.9** | 2,687.2 | 2,385.3 | 1,872.4 |
| 8 |  |  |  | 3,049.4 | **2,577.1** | **2,565.8** |
| 9 |  |  |  |  | 2,442.0 | 1,831.8 |
| 10 |  |  |  |  |  | 2,442.0 |

We will use 9 non-certainty stratums because it produced the lowest WhSh which also has one of the lowest discrepancies between different stratums.

The table below describes the formed stratums.

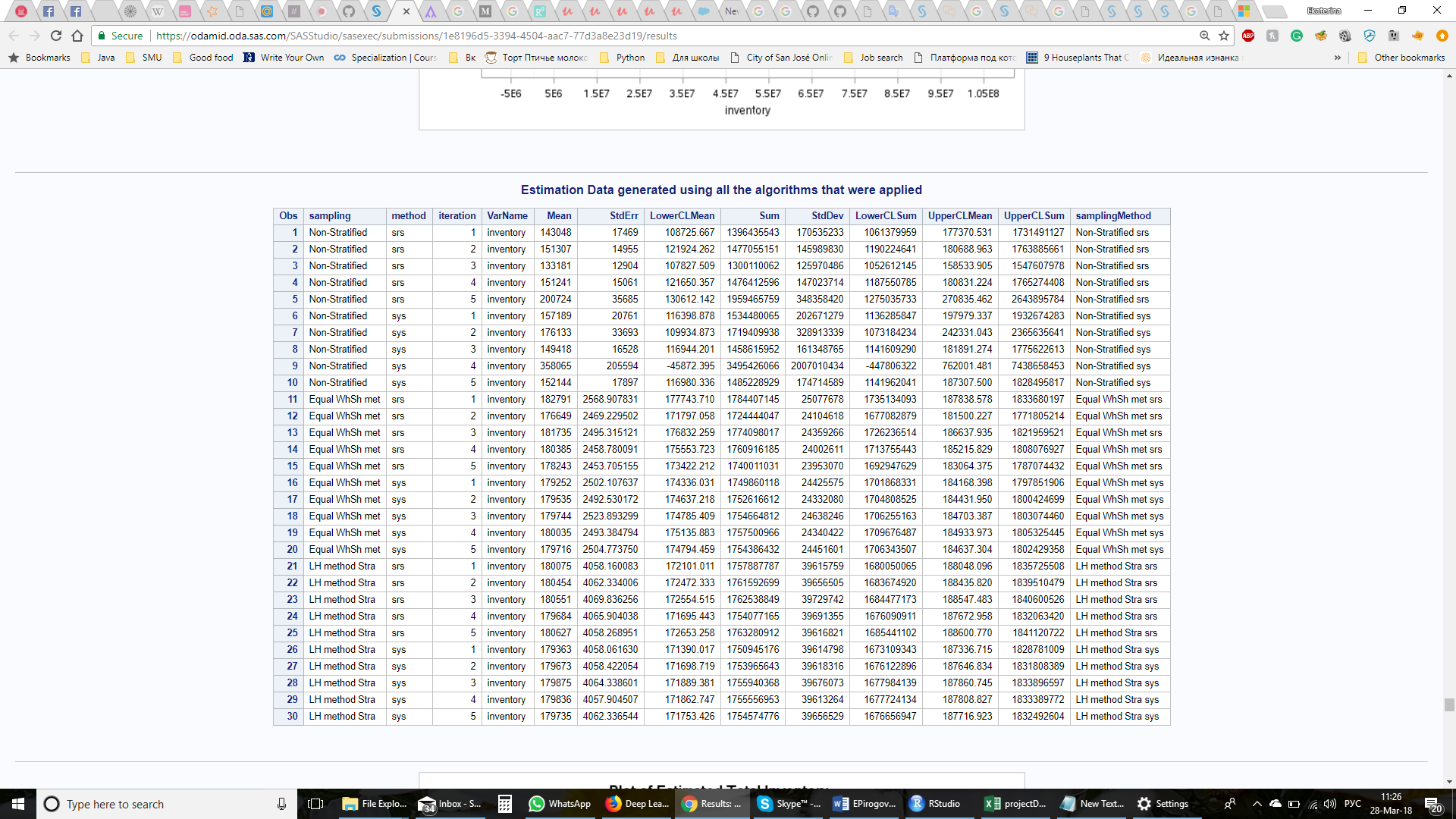
|  |  |  |  |
| --- | --- | --- | --- |
| Stratum # | Number of units | Standard deviation of stratum | Range of values |
| 1 | 3990 | 5,780.963 | 3.03 – 28,469.04 |
| 2 | 1527 | 14,414.95 | 28,472.14 - 75,679.50 |
| 3 | 1650 | 14,534.18 | 75,738.03 - 128,252.19 |
| 4 | 1091 | 20,642.96 | 128,252.60 - 201,510.32 |
| 5 | 620 | 37,311.57 | 201,529.40 - 329,203.60 |
| 6 | 820 | 394,691.07 | 330,569.49 - 2,375,850.76 |
| 7 | 41 | 553,089.90 | 2,380,003.01 - 4,341,052.82 |
| 8 | 16 | 1,372,893.10 | 4,426,464.51 - 8,925,962.41 |
| 9 | 5 | 4,263,559.31 | 10,388,484.51 - 2,819,666.57 |
| 10 | 2 | 1,850,555.95 | 101,678,441.00 - 105,379,552.90 |

For comparison we also provide table of stratums produced by LH method:

|  |  |  |  |
| --- | --- | --- | --- |
| Stratum # | Number of units | Standard deviation of stratum | Range of values |
| 1 | 3990 | 5,780.963 | 3.03 – 18,804.95 |
| 2 | 1527 | 14,414.95 | 18,812.39 - 36,357.48 |
| 3 | 1650 | 14,534.18 | 36,407.34 - 72,752.62 |
| 4 | 1091 | 20,642.96 | 72,784.83 - 113,595.71 |
| 5 | 620 | 37,311.57 | 113,808.72 - 172,603.46 |
| 6 | 820 | 394,691.07 | 172,738.10 - 269,370.45 |
| 7 | 41 | 553,089.90 | 270,223.06 - 471,485.51 |
| 8 | 16 | 1,372,893.10 | 475,206.35 - 903,113.01 |
| 9 | 5 | 4,263,559.31 | 917,023.28 - 22,819,666.57 |
| 10 | 2 | 1,850,555.95 | 101,678,441.00 - 105,379,552.90 |

With chosen strata boundaries we used Neyman allocation to allocate 500 observations between stratums for LH and equal WhSh method using simple random and systematic sampling within stratum boundaries. After that we calculated the mean inventory value for each chosen sample.

The table below provides results for calculation of estimated mean and total Inventory values:



As we can see Non-Stratified simple random sampling produced the worst results with standard error for population mean ranging from about 13 thousands to 206 thousands and produced absolutely non-acceptable results.

On the other hand, both equal WhSh and LH methods captured true population mean and total and produced decent results with standard error for the mean about 2.5 and 4 thousands respectively. Overall, we can say that equal WhSh method produced the best result in estimating mean and population total of inventory.

|  |  |
| --- | --- |
|  |  |

Now let’s compare the LH and equal WhSh methods and systematic vs simple random sampling.

|  |  |
| --- | --- |
|  |  |

As we can see systematic sampling is superior of simple random sampling and produces less variation which is expected with such high skewness of data.

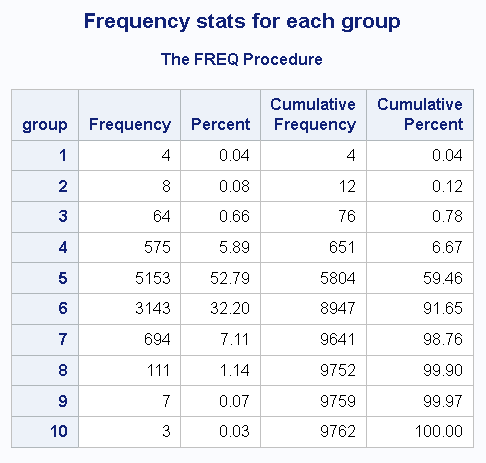
LH method produces values closer to the mean but has higher standard error for the mean.

### **Stratification Using Sales**

For the construction of stratums for estimation of sales, we used two different approaches. Initially after looking into the distribution of Sales data, we decided to take the log of sales to improve the distribution from extremely right skewed with heavy tail to somewhat normalized distribution.

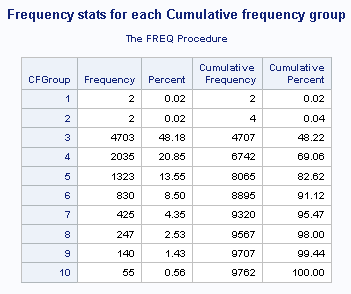
|  |  |
| --- | --- |
|  |  |

The data distribution now looks more to be concentrated on the middle with some observations at the tail. Now we computed the range of the log scaled sales value and divided that range into 10 equal parts. The range of log Sales was 2.09492 to 7.717308 so to divide this range into 10 equal parts, we subtracted minimum value of log Sales from its maximum value and divided by 10. This gave us the interval of 0.5622382. So now to create stratums, what we did was to look into the individual log Sales value and based of who far its from the minimum log Sales value, we assigned the stratum to the observation. This distance of the actual log Sales value from the minimum value was measured in terms of interval (0.5622382). Stratum 1 was assigned to the observations having log Sales value between minimum log sales value and minimum log sales value plus interval. Stratum 2 was observations between minimum log sales value plus interval and minimum log sales value plus twice the interval and so on. Below table shows the strata distribution.



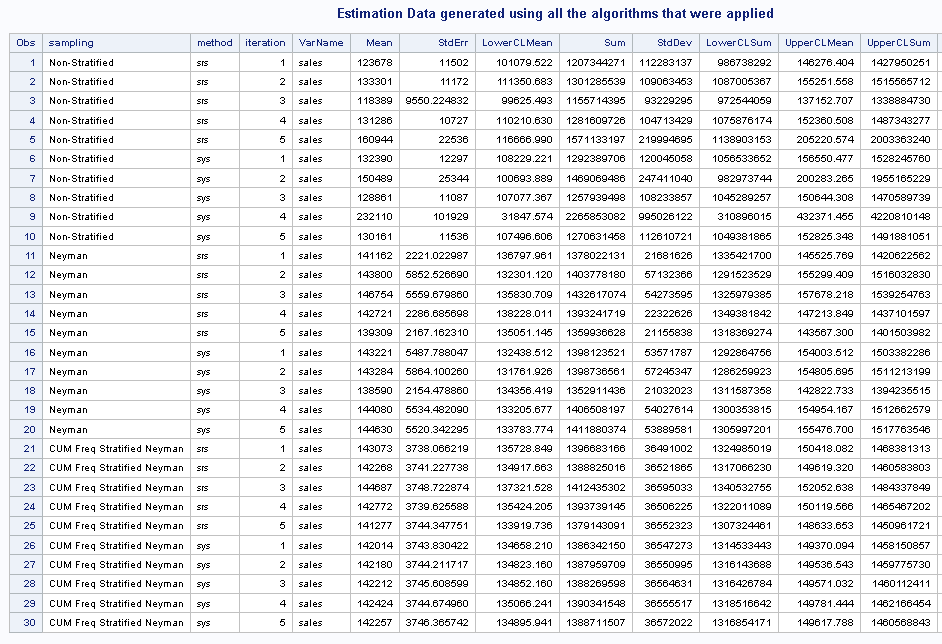
With chosen strata boundaries we used Neyman allocation to allocate 500 observations using simple random and systematic sampling within stratum boundaries.

The second approach was Cumulative Method of forming strata wherein firstly the observations were sorted by Sales and then the cumulative sum of sales was computed. From the QQ-plot of log sales, we observed that the highest 4 values of sales were outlier values and were not fitting greatly into the normalized quantile plot line. So, the four observations with highest sales values were considered as part of certainty stratum. These four observations were divided into two stratums as we wanted to be sure that whatever algorithm is chosen, all the four observations are included in it. Then the remaining observations were divided into 8 stratums using the cumulative sum as the decision maker of the strata where the observation would end up in. To do this, the total of sales excluding the certainty stratum observations were computed and then it was divided by 8 to get the interval. Now once we had the interval, we started creating stratums such that the observations in one stratum are one interval apart from the observation in adjacent stratum. Below table shows the strata distribution.



With these chosen strata boundaries, we used Neyman allocation to allocate 500 observations using simple random and systematic sampling within stratum boundaries

The table below provides results for calculation of estimated mean and total Sales values:



Non-Stratified simple random sampling produced the worst results with standard error for population mean ranging from about 111k to 1019k and are not acceptable results.

On the other hand, both of the stratified sampling approaches captured true population mean and total of sales and produced decent results with average standard error for the mean about 37k. Overall, the stratification approach using cumulative sum of sales with Neyman Sampling yielded best results as the variation in standard error for different samples are nearly the same. The graphs below compares the results obtained

|  |  |
| --- | --- |
|  |  |

Since the results from non-stratified sampling was the worst when compared with the actual population statistics. So, removed that from the dataset and then compared the results from the two stratified approaches using both systematic vs simple random sampling.

|  |  |
| --- | --- |
|  |  |

As we can see systematic sampling is superior of simple random sampling and produces less variation which is expected with such high skewness of data. Cumulative Frequency stratification with Neyman Sampling along with systematic sampling gave the best results with the mean and total sales values closest to the population mean and total sales values.

The graphs below visually compare the average of the estimates obtained in each algorithm with the population statistics. These graphs concludes the same as before that Cumulative Frequency stratification with Neyman Sampling along with systematic sampling gave the best results with the mean and total sales values closest to the population mean and total sales values.

|  |  |
| --- | --- |
|  |  |

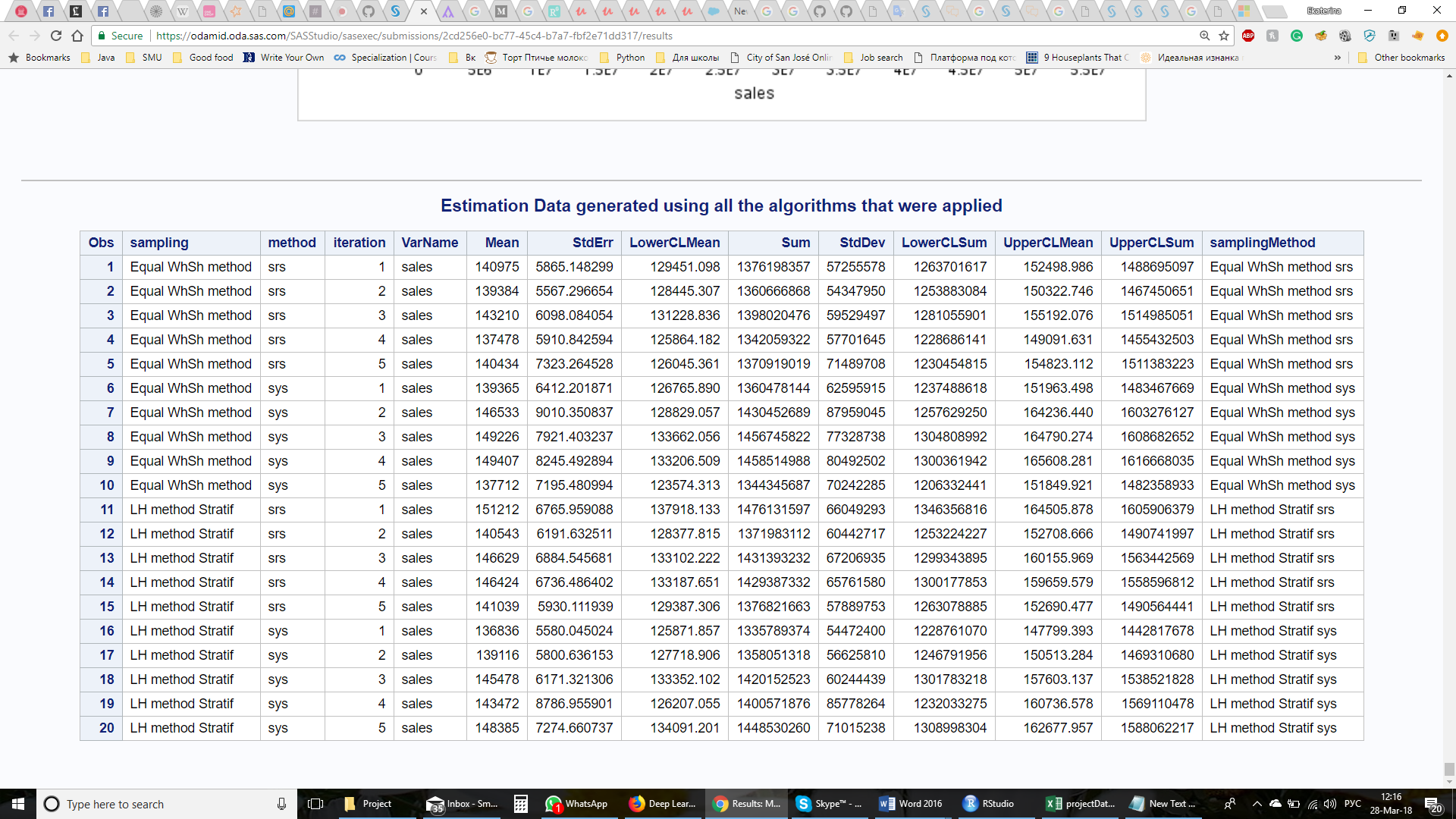
### **Stratification variable choice**

We need to choose variable for stratification. Our main goal is to estimate sales, but inventory variable is also important for us. Below are the charts showing the correlation between Sales and Inventory data.

|  |  |
| --- | --- |
|  |  |
|  | |

In order to choose the best variable for stratification we will plug in the one variable in a stratification method of the other variable and compare produced results.

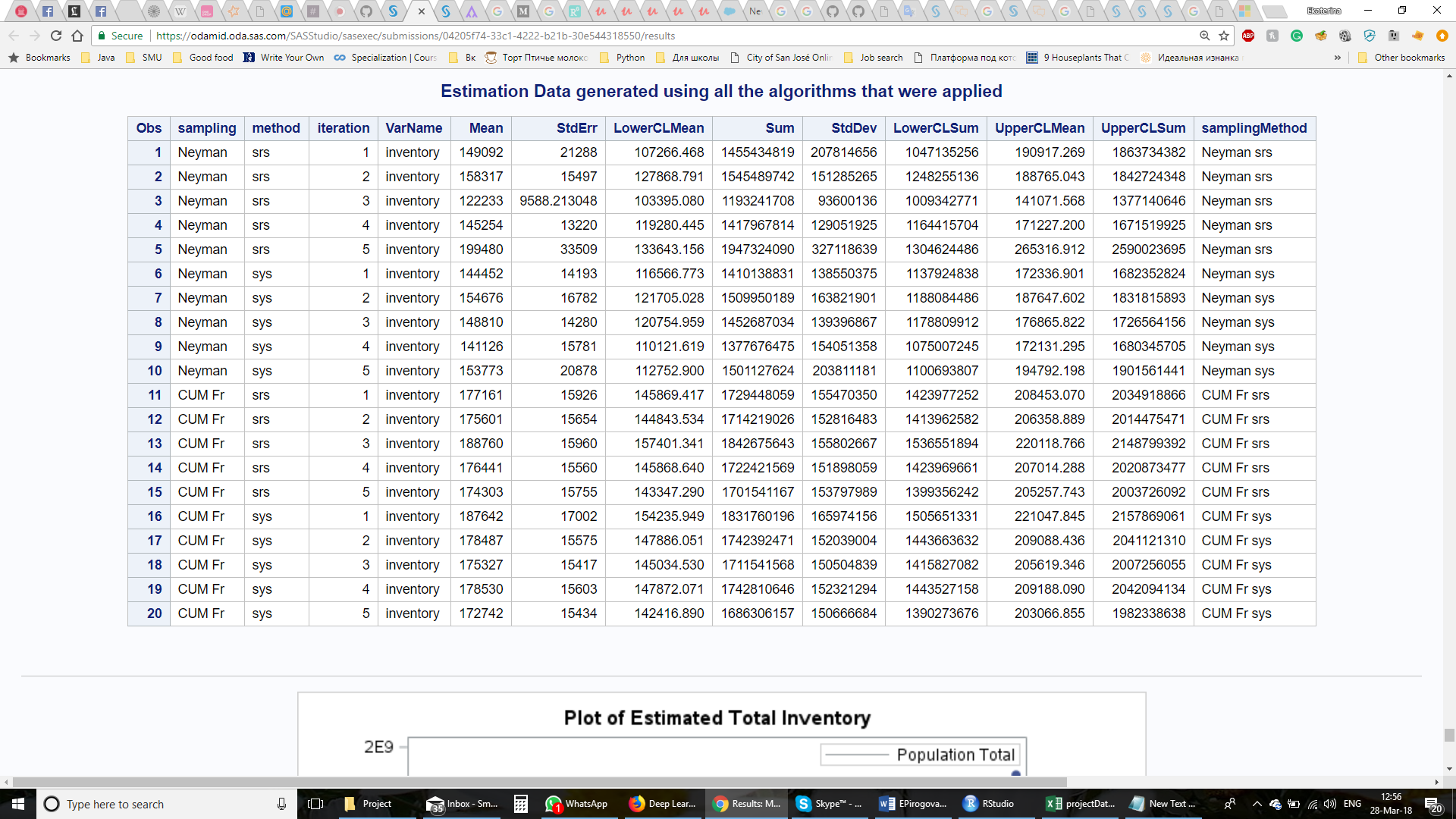
**The mean sales variable estimation using inventory stratification.**



|  |  |
| --- | --- |
|  |  |

As we can see both methods caught true sales mean and total value.

**The mean inventory variable estimation using sales stratification.**



|  |  |
| --- | --- |
|  |  |

As we can see stratification on sales variable underestimates mean inventory value and produces unacceptable high standard error.

The table below show the methods which produced the smallest standard errors for mean for each variable. We were choosing the methods which produces the smallest and less fluctuating standard errors for the mean.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Std.error **Sales** variable | Std.error **Sales** variable | Std.error **Inventory** variable | Std.error **Inventory** variable |
| Stratification | **Sales** | **Inventory** | **Inventory** | **Sales** |
| Method | SYS with cum | SRS with Equal WhSh | SYS with Equal WhSh | SRS with cum |
|  | 3,743 | 5,865 | 2,502 | 15,926 |
| 3,744 | 5,567 | 2,492 | 15,654 |
| 3,745 | 6,098 | 2,523 | 15,960 |
| 3,744 | 5,910 | 2,493 | 15,560 |
| 3,746 | 7,323 | 2,504 | 15,755 |
| Mean std. error | 3,744.40 | 6,152.60 | 2,502.80 | 15,771.00 |

As we can see from the table above stratification sales on inventory variable increases mean standard error in 1.64 times, but stratification inventory on sales variable increases the mean standard error in 6.3 times. While we can accept the increase of standard error for sales variable in 1.63 times, it’s absolutely not acceptable to have such a high standard error for inventory variable under sales variable stratification. For this reason, we have chosen for stratification inventory variable using simple random sampling on stratums produced by equal WhSh method.

## **Conclusion**

Each group should discuss which stratification variable they would choose and support their decision. The discussion should include any trade-offs and compromises that were made in forming the decision. Remember that the quality of the estimate of total Sales has priority over the estimate of total Inventory although both are important. Summarize what you learned about how to stratify a population and design a sampling plan.

We have chosen inventory variable with simple random sampling for stratification purposes using equal WhSh method. Despite that our primary interest is sales variable, stratification on this variable provides unacceptable high standard errors for inventory variable – 6.3 times higher than with stratification on inventory variable. On the other hand, the stratification using inventory variable increases standard error for the mean of the sales not so dramatically – only in 1.63 times. The chosen stratification variable and method will provide us with quality estimates of variables and parameters of interest with highly increased precision compared with estimations without stratification.

## Appendix

### Code for Sales Data Analysis

#### SAS Code for Sales Data Analysis (Includes Inventory Estimation using Sales)

/\* MSDS 6370 - Statistical Sampling \*/

/\* Spring Project \*/

/\* Team - Ekaterina Pirogova, Hari Narayan Sanadhya \*/

/\* Drop the file if already exists \*/

%web\_drop\_table(WORK.industryData);

%web\_drop\_table(WORK.projectOutputSummary);

%web\_drop\_table(WORK.projectSummary);

%web\_drop\_table(WORK.projectOutInventory);

/\* Path to the file \*/

FILENAME datafile '/home/harisanadhya0/sasuser.v94/MSDS 6370/projectSpring2018/projectData.csv';

/\* Import the dataset \*/

data WORK.industryData;

infile datafile delimiter = ',' MISSOVER DSD firstobs=2 ;

informat

coID best32.

sales best32.

inventory best32. ;

format

coID best16.

sales best16.

inventory best16. ;

input coID sales inventory;

run;

/\* Print the first 10 records of the dataset to verify the data imported \*/

title "Raw Dataset (First 10 records)";

proc print data=work.industrydata(obs=10);

run;

/\* Generate the descriptive statistics \*/

title "Descriptive Statistics";

proc means data=work.industrydata n min q1 median q3 max mean var std sum skewness kurtosis missing nmiss;

run;

/\* Worked on Sales variable so sorting by Sales \*/

proc sort data=work.industrydata out=work.industrydata;

by sales;

run;

/\* Descriptive statistics for the sales output shows the distribution \*/

/\* is right skewness and has a large tail (Kurtosis value)\*/

/\* Plot histogram and qqplot to view confirm the same \*/

title "Distribution of Sales Data";

proc univariate data=work.industrydata noprint;

var sales;

histogram sales;

qqplot Sales/normal(mu=est sigma=est color=red l=2) square;

run;

/\* Since the sales data is rightskewed, perform log transformation to reduce the skewness \*/

data work.industrydata;

set work.industrydata;

logSales = log10(sales);

logInventory = log10(inventory);

run;

/\* Plot the scatter plot showing Correlation between Sales and Inventory \*/

ods exclude all;

ods select where=(\_name\_ ? 'ScatterPlot');

proc corr data=work.industrydata plots=scatter plots(maxpoints = 10000);

var sales inventory;

title "Correlation between Sales and Inventory";

run;

ods exclude all;

ods select where=(\_name\_ ? 'ScatterPlot');

proc corr data=work.industrydata plots(maxpoints = 10000) plots=scatter ;

var sales inventory;

where sales<200000 and inventory<200000;

title "Correlation between Sales and Inventory(without outliers)";

run;

ods exclude all;

ods select where=(\_name\_ ? 'ScatterPlot');

proc corr data=work.industrydata plots=all plots(maxpoints = 10000);

var logsales loginventory;

title "Correlation between logSales and logInventory";

run;

ods select all;

/\* Plot histogram and qqplot \*/

/\* Distribution of Sales Data after log transformation \*/

title "Distribution of Log of Sales Data";

proc univariate data=work.industrydata noprint;

var logSales;

histogram logsales;

qqplot logSales/ normal(mu=est sigma=est color=red l=2) square;

run;

/\* Get the population count, total, mean, minimum and maximum value for sales variable \*/

%macro getStats();

%global recordCount minSalesValue maxSalesValue meanPopSales totalPopSales;

proc sql noprint;

select count(logSales), min(logSales), max(logSales), avg(sales), sum(sales)

into :recordCount, :minSalesValue, :maxSalesValue, :meanPopSales, :totalPopSales

from work.industrydata;

quit;

%mend;

%getStats();

/\* Interval is the range of logsales value based of which 10 statrums are created \*/

%let interval = %sysevalf((&maxSalesValue. - &minSalesValue.)/10);

%put &recordCount &minSalesValue &maxSalesValue &interval;

/\* Group data based of the log transformed column \*/

data work.industrydata;

set work.industrydata;

group = 10;

if logSales<=(&minSalesValue + &interval) then group=1;

else if logSales<=(&minSalesValue + (2 \* &interval)) then group=2;

else if logSales<=(&minSalesValue + (3 \* &interval)) then group=3;

else if logSales<=(&minSalesValue + (4 \* &interval)) then group=4;

else if logSales<=(&minSalesValue + (5 \* &interval)) then group=5;

else if logSales<=(&minSalesValue + (6 \* &interval)) then group=6;

else if logSales<=(&minSalesValue + (7 \* &interval)) then group=7;

else if logSales<=(&minSalesValue + (8 \* &interval)) then group=8;

else if logSales<=(&minSalesValue + (9 \* &interval)) then group=9;

run;

title "Descriptive Statistics with logSales";

ods exclude all;

proc means data=work.industrydata n min max mean var std skewness kurtosis;

var sales logsales;

by group;

ods output Summary = groupedSummary;

run;

ods select all;

proc print data=groupedSummary;

run;

title '';

/\* Get the variance for each group and save them in the variable \*/

%macro findVar();

%global varSalesStrata;

proc sql noprint;

select var(sales) format 16.

into :varSalesStrata separated by ','

from work.industrydata group by group;

run;

%mend;

%findVar();

/\* Print the variance of reach strata (comma separated) \*/

%put &varSalesStrata;

/\* View the distribution of the group - i.e. number of obs within each group \*/

proc freq data=work.industrydata(keep=group);

title "Frequency stats for each group";

run;

/\* Seeds used \*/

%let seed1 = 306609547;

%let seed2 = 606868685;

%let seed3 = 621046942;

%let seed4 = 635888608;

%let seed5 = 252341977;

/\* Weights for SRS \*/

data work.industrydata;

set work.industrydata;

samplingWeight = &recordCount/500;

run;

/\* Macro running the sampling and then estimation process 5 times \*/

%macro srs\_sys\_Sampling(srs\_sys);

%do i=1 %to 5;

/\* Estimation of Sales using SRS \*/

proc surveyselect data=work.industrydata sampsize=500 out=SRSSAMPLE method= &srs\_sys seed=&&seed&i;

%let title = Sampling using &srs\_sys;

title &title;

run;

proc surveymeans data=SRSSAMPLE total=&recordCount mean stderr clm sum std clsum;

var sales;

weight samplingWeight;

ods output Statistics = surveyMeansoutput;

run;

%let temp=%TSLIT(&srs\_sys);

%put &temp;

%if %sysfunc(exist(WORK.projectOutputSummary)) %then %do;

proc sql noprint;

insert into work.projectOutputSummary

select 'Non-Stratified' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

run;

%end;

%else %do;

proc sql noprint;

create table work.projectOutputSummary as

select 'Non-Stratified' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

run;

%end;

%end;

%mend;

%srs\_sys\_sampling(srs);

%srs\_sys\_sampling(sys);

/\* Drop the samplingWeights that was used for SRS \*/

data work.industrydata;

set work.industrydata;

drop samplingWeight;

run;

%macro srs\_sys\_sampling\_neyman(srs\_sys);

%local temp;

%do i=1 %to 5;

/\* Estimation of Sales using SRS \*/

proc surveyselect data=work.industrydata sampsize=500 out=SRSSAMPLE method= &srs\_sys seed=&&seed&i;

%let title = Neyman Sampling using &srs\_sys;

strata group/ alloc=neyman var=(&varSalesStrata) allocmin=2;

title &title;

run;

proc surveymeans data=SRSSAMPLE total=&recordCount mean stderr clm sum std clsum;

var sales;

strata group;

weight samplingWeight;

ods output Statistics = surveyMeansoutput;

run;

%let temp=%TSLIT(&srs\_sys);

%put &temp;

%if %sysfunc(exist(WORK.projectOutputSummary)) %then %do;

proc sql noprint;

insert into work.projectOutputSummary

select 'Neyman' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

run;

%end;

%else %do;

proc sql noprint;

create table work.projectOutputSummary as

select 'Neyman' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

run;

%end;

%end;

%mend;

%srs\_sys\_sampling\_neyman(srs);

%srs\_sys\_sampling\_neyman(sys);

/\* Cumulative Frequency method \*/

title "Certainity group data - LogSales>=7 (Highest Outlier values)";

proc sql;

/\* These are the records in the certainity group \*/

select \* from WORK.industryData where logSales>=7.0;

run;

/\* Get the total of the sales column without including the data in certainity data \*/

proc sql noprint;

select sum(sales), min(Sales) into :totalSales, :minSales from WORK.industryData where logSales<7.0;

run;

%let stratumRange = %sysevalf(&totalSales/8);

%put &stratumRange &minSales;

/\* Get the Cumulative sum of sales \*/

data work.industrydata;

set work.industrydata;

retain sum 0;

sum = sum + sales;

keep sum coID sales logSales group inventory;

run;

/\* Divide data into stratums - First 2 stratums are the certainity data \*/

data work.industrydata;

set work.industrydata;

retain count 0;

CFGroup = 10;

if logSales>=7.0 then do;

if count<2 then do;

CFGroup = 1;

count = count + 1;

end;

else CFGroup = 2;

end;

else if sum <=(&minSales + &stratumRange) then CFGroup=3;

else if sum <=(&minSales + (2 \* &stratumRange)) then CFGroup=4;

else if sum <=(&minSales + (3 \* &stratumRange)) then CFGroup=5;

else if sum <=(&minSales + (4 \* &stratumRange)) then CFGroup=6;

else if sum <=(&minSales + (5 \* &stratumRange)) then CFGroup=7;

else if sum <=(&minSales + (6 \* &stratumRange)) then CFGroup=8;

else if sum <=(&minSales + (7 \* &stratumRange)) then CFGroup=9;

run;

/\* View the distribution of the group - i.e. number of obs within each group created using Cumulative frequency \*/

proc freq data=work.industrydata(keep=CFGroup);

title "Frequency stats for each Cumulative frequency group";

run;

proc sort data=work.industrydata;

by cfgroup;

run;

/\* Increase the size of the sampling column in work.projectOutputSummary dataset \*/

proc sql noprint;

alter table work.projectOutputSummary modify sampling varchar(50);

run;

proc print data=work.projectOutputSummary ;

run;

/\* Get the variance for each group and save them in the variable \*/

%macro findVar();

%global varSalesStrata;

proc sql noprint;

select var(sales) format 16.

into :varSalesStrata separated by ','

from work.industrydata group by CFGroup;

run;

%mend;

%findVar();

%put &varSalesStrata;

%macro srs\_sys\_sampling\_neyman\_cf(srs\_sys);

%local temp;

%do i=1 %to 5;

/\* Estimation of Sales using SRS \*/

proc surveyselect data=work.industrydata sampsize=500 out=SRSSAMPLE method= &srs\_sys seed=&&seed&i;

%let title = Neyman Sampling using &srs\_sys - Stratified using Cumulative Frequency;

strata CFGroup/ alloc=neyman var=(&varSalesStrata) allocmin=2;

title &title;

run;

proc surveymeans data=SRSSAMPLE total=&recordCount mean stderr clm sum std clsum;

var sales;

strata CFGroup;

weight samplingWeight;

ods output Statistics = surveyMeansoutput;

run;

%let temp=%TSLIT(&srs\_sys);

%put &temp;

%if %sysfunc(exist(WORK.projectOutputSummary)) %then %do;

proc sql noprint;

insert into work.projectOutputSummary

select 'CUM Freq Stratified Neyman' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

run;

%end;

%else %do;

proc sql noprint;

create table work.projectOutputSummary as

select 'CUM Freq Stratified Neyman' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

run;

%end;

%end;

%mend;

%srs\_sys\_sampling\_neyman\_cf(srs);

%srs\_sys\_sampling\_neyman\_cf(sys);

/\* Since all the algorithms were executed 5 times - Compute the average estimated value of mean and sum of sales \*/

ods exclude all;

proc means mean data=work.projectoutputsummary(keep=sum mean sampling method);

by sampling method notsorted;

ods output summary=projectSummary;

run;

ods select all;

/\* Combine the sampling and method columns in the projectoutputsummary dataset to create the samplingMethod column \*/

proc sql noprint;

alter table work.projectoutputsummary add samplingMethod varchar(50);

update work.projectoutputsummary set samplingMethod= cat(strip(sampling),cat(" ", strip(method)));

run;

proc print data=work.projectoutputsummary;

title "Estimation Data generated using all the algorithms that were applied";

run;

/\* Plot the estimated total sales value obtained by each algorithm for every iteration \*/

proc sgplot data=work.projectoutputsummary des="Plot of Estimated Total Sales";

series x = iteration y = sum /group=samplingMethod markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=2)

name="Legend2" legendlabel="Sampling Method";;

refline &totalPopSales/AXIS=y legendlabel="Population Total" name="Legend1" label;

keylegend "Legend1"/location=inside position=topright;

keylegend "Legend2"/location=outside position=bottom title="Sampling Method";

title "Plot of Estimated Total Sales";

run;

/\* Plot the estimated average sales value obtained by each algorithm for every iteration \*/

proc sgplot data=work.projectoutputsummary des="Plot of Estimated Mean Sales";

refline &meanPopSales/AXIS=y legendlabel="Population Mean" name="Legend1" label;

series x = iteration y = mean / group=samplingMethod markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=2)

name="Legend2" legendlabel="Sampling Method";

keylegend "Legend1"/location=inside position=topright;

keylegend "Legend2"/location=outside position=bottom title="Sampling Method";

title "Plot of Estimated Mean Sales";

run;

/\* Above plots showed that of all the algorithms applied, Non-stratified design gave the worst predictions \*/

/\* Remove the Non-stratified design predictions and replot the data for more clear visualization of the estimations \*/

proc sql noprint;

delete from work.projectoutputsummary where sampling= 'Non-Stratified';

run;

/\* Plot the estimated total sales value obtained by each algorithm for every iteration \*/

proc sgplot data=work.projectoutputsummary des="Plot of Estimated Total Sales";

series x = iteration y = sum /group=samplingMethod markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=2)

name="Legend2" legendlabel="Sampling Method";;

refline &totalPopSales/AXIS=y legendlabel="Population Total" name="Legend1" label;

keylegend "Legend1"/location=inside position=topright;

keylegend "Legend2"/location=outside position=bottom title="Sampling Method";

title "Plot of Estimated Total Sales";

run;

/\* Plot the estimated mean sales value obtained by each algorithm for every iteration \*/

proc sgplot data=work.projectoutputsummary des="Plot of Estimated Mean Sales";

refline &meanPopSales/AXIS=y legendlabel="Population Mean" name="Legend1" label;

series x = iteration y = mean / group=samplingMethod markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=2)

name="Legend2" legendlabel="Sampling Method";

keylegend "Legend1"/location=inside position=topright;

keylegend "Legend2"/location=outside position=bottom title="Sampling Method";

title "Plot of Estimated Mean Sales";

run;

/\* Combine the sampling and method columns in the projectsummary dataset to create the samplingMethod column \*/

/\* This dataset has the average estimation received from all the iterations for a given algorithm \*/

proc sql noprint;

alter table work.projectsummary add samplingMethod varchar(50);

update work.projectsummary set samplingMethod= cat(strip(sampling),cat(" ", strip(method)));

run;

/\* Plot the estimated mean sales value obtained by each algorithm averaged over the iterations \*/

proc sgplot data=work.projectsummary des="Plot of Estimated Mean Sales - Summarized Info";

scatter x = samplingMethod y = Mean\_Mean;

refline &meanPopSales/AXIS=y legendlabel="Population Mean" name="Legend1" label;

title "Plot of Estimated Mean Sales - Average statistics";

run;

/\* Plot the estimated total sales value obtained by each algorithm averaged over the iterations \*/

proc sgplot data=work.projectsummary des="Plot of Estimated Total Sales - Summarized Info";

scatter x = samplingMethod y = Sum\_Mean;

refline &totalPopSales/AXIS=y legendlabel="Population Total" name="Legend1" label;

title "Plot of Estimated Total Sales - Average Statistics";

run;

/\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*inventory variable calculation with sales stratification\*/

/\* Get the variance for each group and save them in the variable \*/

proc sort data=work.industrydata out=work.industrydata;

by group;

run;

/\* Get the population count, total, mean, minimum and maximum value for inventory variable \*/

%macro getStats();

    %global recordCount mininventoryValue maxinventoryValue meanPopInventory totalPopInventory meanPopSales totalPopSales;

proc sql noprint;

select count(inventory), min(inventory), max(inventory),avg(inventory), sum(inventory), avg(sales), sum(sales)

into :recordCount, :mininventoryValue, :maxinventoryValue, :meanPopInventory, :totalPopInventory, :meanPopSales, :totalPopSales

from work.industrydata;

quit;

%mend;

%getStats();

%macro findVar();

    %global varSalesStrata;

    proc sql noprint;

    select var(sales) format 16.

    into :varSalesStrata separated by ','

    from work.industrydata group by group;

    run;

%mend;

%findVar();

/\* Print the variance of reach strata (comma separated) \*/

%put &varSalesStrata;

%macro srs\_sys\_sampling\_neyman(srs\_sys);

%local temp;

    %do i=1 %to 5;

        /\* Estimation of Sales using SRS \*/

        proc surveyselect data=work.industrydata sampsize=500 out=SRSSAMPLE method= &srs\_sys seed=&&seed&i;

        %let title = Neyman Sampling using &srs\_sys;

        strata group/ alloc=neyman var=(&varSalesStrata) allocmin=2;

        title &title;

        run;

        proc surveymeans data=SRSSAMPLE total=&recordCount mean stderr clm sum std clsum;

        var inventory;

        strata group;

        weight samplingWeight;

        ods output Statistics = surveyMeansoutput;

        run;

        %let temp=%TSLIT(&srs\_sys);

        %put &temp;

        %if %sysfunc(exist(WORK.projectOutInventory)) %then %do;

            proc sql noprint;

            insert into work.projectOutInventory

            select 'Neyman' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

            run;

        %end;

        %else %do;

            proc sql noprint;

            create table work.projectOutInventory as

            select 'Neyman' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

            run;

        %end;

    %end;

%mend;

%srs\_sys\_sampling\_neyman(srs);

%srs\_sys\_sampling\_neyman(sys);

/\* Get the variance for each group and save them in the variable \*/

proc sort data=work.industrydata out=work.industrydata;

by CFGroup;

run;

%macro findVar();

    %global varSalesStrata;

    proc sql noprint;

    select var(sales) format 16.

    into :varSalesStrata separated by ','

    from work.industrydata group by CFGroup;

    run;

%mend;

%findVar();

%put &varSalesStrata;

%macro srs\_sys\_sampling\_neyman\_cf(srs\_sys);

%local temp;

    %do i=1 %to 5;

        /\* Estimation of Sales using SRS \*/

        proc surveyselect data=work.industrydata sampsize=500 out=SRSSAMPLE method= &srs\_sys seed=&&seed&i;

        %let title = Neyman Sampling using &srs\_sys - Stratified using Cumulative Frequency;

        strata CFGroup/ alloc=neyman var=(&varSalesStrata) allocmin=2;

        title &title;

        run;

        proc surveymeans data=SRSSAMPLE total=&recordCount mean stderr clm sum std clsum;

        var inventory;

        strata CFGroup;

        weight samplingWeight;

        ods output Statistics = surveyMeansoutput;

        run;

        %let temp=%TSLIT(&srs\_sys);

        %put &temp;

        %if %sysfunc(exist(WORK.projectOutInventory)) %then %do;

            proc sql noprint;

            insert into work.projectOutInventory

            select 'CUM Freq Stratified Neyman' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

            run;

        %end;

        %else %do;

            proc sql noprint;

            create table work.projectOutInventory as

            select 'CUM Freq Stratified Neyman' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

            run;

        %end;

    %end;

%mend;

%srs\_sys\_sampling\_neyman\_cf(srs);

%srs\_sys\_sampling\_neyman\_cf(sys);

/\* Since all the algorithms were executed 5 times - Compute the average estimated value of mean and sum of sales \*/

ods exclude all;

proc means mean data=work.projectOutInventory(keep=sum mean sampling method);

by sampling method notsorted;

ods output summary=projectSummary;

run;

ods select all;

/\* Combine the sampling and method columns in the projectOutInventory dataset to create the samplingMethod column \*/

proc sql noprint;

alter table work.projectOutInventory add samplingMethod varchar(50);

update work.projectOutInventory set samplingMethod= cat(strip(sampling),cat(" ", strip(method)));

run;

proc print data=work.projectOutInventory;

title "Estimation Data generated using all the algorithms that were applied";

run;

/\* Plot the estimated total inventory value obtained by each algorithm for every iteration \*/

proc sgplot data=work.projectOutInventory des="Plot of Estimated Total Inventory";

series x = iteration y = sum /group=samplingMethod markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=2)

    name="Legend2" legendlabel="Sampling Method";;

refline &totalPopInventory/AXIS=y legendlabel="Population Total" name="Legend1" label;

keylegend "Legend1"/location=inside position=topright;

keylegend "Legend2"/location=outside position=bottom title="Sampling Method";

title "Plot of Estimated Total Inventory";

run;

/\* Plot the estimated average inventory value obtained by each algorithm for every iteration \*/

proc sgplot data=work.projectOutInventory des="Plot of Estimated Mean Inventory";

refline &meanPopinventory/AXIS=y legendlabel="Population Mean" name="Legend1" label;

series x = iteration y = mean / group=samplingMethod markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=2)

    name="Legend2" legendlabel="Sampling Method";

keylegend "Legend1"/location=inside position=topright;

keylegend "Legend2"/location=outside position=bottom title="Sampling Method";

title "Plot of Estimated Mean Inventory";

run;

/\* Above plots showed that of all the algorithms applied, Non-stratified design gave the worst predictions \*/

/\* Remove the Non-stratified design predictions and replot the data for more clear visualization of the estimations \*/

proc sql noprint;

delete from work.projectOutInventory where sampling= 'Non-Stratified';

run;

/\* Plot the estimated total Inventory value obtained by each algorithm for every iteration \*/

proc sgplot data=work.projectOutInventory des="Plot of Estimated Total Inventory";

series x = iteration y = sum /group=samplingMethod markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=2)

    name="Legend2" legendlabel="Sampling Method";;

refline &totalPopInventory/AXIS=y legendlabel="Population Total" name="Legend1" label;

keylegend "Legend1"/location=inside position=topright;

keylegend "Legend2"/location=outside position=bottom title="Sampling Method";

title "Plot of Estimated Total Inventory";

run;

/\* Plot the estimated mean Inventory value obtained by each algorithm for every iteration \*/

proc sgplot data=work.projectOutInventory des="Plot of Estimated Mean Inventory";

refline &meanPopInventory/AXIS=y legendlabel="Population Mean" name="Legend1" label;

series x = iteration y = mean / group=samplingMethod markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=2)

    name="Legend2" legendlabel="Sampling Method";

keylegend "Legend1"/location=inside position=topright;

keylegend "Legend2"/location=outside position=bottom title="Sampling Method";

title "Plot of Estimated Mean Inventory";

run;

/\* Combine the sampling and method columns in the projectsummary dataset to create the samplingMethod column \*/

/\* This dataset has the average estimation received from all the iterations for a given algorithm \*/

proc sql noprint;

alter table work.projectsummary add samplingMethod varchar(50);

update work.projectsummary set samplingMethod= cat(strip(sampling),cat(" ", strip(method)));

run;

### Code for Inventory Data Analysis

#### R-code to form stratum's using Lh and equal WhSh method

#this data is already sorted

data = read.csv('C:/Users/25355/OneDrive/Study/SMU/Term4/Statistical Sampling/Project/data.csv',sep=",",header=TRUE)

#drop rows with certanity value

maxIn = max(data$inventory)

data = data[!data$inventory==maxIn,]

maxIn = max(data$inventory)

data = data[!data$inventory==maxIn,]

#data ordered in increasing order by stratification variable

popSize = 9760

#initial values for stratum size

strata.cumrootf(data$inventory,n=500, Ls=5)

strata.LH(data$inventory,n=500, Ls=5)

#initial values for stratum size cum sqrt(f)

s1=8779

s2=725

s3=229

s4=25

s5=2

#LH

s1=4718

s2=2928

s3=1378

s4=495

s5=241

#initial stratum standard deviation

Sh1 = sd(data$inventory[1:s1])

Sh2 = sd(data$inventory[(s1+1):(s1+s2)])

Sh3 = sd(data$inventory[(s1+s2+1):(s1+s2+s3)])

Sh4 = sd(data$inventory[(s1+s2+s3+1):(s1+s2+s3+s4)])

Sh5 = sd(data$inventory[(s1+s2+s3+s4+1):9760])

a=TRUE

i=0

while (a){

k=1/(1/Sh1+1/Sh2+1/Sh3+1/Sh4+1/Sh5)

s1 = round(k\*popSize/Sh1,0)

s2 = round(k\*popSize/Sh2,0)

s3 = round(k\*popSize/Sh3,0)

s4 = round(k\*popSize/Sh4,0)

s5 = round(k\*popSize/Sh5,0)

if((s1+s2+s3+s4+s5)<9760){

less=9760-(s1+s2+s3+s4+s5)

s1 = round((k\*popSize/Sh1)+less,0)

}

if((s1+s2+s3+s4+s5)>9760){

more=(s1+s2+s3+s4+s5)-9760

s1 = round((k\*popSize/Sh1)-more,0)

}

Sh1 = sd(data$inventory[1:s1])

Sh2 = sd(data$inventory[(s1+1):(s1+s2)])

Sh3 = sd(data$inventory[(s1+s2+1):(s1+s2+s3)])

Sh4 = sd(data$inventory[(s1+s2+s3+1):(s1+s2+s3+s4)])

Sh5 = sd(data$inventory[(s1+s2+s3+s4+1):9760])

i=i+1

k1=s1\*Sh1/popSize

k2=s2/popSize\*Sh2

k3=s3/popSize\*Sh3

k4=s4/popSize\*Sh4

k5=s5/popSize\*Sh5

if ( min(k2,k1)/max(k2,k1)>0.992 &

min(k2,k3)/max(k2,k3)>0.992 &

min(k3,k4)/max(k3,k4)>0.992 &

min(k4,k5)/max(k4,k5)>0.992){

print("finished in")

print(i)

print(s1)

print(s2)

print(s3)

print(s4)

print(s5)

print(s6)

print(s1+s2+s3+s4+s5)

print(Sh1)

print(Sh2)

print(Sh3)

print(Sh4)

print(Sh5)

a=FALSE

}

if(i==100000){

print("didn't find")

a=FALSE

}

}

print(k1)

print(k2)

print(k3)

print(k4)

print(k5)

#6

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

strata.cumrootf(data$inventory,n=500, Ls=6)

strata.LH(data$inventory,n=500, Ls=6)

#initial values for stratum size cum sqrt(f)

s1=8543

s2=893

s3=252

s4=67

s5=3

s6=2

#LH

s1=4603

s2=2351

s3=1464

s4=747

s5=353

s6=242

#initial stratum standard deviation

Sh1 = sd(data$inventory[1:s1])

Sh2 = sd(data$inventory[(s1+1):(s1+s2)])

Sh3 = sd(data$inventory[(s1+s2+1):(s1+s2+s3)])

Sh4 = sd(data$inventory[(s1+s2+s3+1):(s1+s2+s3+s4)])

Sh5 = sd(data$inventory[(s1+s2+s3+s4+1):(s1+s2+s3+s4+s5)])

Sh6 = sd(data$inventory[(s1+s2+s3+s4+s5+1):(s1+s2+s3+s4+s5+s6)])

disrepancy=0.42

a=TRUE

i=0

while (a){

k=1/(1/Sh1+1/Sh2+1/Sh3+1/Sh4+1/Sh5+1/Sh6)

s1 = round(k\*popSize/Sh1,0)

s2 = round(k\*popSize/Sh2,0)

s3 = round(k\*popSize/Sh3,0)

s4 = round(k\*popSize/Sh4,0)

s5 = round(k\*popSize/Sh5,0)

s6 = round(k\*popSize/Sh6,0)

if((s1+s2+s3+s4+s5+s6)<9760){

less=9760-(s1+s2+s3+s4+s5+s6)

s1 = round((k\*popSize/Sh1)+less,0)

}

if((s1+s2+s3+s4+s5+s6)>9760){

more=(s1+s2+s3+s4+s5+s6)-9760

s1 = round((k\*popSize/Sh1)-more,0)

}

Sh1 = sd(data$inventory[1:s1])

Sh2 = sd(data$inventory[(s1+1):(s1+s2)])

Sh3 = sd(data$inventory[(s1+s2+1):(s1+s2+s3)])

Sh4 = sd(data$inventory[(s1+s2+s3+1):(s1+s2+s3+s4)])

Sh5 = sd(data$inventory[(s1+s2+s3+s4+1):(s1+s2+s3+s4+s5)])

Sh6 = sd(data$inventory[(s1+s2+s3+s4+s5+1):9760])

i=i+1

k1=s1\*Sh1/popSize

k2=s2/popSize\*Sh2

k3=s3/popSize\*Sh3

k4=s4/popSize\*Sh4

k5=s5/popSize\*Sh5

k6=s6/popSize\*Sh6

if ( min(k2,k1)/max(k2,k1)>disrepancy &

min(k2,k3)/max(k2,k3)>disrepancy &

min(k3,k4)/max(k3,k4)>disrepancy &

min(k4,k5)/max(k4,k5)>disrepancy &

min(k5,k6)/max(k5,k6)>disrepancy){

print("finished in")

print(i)

print(s1)

print(s2)

print(s3)

print(s4)

print(s5)

print(s6)

print(s1+s2+s3+s4+s5+s6)

print(Sh1)

print(Sh2)

print(Sh3)

print(Sh4)

print(Sh5)

print(Sh6)

a=FALSE

}

if(i==100000){

print("didn't find")

a=FALSE

}

}

print(k1)

print(k2)

print(k3)

print(k4)

print(k5)

print(k6)

#7

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

strata.cumrootf(data$inventory,n=500, Ls=7)

strata.LH(data$inventory,n=500, Ls=7)

#initial values for stratum size

#cum root f

s1=8354

s2=777

s3=358

s4=204

s5=61

s6=4

s7=2

#LH method

s1=4556

s2=2135

s3=1116

s4=784

s5=597

s6=328

s7=244

#initial stratum standard deviation

Sh1 = sd(data$inventory[1:s1])

Sh2 = sd(data$inventory[(s1+1):(s1+s2)])

Sh3 = sd(data$inventory[(s1+s2+1):(s1+s2+s3)])

Sh4 = sd(data$inventory[(s1+s2+s3+1):(s1+s2+s3+s4)])

Sh5 = sd(data$inventory[(s1+s2+s3+s4+1):(s1+s2+s3+s4+s5)])

Sh6 = sd(data$inventory[(s1+s2+s3+s4+s5+1):(s1+s2+s3+s4+s5+s6)])

Sh7 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+1):9760])

disrepancy=0.38

a=TRUE

i=0

while (a){

k=1/(1/Sh1+1/Sh2+1/Sh3+1/Sh4+1/Sh5+1/Sh6+1/Sh7)

s1 = round(k\*popSize/Sh1,0)

s2 = round(k\*popSize/Sh2,0)

s3 = round(k\*popSize/Sh3,0)

s4 = round(k\*popSize/Sh4,0)

s5 = round(k\*popSize/Sh5,0)

s6 = round(k\*popSize/Sh6,0)

s7 = round(k\*popSize/Sh7,0)

if((s1+s2+s3+s4+s5+s6+s7)<9760){

less=9760-(s1+s2+s3+s4+s5+s6+s7)

s1 = round((k\*popSize/Sh1)+less,0)

}

if((s1+s2+s3+s4+s5+s6+s7)>9760){

more=(s1+s2+s3+s4+s5+s6+s7)-9760

s1 = round((k\*popSize/Sh1)-more,0)

}

Sh1 = sd(data$inventory[1:s1])

Sh2 = sd(data$inventory[(s1+1):(s1+s2)])

Sh3 = sd(data$inventory[(s1+s2+1):(s1+s2+s3)])

Sh4 = sd(data$inventory[(s1+s2+s3+1):(s1+s2+s3+s4)])

Sh5 = sd(data$inventory[(s1+s2+s3+s4+1):(s1+s2+s3+s4+s5)])

Sh6 = sd(data$inventory[(s1+s2+s3+s4+s5+1):(s1+s2+s3+s4+s5+s6)])

Sh7 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+1):9760])

i=i+1

k1=s1\*Sh1/popSize

k2=s2/popSize\*Sh2

k3=s3/popSize\*Sh3

k4=s4/popSize\*Sh4

k5=s5/popSize\*Sh5

k6=s6/popSize\*Sh6

k7=s7/popSize\*Sh7

if ( min(k2,k1)/max(k2,k1)>disrepancy &

min(k2,k3)/max(k2,k3)>disrepancy &

min(k3,k4)/max(k3,k4)>disrepancy &

min(k4,k5)/max(k4,k5)>disrepancy &

min(k5,k6)/max(k5,k6)>disrepancy &

min(k7,k6)/max(k7,k6)>disrepancy){

print("finished in")

print(i)

print(s1)

print(s2)

print(s3)

print(s4)

print(s5)

print(s6)

print(s7)

print(s1+s2+s3+s4+s5+s6+s7)

print(Sh1)

print(Sh2)

print(Sh3)

print(Sh4)

print(Sh5)

print(Sh6)

print(Sh7)

a=FALSE

}

if(i==100000){

print("didn't find")

a=FALSE

}

}

print(k1)

print(k2)

print(k3)

print(k4)

print(k5)

print(k6)

print(k7)

#8

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

strata.cumrootf(data$inventory,n=500, Ls=8) #didn't work because produced value 0 for 8th stratum

strata.LH(data$inventory,n=500, Ls=8)

#initial values for stratum size

#LH method

s1=1426

s2=3139

s3=2126

s4=1123

s5=775

s6=599

s7=324

s8=248

#initial stratum standard deviation

Sh1 = sd(data$inventory[1:s1])

Sh2 = sd(data$inventory[(s1+1):(s1+s2)])

Sh3 = sd(data$inventory[(s1+s2+1):(s1+s2+s3)])

Sh4 = sd(data$inventory[(s1+s2+s3+1):(s1+s2+s3+s4)])

Sh5 = sd(data$inventory[(s1+s2+s3+s4+1):(s1+s2+s3+s4+s5)])

Sh6 = sd(data$inventory[(s1+s2+s3+s4+s5+1):(s1+s2+s3+s4+s5+s6)])

Sh7 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+1):(s1+s2+s3+s4+s5+s6+s7)])

Sh8 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+s7+1):9760])

disrepancy=0.1

a=TRUE

i=0

while (a){

k=1/(1/Sh1+1/Sh2+1/Sh3+1/Sh4+1/Sh5+1/Sh6+1/Sh7+1/Sh8)

s1 = round(k\*popSize/Sh1,0)

s2 = round(k\*popSize/Sh2,0)

s3 = round(k\*popSize/Sh3,0)

s4 = round(k\*popSize/Sh4,0)

s5 = round(k\*popSize/Sh5,0)

s6 = round(k\*popSize/Sh6,0)

s7 = round(k\*popSize/Sh7,0)

s8 = round(k\*popSize/Sh8,0)

if((s1+s2+s3+s4+s5+s6+s7+s8)<9760){

less=9760-(s1+s2+s3+s4+s5+s6+s7+s8)

s1 = round((k\*popSize/Sh1)+less,0)

}

if((s1+s2+s3+s4+s5+s6+s7+s8)>9760){

more=(s1+s2+s3+s4+s5+s6+s7+s8)-9760

s1 = round((k\*popSize/Sh1)-more,0)

}

Sh1 = sd(data$inventory[1:s1])

Sh2 = sd(data$inventory[(s1+1):(s1+s2)])

Sh3 = sd(data$inventory[(s1+s2+1):(s1+s2+s3)])

Sh4 = sd(data$inventory[(s1+s2+s3+1):(s1+s2+s3+s4)])

Sh5 = sd(data$inventory[(s1+s2+s3+s4+1):(s1+s2+s3+s4+s5)])

Sh7 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+1):(s1+s2+s3+s4+s5+s6+s7)])

Sh8 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+s7+1):9760])

i=i+1

k1=s1\*Sh1/popSize

k2=s2/popSize\*Sh2

k3=s3/popSize\*Sh3

k4=s4/popSize\*Sh4

k5=s5/popSize\*Sh5

k6=s6/popSize\*Sh6

k7=s7/popSize\*Sh7

k8=s8/popSize\*Sh8

if ( min(k2,k1)/max(k2,k1)>disrepancy &

min(k2,k3)/max(k2,k3)>disrepancy &

min(k3,k4)/max(k3,k4)>disrepancy &

min(k4,k5)/max(k4,k5)>disrepancy &

min(k5,k6)/max(k5,k6)>disrepancy &

min(k7,k6)/max(k7,k6)>disrepancy &

min(k7,k8)/max(k7,k8)>disrepancy){

print("finished in")

print(i)

print(s1)

print(s2)

print(s3)

print(s4)

print(s5)

print(s6)

print(s7)

print(s8)

print(s1+s2+s3+s4+s5+s6+s7+s8)

print(Sh1)

print(Sh2)

print(Sh3)

print(Sh4)

print(Sh5)

print(Sh6)

print(Sh7)

print(Sh8)

a=FALSE

}

if(i==100000){

print("didn't find")

a=FALSE

}

}

print(k1)

print(k2)

print(k3)

print(k4)

print(k5)

print(k6)

print(k7)

print(k8)

#9

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

strata.cumrootf(data$inventory,n=500, Ls=9) #didn't work because produced value 0 for 8th stratum

strata.LH(data$inventory,n=500, Ls=9)

#initial values for stratum size

#LH method

s1=1174

s2=3163

s3=1089

s4=1455

s5=1029

s6=703

s7=575

s8=316

s9 =256

#initial stratum standard deviation

Sh1 = sd(data$inventory[1:s1])

Sh2 = sd(data$inventory[(s1+1):(s1+s2)])

Sh3 = sd(data$inventory[(s1+s2+1):(s1+s2+s3)])

Sh4 = sd(data$inventory[(s1+s2+s3+1):(s1+s2+s3+s4)])

Sh5 = sd(data$inventory[(s1+s2+s3+s4+1):(s1+s2+s3+s4+s5)])

Sh6 = sd(data$inventory[(s1+s2+s3+s4+s5+1):(s1+s2+s3+s4+s5+s6)])

Sh7 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+1):(s1+s2+s3+s4+s5+s6+s7)])

Sh8 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+s7+1):(s1+s2+s3+s4+s5+s6+s7+s8)])

Sh9 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+s7+s8+1):9760])

disrepancy = 0.91

a=TRUE

i=0

while (a){

k=1/(1/Sh1+1/Sh2+1/Sh3+1/Sh4+1/Sh5+1/Sh6+1/Sh7+1/Sh8 +1/Sh9)

s1 = round(k\*popSize/Sh1,0)

s2 = round(k\*popSize/Sh2,0)

s3 = round(k\*popSize/Sh3,0)

s4 = round(k\*popSize/Sh4,0)

s5 = round(k\*popSize/Sh5,0)

s6 = round(k\*popSize/Sh6,0)

s7 = round(k\*popSize/Sh7,0)

s8 = round(k\*popSize/Sh8,0)

s9 = round(k\*popSize/Sh9,0)

if((s1+s2+s3+s4+s5+s6+s7+s8+s9)<9760){

less=9760-(s1+s2+s3+s4+s5+s6+s7+s8+s9)

s1 = round((k\*popSize/Sh1)+less,0)

}

if((s1+s2+s3+s4+s5+s6+s7+s8+s9)>9760){

more=(s1+s2+s3+s4+s5+s6+s7+s8+s9)-9760

s1 = round((k\*popSize/Sh1)-more,0)

}

Sh1 = sd(data$inventory[1:s1])

Sh2 = sd(data$inventory[(s1+1):(s1+s2)])

Sh3 = sd(data$inventory[(s1+s2+1):(s1+s2+s3)])

Sh4 = sd(data$inventory[(s1+s2+s3+1):(s1+s2+s3+s4)])

Sh5 = sd(data$inventory[(s1+s2+s3+s4+1):(s1+s2+s3+s4+s5)])

Sh7 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+1):(s1+s2+s3+s4+s5+s6+s7)])

Sh8 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+s7+1):(s1+s2+s3+s4+s5+s6+s7+s8)])

Sh9 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+s7+s8+1):9760])

i=i+1

k1=s1/popSize\*Sh1

k2=s2/popSize\*Sh2

k3=s3/popSize\*Sh3

k4=s4/popSize\*Sh4

k5=s5/popSize\*Sh5

k6=s6/popSize\*Sh6

k7=s7/popSize\*Sh7

k8=s8/popSize\*Sh8

k9=s9/popSize\*Sh9

if ( min(k2,k1)/max(k2,k1)>disrepancy &

min(k2,k3)/max(k2,k3)>disrepancy &

min(k3,k4)/max(k3,k4)>disrepancy &

min(k4,k5)/max(k4,k5)>disrepancy &

min(k5,k6)/max(k5,k6)>disrepancy &

min(k7,k6)/max(k7,k6)>disrepancy &

min(k7,k8)/max(k7,k8)>disrepancy &

min(k9,k8)/max(k9,k8)>disrepancy){

print("finished in")

print(i)

print(s1)

print(s2)

print(s3)

print(s4)

print(s5)

print(s6)

print(s7)

print(s8)

print(s9)

print(s1+s2+s3+s4+s5+s6+s7+s8+s9)

print(Sh1)

print(Sh2)

print(Sh3)

print(Sh4)

print(Sh5)

print(Sh6)

print(Sh7)

print(Sh8)

print(Sh9)

a=FALSE

}

if(i==100000){

print("didn't find")

a=FALSE

}

}

print(k1)

print(k2)

print(k3)

print(k4)

print(k5)

print(k6)

print(k7)

print(k8)

print(k9)

#10

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

strata.cumrootf(data$inventory,n=500, Ls=10) #didn't work because produced value 0 for 9th stratum

strata.LH(data$inventory,n=500, Ls=10)

#initial values for stratum size

#LH method

s1=1165

s2=3165

s3=1072

s4=1397

s5=982

s6=724

s7=535

s8=297

s9 =180

s10 = 243

#initial stratum standard deviation

Sh1 = sd(data$inventory[1:s1])

Sh2 = sd(data$inventory[(s1+1):(s1+s2)])

Sh3 = sd(data$inventory[(s1+s2+1):(s1+s2+s3)])

Sh4 = sd(data$inventory[(s1+s2+s3+1):(s1+s2+s3+s4)])

Sh5 = sd(data$inventory[(s1+s2+s3+s4+1):(s1+s2+s3+s4+s5)])

Sh6 = sd(data$inventory[(s1+s2+s3+s4+s5+1):(s1+s2+s3+s4+s5+s6)])

Sh7 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+1):(s1+s2+s3+s4+s5+s6+s7)])

Sh8 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+s7+1):(s1+s2+s3+s4+s5+s6+s7+s8)])

Sh9 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+s7+s8+1):(s1+s2+s3+s4+s5+s6+s7+s8+s9)])

Sh10 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+s7+s8+s9+1):9760])

disrepancy = 0.7

a=TRUE

i=0

while (a){

k=1/(1/Sh1+1/Sh2+1/Sh3+1/Sh4+1/Sh5+1/Sh6+1/Sh7+1/Sh8 +1/Sh9 + 1/Sh10)

s1 = round(k\*popSize/Sh1,0)

s2 = round(k\*popSize/Sh2,0)

s3 = round(k\*popSize/Sh3,0)

s4 = round(k\*popSize/Sh4,0)

s5 = round(k\*popSize/Sh5,0)

s6 = round(k\*popSize/Sh6,0)

s7 = round(k\*popSize/Sh7,0)

s8 = round(k\*popSize/Sh8,0)

s9 = round(k\*popSize/Sh9,0)

s10 = round(k\*popSize/Sh10,0)

if((s1+s2+s3+s4+s5+s6+s7+s8+s9+s10)<9760){

less=9760-(s1+s2+s3+s4+s5+s6+s7+s8+s9+s10)

s1 = round((k\*popSize/Sh1)+less,0)

}

if((s1+s2+s3+s4+s5+s6+s7+s8+s9+s10)>9760){

more=(s1+s2+s3+s4+s5+s6+s7+s8+s9+s10)-9760

s1 = round((k\*popSize/Sh1)-more,0)

}

Sh1 = sd(data$inventory[1:s1])

Sh2 = sd(data$inventory[(s1+1):(s1+s2)])

Sh3 = sd(data$inventory[(s1+s2+1):(s1+s2+s3)])

Sh4 = sd(data$inventory[(s1+s2+s3+1):(s1+s2+s3+s4)])

Sh5 = sd(data$inventory[(s1+s2+s3+s4+1):(s1+s2+s3+s4+s5)])

Sh7 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+1):(s1+s2+s3+s4+s5+s6+s7)])

Sh8 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+s7+1):(s1+s2+s3+s4+s5+s6+s7+s8)])

Sh9 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+s7+s8+1):(s1+s2+s3+s4+s5+s6+s7+s8+s9)])

Sh10 = sd(data$inventory[(s1+s2+s3+s4+s5+s6+s7+s8+s9+1):9760])

i=i+1

k1=s1\*Sh1/popSize

k2=s2/popSize\*Sh2

k3=s3/popSize\*Sh3

k4=s4/popSize\*Sh4

k5=s5/popSize\*Sh5

k6=s6/popSize\*Sh6

k7=s7/popSize\*Sh7

k8=s8/popSize\*Sh8

k9=s9/popSize\*Sh9

k10=s10/popSize\*Sh10

if ( min(k2,k1)/max(k2,k1)>disrepancy &

min(k2,k3)/max(k2,k3)>disrepancy &

min(k3,k4)/max(k3,k4)>disrepancy &

min(k4,k5)/max(k4,k5)>disrepancy &

min(k5,k6)/max(k5,k6)>disrepancy &

min(k7,k6)/max(k7,k6)>disrepancy &

min(k7,k8)/max(k7,k8)>disrepancy &

min(k9,k8)/max(k9,k8)>disrepancy &

min(k9,k10)/max(k9,k10)>disrepancy){

print("finished in")

print(i)

print(s1)

print(s2)

print(s3)

print(s4)

print(s5)

print(s6)

print(s7)

print(s8)

print(s9)

print(s10)

print(s1+s2+s3+s4+s5+s6+s7+s8+s9+s10)

print(Sh1)

print(Sh2)

print(Sh3)

print(Sh4)

print(Sh5)

print(Sh6)

print(Sh7)

print(Sh8)

print(Sh9)

print(Sh10)

a=FALSE

}

if(i==100000){

print("didn't find")

a=FALSE

}

}

print(k1)

print(k2)

print(k3)

print(k4)

print(k5)

print(k6)

print(k7)

print(k8)

print(k9)

print(k10)

#### SAS Code for Stratified Inventory Data Analysis (Includes Sales Estimation using Inventory)

/\* MSDS 6370 - Statistical Sampling \*/

/\* Spring Project \*/

/\* Team - Ekaterina Pirogova, Hari Narayan Sanadhya \*/

/\* Drop the file if already exists \*/

%web\_drop\_table(WORK.industryData);

%web\_drop\_table(WORK.projectOutputSummary);

%web\_drop\_table(WORK.projectSummary);

%web\_drop\_table(WORK.projectOutSales);

/\* Path to the file \*/

/\* This data is generated using R code \*/

FILENAME datafile '/home/harisanadhya0/sasuser.v94/MSDS 6370/projectSpring2018/data.csv';

/\* Import the dataset \*/

data WORK.industryData;

infile datafile delimiter = ',' MISSOVER DSD firstobs=2 ;

informat

        coID best32.

        sales best32.

        inventory best32. ;

format

        coID best16.

        sales best16.

        inventory best16. ;

input coID sales inventory stratumWS stratumLH;

run;

/\* Print the first 10 records of the dataset to verify the data imported \*/

title "Raw Dataset (First 10 records)";

proc print data=work.industrydata(obs=10);

run;

/\* Generate the descriptive statistics \*/

title "Descriptive Statistics";

proc means data=work.industrydata n min q1 median q3 max mean std sum skewness kurtosis;

var sales inventory;

run;

/\* Worked on inventory variable so sorting by inventory \*/

proc sort data=work.industrydata out=work.industrydata;

by inventory;

run;

/\* Descriptive statistics for the inventory output shows the distribution \*/

/\* is right skewness and has a large tail (Kurtosis value)\*/

/\* Plot histogram and qqplot to view confirm the same \*/

title "Distribution of Sales and Inventory variables";

proc univariate data=work.industrydata noprint;

var sales inventory;

histogram sales inventory;

qqplot sales inventory/normal(mu=est sigma=est color=red l=2) square;

run;

/\* Get the population count, total, mean, minimum and maximum value for inventory variable \*/

%macro getStats();

    %global recordCount mininventoryValue maxinventoryValue meanPopInventory totalPopInventory meanPopSales totalPopSales;

proc sql noprint;

select count(inventory), min(inventory), max(inventory),avg(inventory), sum(inventory), avg(sales), sum(sales)

into :recordCount, :mininventoryValue, :maxinventoryValue, :meanPopInventory, :totalPopInventory, :meanPopSales, :totalPopSales

from work.industrydata;

quit;

%mend;

%getStats();

/\* Get the variance for each group and save them in the variable for equal WhSh method\*/

%macro findVar();

    %global stratumWS;

    proc sql noprint;

    select var(inventory) format 16.

    into :stratumWS separated by ','

    from work.industrydata group by stratumWS;

    run;

%mend;

%findVar();

/\* Print the variance of reach strata (comma separated) \*/

%put &stratumWS;

/\* Get the variance for each group and save them in the variable for LH method\*/

%macro findVar();

    %global stratumLH;

    proc sql noprint;

    select var(inventory) format 16.

    into :stratumLH separated by ','

    from work.industrydata group by stratumLH;

    run;

%mend;

%findVar();

/\* Print the variance of reach strata (comma separated) \*/

%put &stratumLH;

/\* View the distribution of the group - i.e. number of obs within each group \*/

proc freq data=work.industrydata(keep=stratumWS);

title "Frequency stats for equal WhSh method";

run;

proc freq data=work.industrydata(keep=stratumLH);

title "Frequency stats for LH method";

run;

/\* Seeds used \*/

%let seed1 = 306609547;

%let seed2 = 606868685;

%let seed3 = 621046942;

%let seed4 = 635888608;

%let seed5 = 252341977;

/\* Weights for SRS \*/

data work.industrydata;

set work.industrydata;

samplingWeight = &recordCount/500;

run;

/\* Macro running the sampling and then estimation process 5 times \*/

%macro srs\_sys\_Sampling(srs\_sys);

    %do i=1 %to 5;

        /\* Estimation of inventory using SRS \*/

        proc surveyselect data=work.industrydata sampsize=500 out=SRSSAMPLE method= &srs\_sys seed=&&seed&i;

        %let title = Sampling using &srs\_sys;

        title &title;

        run;

        proc surveymeans data=SRSSAMPLE total=&recordCount mean stderr clm sum std clsum;

        var inventory;

        weight samplingWeight;

        ods output Statistics = surveyMeansoutput;

        run;

        %let temp=%TSLIT(&srs\_sys);

        %put &temp;

        %if %sysfunc(exist(WORK.projectOutputSummary)) %then %do;

            proc sql noprint;

            insert into work.projectOutputSummary

            select 'Non-Stratified' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

            run;

        %end;

        %else %do;

            proc sql noprint;

            create table work.projectOutputSummary as

            select 'Non-Stratified' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

            run;

        %end;

    %end;

%mend;

%srs\_sys\_sampling(srs);

%srs\_sys\_sampling(sys);

/\* Drop the samplingWeights that was used for SRS \*/

data work.industrydata;

set work.industrydata;

drop samplingWeight;

run;

%macro srs\_sys\_sampling\_neyman(srs\_sys);

%local temp;

    %do i=1 %to 5;

        /\* Estimation of inventory using SRS \*/

        proc surveyselect data=work.industrydata sampsize=500 out=SRSSAMPLE method= &srs\_sys seed=&&seed&i;

        %let title = Equal WhSh method using &srs\_sys;

        strata stratumWS/ alloc=neyman var=(&stratumWS) allocmin=2;

        title &title;

        run;

        proc surveymeans data=SRSSAMPLE total=&recordCount mean stderr clm sum std clsum;

        var inventory;

        strata stratumWS;

        weight samplingWeight;

        ods output Statistics = surveyMeansoutput;

        run;

        %let temp=%TSLIT(&srs\_sys);

        %put &temp;

        %if %sysfunc(exist(WORK.projectOutputSummary)) %then %do;

            proc sql noprint;

            insert into work.projectOutputSummary

            select 'Equal WhSh method' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

            run;

        %end;

        %else %do;

            proc sql noprint;

            create table work.projectOutputSummary as

            select 'Equal WhSh method' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

            run;

        %end;

    %end;

%mend;

%srs\_sys\_sampling\_neyman(srs);

%srs\_sys\_sampling\_neyman(sys);

%macro srs\_sys\_sampling\_neyman\_LH(srs\_sys);

%local temp;

    %do i=1 %to 5;

        /\* Estimation of Inventory using SRS \*/

        proc surveyselect data=work.industrydata sampsize=500 out=SRSSAMPLE method= &srs\_sys seed=&&seed&i;

        %let title = Neyman Sampling using &srs\_sys - Stratified using LH method;

        strata stratumLH/ alloc=neyman var=(&stratumLH) allocmin=2;

        title &title;

        run;

        proc surveymeans data=SRSSAMPLE total=&recordCount mean stderr clm sum std clsum;

        var inventory;

        strata stratumLH;

        weight samplingWeight;

        ods output Statistics = surveyMeansoutput;

        run;

        %let temp=%TSLIT(&srs\_sys);

        %put &temp;

        %if %sysfunc(exist(WORK.projectOutputSummary)) %then %do;

            proc sql noprint;

            insert into work.projectOutputSummary

            select 'LH method Stratified Neyman' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

            run;

        %end;

        %else %do;

            proc sql noprint;

            create table work.projectOutputSummary as

            select 'LH method Stratified Neyman' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

            run;

        %end;

    %end;

%mend;

%srs\_sys\_sampling\_neyman\_LH(srs);

%srs\_sys\_sampling\_neyman\_LH(sys);

/\* Since all the algorithms were executed 5 times - Compute the average estimated value of mean and sum of inventory \*/

ods exclude all;

proc means mean data=work.projectoutputsummary(keep=sum mean sampling method);

by sampling method notsorted;

ods output summary=projectSummary;

run;

ods select all;

/\* Combine the sampling and method columns in the projectoutputsummary dataset to create the samplingMethod column \*/

proc sql noprint;

alter table work.projectoutputsummary add samplingMethod varchar(50);

update work.projectoutputsummary set samplingMethod= cat(strip(sampling),cat(" ", strip(method)));

run;

proc print data=work.projectoutputsummary;

title "Estimation Data generated using all the algorithms that were applied";

run;

/\* Plot the estimated total inventory value obtained by each algorithm for every iteration \*/

proc sgplot data=work.projectoutputsummary des="Plot of Estimated Total inventory";

series x = iteration y = sum /group=samplingMethod markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=2)

    name="Legend2" legendlabel="Sampling Method";;

refline &totalPopInventory/AXIS=y legendlabel="Population Total" name="Legend1" label;

keylegend "Legend1"/location=inside position=topright;

keylegend "Legend2"/location=outside position=bottom title="Sampling Method";

title "Plot of Estimated Total Inventory";

run;

proc print data=work.projectoutputsummary;

run;

/\* Plot the estimated average inventory value obtained by each algorithm for every iteration \*/

proc sgplot data=work.projectoutputsummary des="Plot of Estimated Mean inventory";

refline &meanPopInventory/AXIS=y legendlabel="Population Mean" name="Legend1" label;

series x = iteration y = mean / group=samplingMethod markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=2)

    name="Legend2" legendlabel="Sampling Method";

keylegend "Legend1"/location=inside position=topright;

keylegend "Legend2"/location=outside position=bottom title="Sampling Method";

title "Plot of Estimated Mean inventory";

run;

/\* Above plots showed that of all the algorithms applied, Non-stratified design gave the worst predictions \*/

/\* Remove the Non-stratified design predictions and replot the data for more clear visualization of the estimations \*/

proc sql noprint;

delete from work.projectoutputsummary where sampling= 'Non-Stratified';

run;

/\* Plot the estimated total inventory value obtained by each algorithm for every iteration \*/

proc sgplot data=work.projectoutputsummary des="Plot of Estimated Total inventory";

series x = iteration y = sum /group=samplingMethod markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=2)

    name="Legend2" legendlabel="Sampling Method";;

refline &totalPopInventory/AXIS=y legendlabel="Population Total" name="Legend1" label;

keylegend "Legend1"/location=inside position=topright;

keylegend "Legend2"/location=outside position=bottom title="Sampling Method";

title "Plot of Estimated Total inventory";

run;

/\* Plot the estimated mean inventory value obtained by each algorithm for every iteration \*/

proc sgplot data=work.projectoutputsummary des="Plot of Estimated Mean inventory";

refline &meanPopInventory/AXIS=y legendlabel="Population Mean" name="Legend1" label;

series x = iteration y = mean / group=samplingMethod markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=2)

    name="Legend2" legendlabel="Sampling Method";

keylegend "Legend1"/location=inside position=topright;

keylegend "Legend2"/location=outside position=bottom title="Sampling Method";

title "Plot of Estimated Mean Inventory";

run;

/\* Combine the sampling and method columns in the projectsummary dataset to create the samplingMethod column \*/

/\* This dataset has the average estimation received from all the iterations for a given algorithm \*/

proc sql noprint;

alter table work.projectsummary add samplingMethod varchar(50);

update work.projectsummary set samplingMethod= cat(strip(sampling),cat(" ", strip(method)));

run;

/\* Plot the estimated mean inventory value obtained by each algorithm averaged over the iterations \*/

proc sgplot data=work.projectsummary des="Plot of Estimated Mean Inventory - Summarized Info";

scatter x = samplingMethod y = Mean\_Mean;

refline &meanPopInventory/AXIS=y legendlabel="Population Mean" name="Legend1" label;

title "Plot of Estimated Mean inventory - Average statistics";

run;

/\* Plot the estimated total inventory value obtained by each algorithm averaged over the iterations \*/

proc sgplot data=work.projectsummary des="Plot of Estimated Total inventory - Summarized Info";

scatter x = samplingMethod y = Sum\_Mean;

refline &totalPopInventory/AXIS=y legendlabel="Population Total" name="Legend1" label;

title "Plot of Estimated Total inventory - Average Statistics";

run;

/\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*sales variable calculation with inventory stratification\*/

%macro srs\_sys\_sampling\_neyman(srs\_sys);

%local temp;

    %do i=1 %to 5;

        /\* Estimation of sales using SRS \*/

        proc surveyselect data=work.industrydata sampsize=500 out=SRSSAMPLE method= &srs\_sys seed=&&seed&i;

        %let title = Equal WhSh method using &srs\_sys;

        strata stratumWS/ alloc=neyman var=(&stratumWS) allocmin=2;

        title &title;

        run;

        proc surveymeans data=SRSSAMPLE total=&recordCount mean stderr clm sum std clsum;

        var sales;

        strata stratumWS;

        weight samplingWeight;

        ods output Statistics = surveyMeansoutput;

        run;

        %let temp=%TSLIT(&srs\_sys);

        %put &temp;

        %if %sysfunc(exist(WORK.projectOutSales)) %then %do;

            proc sql noprint;

            insert into work.projectOutSales

            select 'Equal WhSh method' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

            run;

        %end;

        %else %do;

            proc sql noprint;

            create table work.projectOutSales as

            select 'Equal WhSh method' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

            run;

        %end;

    %end;

%mend;

%srs\_sys\_sampling\_neyman(srs);

%srs\_sys\_sampling\_neyman(sys);

%macro srs\_sys\_sampling\_neyman\_LH(srs\_sys);

%local temp;

    %do i=1 %to 5;

        /\* Estimation of sales using SRS \*/

        proc surveyselect data=work.industrydata sampsize=500 out=SRSSAMPLE method= &srs\_sys seed=&&seed&i;

        %let title = Neyman Sampling using &srs\_sys - Stratified using LH method;

        strata stratumLH/ alloc=neyman var=(&stratumLH) allocmin=2;

        title &title;

        run;

        proc surveymeans data=SRSSAMPLE total=&recordCount mean stderr clm sum std clsum;

        var sales;

        strata stratumLH;

        weight samplingWeight;

        ods output Statistics = surveyMeansoutput;

        run;

        %let temp=%TSLIT(&srs\_sys);

        %put &temp;

        %if %sysfunc(exist(WORK.projectOutSales)) %then %do;

            proc sql noprint;

            insert into work.projectOutSales

            select 'LH method Stratified Neyman' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

            run;

        %end;

        %else %do;

            proc sql noprint;

            create table work.projectOutSales as

            select 'LH method Stratified Neyman' as sampling, &temp as method, &i as iteration, outtable.\* from surveyMeansoutput as outtable;

            run;

        %end;

    %end;

%mend;

%srs\_sys\_sampling\_neyman\_LH(srs);

%srs\_sys\_sampling\_neyman\_LH(sys);

ods exclude all;

proc means mean data=work.projectOutSales(keep=sum mean sampling method);

by sampling method notsorted;

ods output summary=projectSummary;

run;

ods select all;

/\* Combine the sampling and method columns in the Sales dataset to create the samplingMethod column \*/

proc sql noprint;

alter table work.projectOutSales add samplingMethod varchar(50);

update work.projectOutSales set samplingMethod= cat(strip(sampling),cat(" ", strip(method)));

run;

proc print data=work.projectOutSales;

title "Estimation Data generated using all the algorithms that were applied";

run;

/\* Plot the estimated average Sales value obtained by each algorithm for every iteration \*/

proc sgplot data=work.projectOutSales des="Plot of Estimated Mean Sales";

refline &meanPopSales/AXIS=y legendlabel="Population Mean" name="Legend1" label;

series x = iteration y = mean / group=samplingMethod markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=2)

    name="Legend2" legendlabel="Sampling Method";

keylegend "Legend1"/location=inside position=topright;

keylegend "Legend2"/location=outside position=bottom title="Sampling Method";

title "Plot of Estimated Mean Sales";

run;

/\* Above plots showed that of all the algorithms applied, Non-stratified design gave the worst predictions \*/

/\* Remove the Non-stratified design predictions and replot the data for more clear visualization of the estimations \*/

proc sql noprint;

delete from work.projectOutSales where sampling= 'Non-Stratified';

run;

/\* Plot the estimated total Sales value obtained by each algorithm for every iteration \*/

proc sgplot data=work.projectOutSales des="Plot of Estimated Total Sales";

series x = iteration y = sum /group=samplingMethod markers markerattrs=(symbol=circlefilled) lineattrs=(pattern=2)

    name="Legend2" legendlabel="Sampling Method";;

refline &totalPopSales/AXIS=y legendlabel="Population Total" name="Legend1" label;

keylegend "Legend1"/location=inside position=topright;

keylegend "Legend2"/location=outside position=bottom title="Sampling Method";

title "Plot of Estimated Total sales";

run;

/\* Combine the sampling and method columns in the Sales dataset to create the samplingMethod column \*/

/\* This dataset has the average estimation received from all the iterations for a given algorithm \*/

proc sql noprint;

alter table work.projectOutSales add samplingMethod varchar(50);

update work.projectOutSales set samplingMethod= cat(strip(sampling),cat(" ", strip(method)));

run;