OPTIMIZED STRATIFICATION PROJECT – (7/22/18)

Dr. Jack K. Rasmus-Vorrath

Section 1.Descriptive Statistics

The variable of interest is ‘INVENTORY’.

The **total number of units in the population** is 9762.

```
> nrow(df)
[1] 9762
```

The **true population total** is 1754954823.

```
> sum(df$inventory)
[1] 1754954823
```

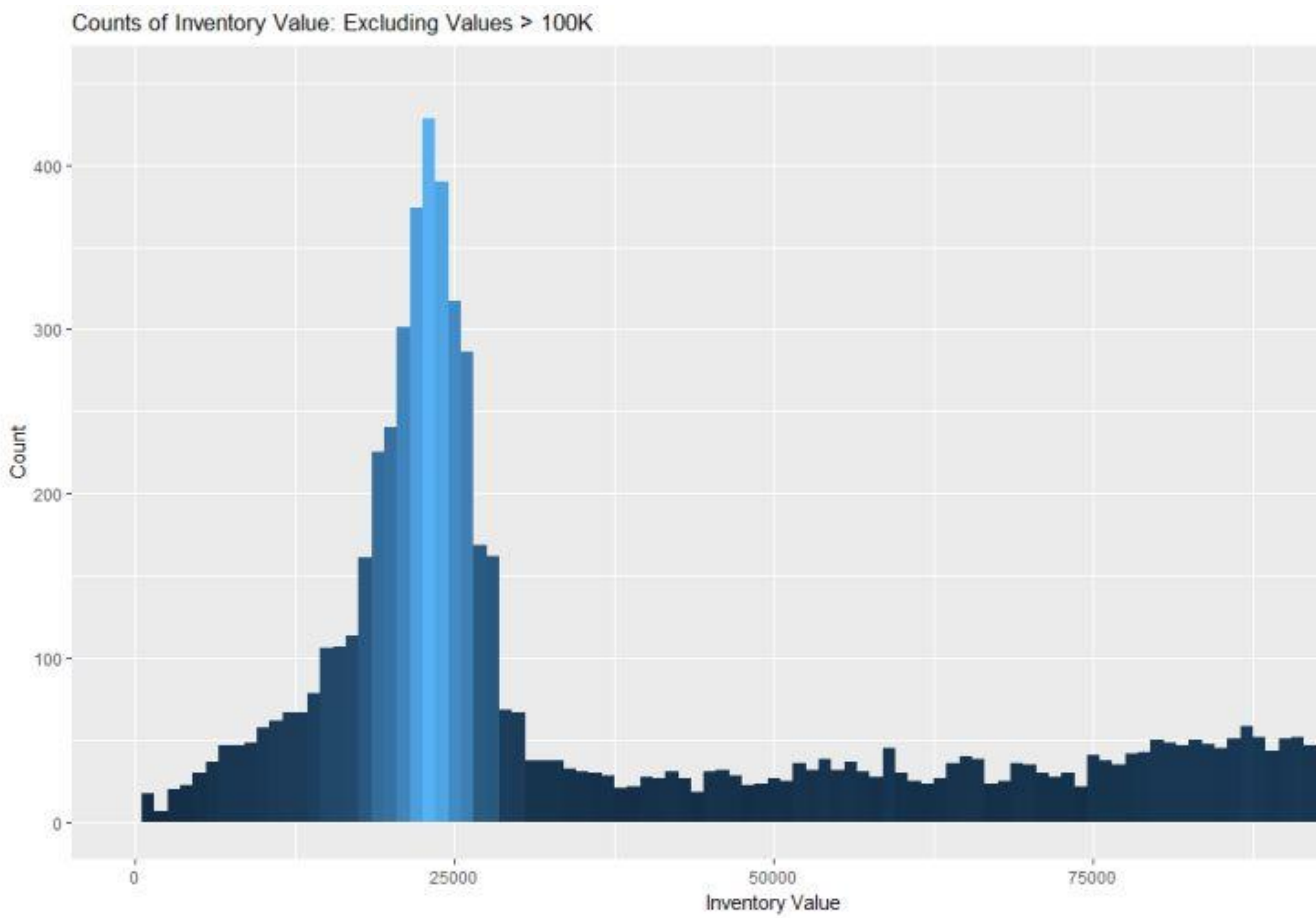
The **true population standard deviation** is 1573358.

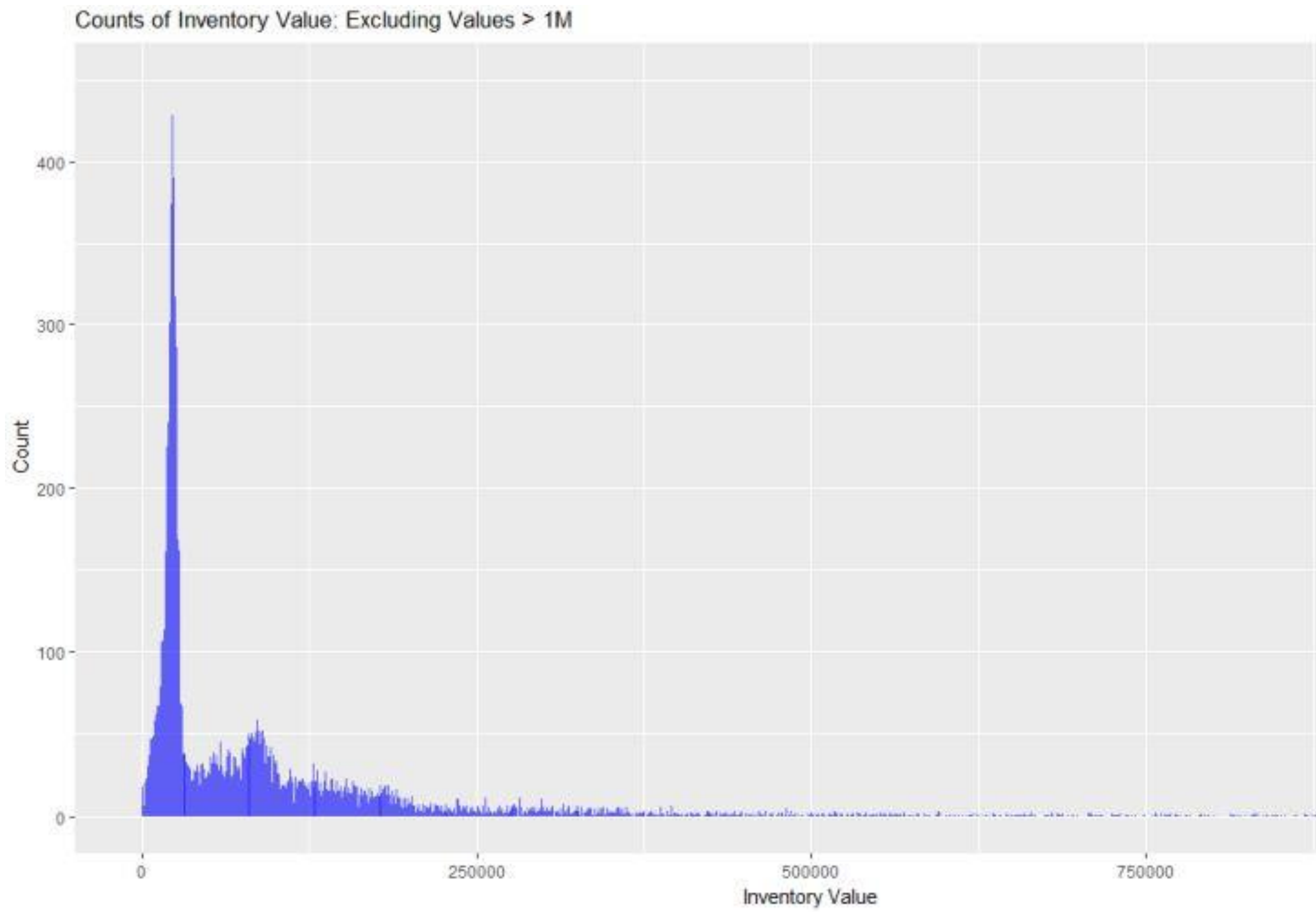
```
> sd(df$inventory)
[1] 1573358
```

The **true population mean** is 179774.1.

```
> mean(df$inventory)
[1] 179774.1
```

Histograms of the variable of interest are shown below:


Counts of Inventory Value: Excluding Values > 100K
Count
400
300
200
100
0
0
25000
50000
75000
Inventory Value



Section 2. Stratification Using Sales(Inventory)

2.1 Description of the process followed in forming sampling strata for the population using the Inventory variable:

- Description of the number and range of values chosen for the certainty stratum and the criteria used for including units in the certainty stratum.

  Particularly when dealing with severely right-skewed data, best analytic practices suggest **assigning the highest-valued records to the certainty stratum** to minimize sampling variance. Cf. https://enablement.acl.com/helpdocs/analytics/13/user-guide/en-us/Content/da_sampling_data/classical_variables_sampling.htm#navlink-5.

  Deciding how many of the highest-valued records to assign was determined according to a custom-made optimization wrapper function designed to minimize the overall coefficient of variation (relative root mean squared error) produced when applying the generalized Lavallee-Hidiroglou stratification method for skewed populations.

Cf. **Lavallée, P. and Hidiroglou, M. (1988). On the stratification of skewed populations, Survey Methodology, 14, 33-43.**

The Lavallee-Hidiroglou stratification method is implemented using the strat.LH() function from the 'Stratification' R package. The function implements Kozak's Random Search method.
Cf. **Kozak, M. (2004). Optimal Stratification Using Random Search Method in Agricultural Surveys, Statistics in Transition, 6, 5, 797-806.**

For an explanation of Lavallee-Hidiroglou stratification and Kozak's Random Search method,
Cf. **Sebnem, Er. (2011). Computational Methods for Optimum Stratification: A Review, Int. Statistical Inst.: Proc. 58th World Statistical Congress, 2011, Dublin (Session STS058).**

As shown in Sebnem (2011), the Lavallee-Hidiroglou stratification method works as follows:

Lavallée & Hidiroglou tries to find such values (Lavallée & Hidiroglou (1988), pp.36)

$$b_1 < b_2 < \cdots < b_{L-1} \tag{18}$$

that minimizes n considering a take-all top stratum,

$$n = N_L + \left(\sum_{h=1}^{L-1} N_h^2\sigma_h^2/a_h\right)\left(N^2\mu^2CV^2 + \sum_{h=1}^{L-1} N_h{\sigma_h}^2\right)^{-1}. \tag{19}$$

given the level of precision (CV) and the specified allocation scheme represented by a_h (Lavall
Hidiroglou (1988), pp.36).
In their paper Lavallée & Hidiroglou (1988) mainly consider X-proportional power allocation:

$$a_h = \frac{(W_h\mu_h)^p}{\sum_{h=1}^{L-1}(W_h\mu_h)^p} \tag{20}$$

where $0 < p < \infty$. Putting (20) in (19)

$$n = NW_L + \frac{N\left[\sum_{h=1}^{L-1}(W_h\sigma_h)^2(W_h\mu_h)^p\right]\left[\sum_{h=1}^{L-1}(W_h\mu_h)^p\right]}{NCV^2\mu^2 + \sum_{h=1}^{L-1} W_h\sigma_h^2}. \tag{21}$$

As shown in Sebnem (2011), Kozak's Random Search method works as follows:

Kozak's (2004) Random Search Method tries to find such values

$$b_1 < b_2 < \cdots < b_{L-1} \tag{22}$$

that minimizes the objective function considering a take-all top stratum with Neyman allocation

$$n = N_L + \left(\sum_{h=1}^{L-1} W_h \sigma_h\right)^2 \left(\mu^2 CV^2 + \frac{1}{N}\sum_{h=1}^{L-1} W_h {\sigma_h}^2\right)^{-1}; \tag{23}$$

under the constraints $N_h \geq 2$ for $h = 1, 2, \ldots, L$ and $2 \leq n_h \leq N_h$ for $h = 1, 2, \ldots, L-1$, whe
is the minimizing sample size required to achieve the given precision CV of the mean estimate o
strafication when the strata boundaries are $\mathbf{b} = (b_1, b_2, \ldots, b_{L-1})^T$.
Kozak's (2004) random search method chooses an initial set of strata boundaries and calculates
function values of n. Then for a certain number of iterations the following steps are repeated:

1. Generate the set of boundaries $\mathbf{b}'$ by choosing one stratum boundary b_h and changing
follows:

$$b_h' = b_h + j \tag{24}$$

where j is the random integer, $j \in \langle -p; -1\rangle \cup \langle 1; p\rangle$ and p is a given integer according to the
of the population.

2. Calculate the function value of n'.

3. If the constraints are satisfied and $n' < n$, the new set of boundaries b_h' are accepted.

The custom wrapper function (written in R) iterated through several sizes of highest-valued certainty strata before arriving at an optimum overall coefficient of variation (CV) value of **0.003779453**, using **110 records** with the highest inventory value. The values of these 110 records allocated to the certainty stratum **ranged from 1372514.04 to the maximum 105379553.91**.

- Description of the process for deciding on the number of non-certainty strata.

The same custom wrapper function served to jointly optimize the number of non-certainty strata, arriving at the number **10**.

- Table showing a row for each stratum, with the stratum number, number of units, and range of values.

The optimal configuration of highest-valued records and number of non-certainty strata is shown below, with stratum numbers, number of units (Nh) and their respective ranges of values (bh):

```
> print(Optim_Config)
Given arguments:
x = df$inventory
n = 500, Ls = 10, takenone = 0, takeall = 0
allocation: q1 = 0.5, q2 = 0, q3 = 0.5
model = none
algo = Kozak: minsol = 1000, idopti = nh, minNh = 2, maxiter = 10000,
              maxstep = 100, maxstill = 500, rep = 5, trymany = TRUE

Strata information:
           |      type rh |          bh        E(Y)     Var(Y)   Nh  nh   fh
stratum 1  | take-some  1 |    36703.82    21229.61   41103797 4346  79 0.02
stratum 2  | take-some  1 |    71986.24    55039.89   99472460 1054  30 0.03
stratum 3  | take-some  1 |   106848.40    88032.43   79086957 1334  33 0.02
stratum 4  | take-some  1 |   151201.78   128117.76  158852979  852  30 0.04
stratum 5  | take-some  1 |   210748.83   176507.24  257069766  730  33 0.05
stratum 6  | take-some  1 |   295186.09   251081.23  612253308  416  29 0.07
stratum 7  | take-some  1 |   413551.11   343819.24 1063090418  361  33 0.09
stratum 8  | take-some  1 |   614205.40   499757.45 3131901977  248  39 0.16
stratum 9  | take-some  1 |   974662.93   756680.65 8825392705  176  47 0.27
stratum 10 | take-some  1 | 105379553.90 1193526.37 9217654279  135  37 0.27
           |   certain  1 |           - 5476606.91          -  110 110 1.00
Total                                                        9762 500 0.05

Total sample size: 500
Anticipated population mean: 179774.1
Anticipated CV: 0.003779453
Note: CV=RRMSE (Relative Root Mean Squared Error) because takenone=0.
```

2.2 Neyman allocation of units to a stratified sample of size 500.

The Neyman allocation of sample strata followed the procedure shown below. The sample sizes come from the ‘nh’ vector of the ‘Optim_Config’ object returned by the stratification::strata.LH() function. The size of the certainty stratum is appended before performing the sampling procedure.

```
> #adjusted Neyman Allocation, assuming values provided by the LH optimizatio
n
> Ney_Alloc_3 <- append(Optim_Config$nh, 110)
>
> #Set Random Sampling Seed
> set.seed(232)
>
> #Sampling using Neyman Allocation
> Ney_Samp_12 <- sampling::strata(df, stratanames='Stratum', size=Ney_Alloc_3
, method="srswor")
>
> #Drop duplicated column
> Ney_Samp_12 <- Ney_Samp_12[, !duplicated(colnames(Ney_Samp_12))]
>
> #Rename ID column to match original data frame, for merging
> colnames(Ney_Samp_12)[2] <- "coID"
> Ney_Samp_12_merged <- merge(x = df, y = Ney_Samp_12)
>
> #sort by Stratum
```

```
> Ney_Samp_12_merged <- Ney_Samp_12_merged[order(Ney_Samp_12_merged$Stratum),
]
```

2.3 Selection of 5 stratified samples using Neyman allocation.

After preparing the population data frame (as shown below), the procedure above was used to select the 5 samples used by the analysis.

```
> #Preparing the data frame###################################################
###################################################
>
> #Identify boundary separating highest-valued certainty stratum from other s
trata
> options(digits=15)
> min(df[order(df$inventory, decreasing=T)[1:110],]$inventory)
[1] 1372514.035
> #Returns 1372514.035
>
> #Assign Strata
> df$Stratum <- NA
> df$Stratum[df$`inventory` < 36703.82] <- 1
> df$Stratum[(df$`inventory` < 71655.62) & (df$`inventory` >= 36703.82)] <- 2
> df$Stratum[(df$`inventory` < 106848.40) & (df$`inventory` >= 71655.62)] <-
3
> df$Stratum[(df$`inventory` < 151104.80) & (df$`inventory` >= 106848.40)] <-
4
> df$Stratum[(df$`inventory` < 209954.79) & (df$`inventory` >= 151104.80)] <-
5
> df$Stratum[(df$`inventory` < 295186.09) & (df$`inventory` >= 209954.79)] <-
6
> df$Stratum[(df$`inventory` < 413551.11) & (df$`inventory` >= 295186.09)] <-
7
> df$Stratum[(df$`inventory` < 614205.40) & (df$`inventory` >= 413551.11)] <-
8
> df$Stratum[(df$`inventory` < 974662.93) & (df$`inventory` >= 614205.40)] <-
9
> df$Stratum[(df$`inventory` < 1372514.04) & (df$`inventory` >= 974662.93)] <
- 10
> df$Stratum[(df$`inventory` < 105379553.91) & (df$`inventory` >= 1372514.04)
] <- 11
>
> #Check if any NAs in Stratum column
> any(is.na(df$Stratum))
[1] FALSE
>
> #Assign column for Finite Population Correction
> df$Num <- NA
> df$Num[df$Stratum == 1] <- 4346
> df$Num[df$Stratum == 2] <- 1047
> df$Num[df$Stratum == 3] <- 1341
> df$Num[df$Stratum == 4] <- 851
> df$Num[df$Stratum == 5] <- 723
> df$Num[df$Stratum == 6] <- 424
> df$Num[df$Stratum == 7] <- 361
```

```
> df$Num[df$Stratum == 8] <- 248
> df$Num[df$Stratum == 9] <- 176
> df$Num[df$Stratum == 10] <- 135
> df$Num[df$Stratum == 11] <- 110
```

2.4 Estimations of Inventory for each sample: the population mean, the standard error of the mean, the 95% confidence interval for the mean, the population total, the standard error for the total, and the 95% confidence interval for the total.

	Mean_Estimate	Mean_SE	Mean_95%_CI	Includes_True_Mean	Total_Estimate	Total_SE	Total_95%_C
Stratified_Test 1	169756.5	3452.98	[162988.75, 176524.17]	NO	1657162518	33707961	[1591096127.57, 17232
Stratified_Test 2	171029.2	3328.93	[164504.66, 177553.81]	NO	1669587375	32496969	[1605894485.29, 17332
Stratified_Test 3	170941.7	3432.96	[164213.25, 177670.19]	NO	1668733078	33512528	[1603049730.45, 17344
Stratified_Test 4	180661.4	4638.57	[171569.94, 189752.80]	YES	1763616255	45281715	[1674865723.84, 18523
Stratified_Test 5	188972.1	4634.39	[179888.87, 198055.35]	NO	1844745764	45240921	[1756075189.22, 19334
Stratified_No_Lonely_PSU 1	185135.8	4675.96	[175971.11, 194300.55]	YES	1807295958	45646764	[1717829943.82, 18967
Stratified_No_Lonely_PSU 2	179848.5	4643.22	[170747.97, 188949.05]	YES	1755681140	45327086	[1666841683.02, 18445
Stratified_No_Lonely_PSU 3	184075.6	4733.82	[174797.44, 193353.68]	YES	1796945611	46211566	[1706372606.51, 18875
Stratified_No_Lonely_PSU 4	182583.6	5040.98	[172703.45, 192463.72]	YES	1782380940	49210021	[1685931070.09, 18788
Stratified_No_Lonely_PSU 5	170758.0	3455.04	[163986.30, 177529.80]	NO	1666940099	33728064	[1600834307.76. 17330
SRS	314438.5	203292.38	[-84007.23, 712884.27]	YES*	157219259	101646191	[-42003614.45, 35644
Stratified_LH-Optimized 1	173299.0	7965.95	[157686.07, 188912.04]	YES	1691745349	77763647	[1539331402.05, 18441
Stratified_LH-Optimized 2	170343.7	7660.28	[155329.87, 185357.61]	YES	1662895555	74779635	[1516330162.61, 18094
Stratified_LH-Optimized 3	175205.2	7810.02	[159897.81, 190512.53]	YES	1710352871	76241440	[1560922394.72, 18597
Stratified_LH-Optimized 4	174342.7	8034.61	[158595.18, 190090.28]	YES	1701933736	78433884	[1548206147.84, 18556
Stratified_LH-Optimized 5	170959.9	7684.90	[155897.78, 186022.04]	YES	1668910633	75020020	[1521874096.22, 18159

A table showing the results of the analysis is shown above. The first 5 rows show the results of a stratified sampling procedure which used a 'lonely PSU' adjustment, allocating a single record to the highest-valued certainty stratum. The second set of 5 rows show the results of a stratified sampling procedure which did not employ a 'lonely PSU' adjustment, allocating two records to the highest-valued certainty stratum. The 11th row shows the results of an unstratified simple random sampling (SRS) procedure. Altogether, the first 11 rows represent a performance comparison to the results output by a stratified sampling design that employs the generalized Lavallee-Hidiroglou stratification method for skewed populations, allocating the 110 highest-valued records to the certainty stratum as determined by the optimization procedure. Although this strategy did result in broader confidence intervals and larger values of standard error (incurred by including more of the highly variant higher-valued records in the certainty stratum), it also resulted in a more accurate estimation, successfully including the population mean and total 100% of the time.

Using the generalized Lavallee-Hidiroglou stratification method for 5 sampling procedures, the mean of the estimate of mean, its standard error, as well as the mean of the estimate of total and its standard error are shown below:

```
> #Mean of the Total and Mean Estimates and their Standard Errors (across Samples assuming the Sizes allocated by LH Optimization)
>
> m.est_of_mean <- mean(173299.05, 170343.74, 175205.17, 174342.73, 170959.91)
> m.est_of_mean
[1] 173299.05
> m.est_of_mean_SE <- mean(7965.95, 7660.28, 7810.02, 8034.61, 7684.90)
> m.est_of_mean_SE
[1] 7965.95
> m.est_of_total <- mean(1691745349.49, 1662895554.54, 1710352871.10, 1701933735.66, 1668910633.11)
> m.est_of_total
[1] 1691745349.49
> m.est_of_total_SE <- mean(77763647.01, 74779635.28, 76241439.93, 78433884.00, 75020019.78)
> m.est_of_total_SE
[1] 77763647.01
```

2.5. Estimates of mean and total compared to the true values of mean and total.

When using the 'lonely PSU' adjustment (allocating a single record to the highest-valued certainty stratum), the confidence intervals of the estimations included the true values only 20% of the time.

When not using the 'lonely PSU' adjustment (allocating two records to the highest-valued certainty stratum), the confidence intervals of the estimations included the true values 80% of the time.

When applying optimized stratification using the generalized Lavallee-Hidiroglou method, (allocating 110 records to the highest-valued certainty stratum), the confidence intervals of the estimations included the true values 100% of the time.

2.6 Results of performing the analysis without stratification.

When no stratification procedure was applied, and a simple random sampling (SRS) of the population data was used, the confidence intervals of the estimations failed entirely to include the true values, capturing it 0%* of the time. More precisely, the true mean was included by a completely uninformative confidence interval, ranging from [-84007.23, 712884.27]—when corrected for nonsensical values, [0, 712884.27]. Despite the similarly enormous confidence interval of the SRS estimate of total ([-42003614.45, 356442133.11], or [0, 356442133.11], it still failed to capture the true value.

Section 4. Conclusion

Stratification can greatly enhance the precision of one's estimates. Dealing with highly skewed data presents particular challenges when determining the optimal partitioning of strata. Applications of the Lavallee-Hidiroglou method outperformed the geometric method when minimizing the overall coefficient of variation (relative root mean squared error) to obtain the most precise estimates. Wrapping the L-H function in a custom-written optimization function facilitated the process of identifying the most performant configuration of parameters (including the number of records to include in the certainty stratum, and the number of strata to create). Although including the highest-valued records of this strongly right-skewed data in the certainty stratum resulted in larger standard error and broader confidence intervals than were obtained when limiting the number of these highly variant primary sampling units (PSUs), it also resulted in better estimations, capturing the true values of mean and total 100% of the time.

References

Baillargeon S. and Rivest, Louis-Paul (2011). The Construction of Stratified Designs in R with the Package Stratification. Survey Methodology. Statistics Canada, Catalogue No. 12-001-X. Vol. 37, No. 1, pp. 53-65.

Ballin, M. and Catanese, E. "Stratificaion in Business and Agricultural Surveys with R", in New Challenges for Statistical Software - The Use of R in Official Statistics. 4th International Conference. Bucharest, Romania, 2016.

Gunning, P. and Horgan, J. (2004). A New Algorithm for the Construction of Stratum Boundaries in Skewed Populations. Survey Methodology. Statistics Canada, Catalogue No. 12-001. Vol. 30, No.2, pp. 159-166.

Kozak, M. (2004). Optimal Stratification Using Random Search Method in Agricultural Surveys, Statistics in Transition, 6, 5, 797-806.

Lavallée, P. and Hidiroglou, M. (1988). On the stratification of skewed populations, Survey Methodology, 14, 33-43.

Lisic, J., Sang, H., Zhu, Z., and Zimmer, S. (2018). Optimal Stratification and Allocation for the June Agricultural Survey. Journal of Official Statistics, Vol. 34, No. 1, pp. 121-148.

Sebnem, E. (2011). Computational Methods for Optimum Stratification: A Review, Int. Statistical Inst.: Proc. 58th World Statistical Congress, 2011, Dublin (Session STS058).

"Classical variables sampling", Enablement.acl.com, 2018. [Online]. Available: https://enablement.acl.com/helpdocs/analytics/13/user-guide/en-us/Content/da_sampling_data/classical_variables_sampling.htm#navlink-5. [Accessed: 22- Jul- 2018].

"Sampling methods applied to fisheries science: a manual.", Fao.org, 2018. [Online]. Available: http://www.fao.org/docrep/009/a0198e/A0198E06.htm. [Accessed: 22- Jul- 2018].

Appendix. R Code

```r
#IMPORTS##################################################################
library(readxl)
library(Hmisc)
library(plotrix)


library(ggplot2)


library(stratification)
library(sampling)
library(PracTools)
library(survey)


library(gridExtra)




#read individual sheet to dataframe object
df <- readxl::read_excel('projectData.xlsx',sheet='projectData')
head(df)


#Summary Statistics########################################################


summary(df)


nrow(df)
#Returns: 9762


sum(df$inventory)
#Returns: 1754954823
```

```
mean(df$inventory)
#Returns: 179774.1


sd(df$inventory)
#Returns: 1573358


plotrix::std.error(df$inventory)
#Returns: 15924.22


Hmisc::describe(df)


#VISUALIZING VARIABLE OF INTEREST###########################################


#Histogram of inventory, excluding values above 100K
ggplot(df,aes(x=inventory)) + geom_histogram(binwidth=1000,aes(fill=..count..)) +
  xlab('Inventory Value') + ylab('Count') + ggtitle('Counts of Inventory Value: Excluding Values > 100K')
  xlim(c(0, 100000)) + ylim(c(0,450))


#Histogram of inventory, excluding values above 1M
ggplot(df,aes(x=inventory)) + geom_histogram(binwidth=1000,fill='blue',alpha=0.6) +
  xlab('Inventory Value') + ylab('Count') + ggtitle('Counts of Inventory Value: Excluding Values > 1M') +
  xlim(c(0, 1000000)) + ylim(c(0,450))





##########################################################################################################


#Using highest valued Inventory records as the certainty stratum
#   CF. https://enablement.acl.com/helpdocs/analytics/13/user-guide/en-us/Content/da_sampling_data/class


#NB: The stratification::strata.LH() function below accepts an index vector as input to the argument spe
cert_vec <- list(top_certainty_vec <- c(which(df$inventory==max(df$inventory))),
```

```
                  top5_certainty_vec <- df[order(df$inventory, decreasing=T)[1:5],]$coID,
                  top10_certainty_vec <- df[order(df$inventory, decreasing=T)[1:10],]$coID,
                  top20_certainty_vec <- df[order(df$inventory, decreasing=T)[1:20],]$coID,
                  top50_certainty_vec <- df[order(df$inventory, decreasing=T)[1:50],]$coID,
                  top80_certainty_vec <- df[order(df$inventory, decreasing=T)[1:80],]$coID,
                  top90_certainty_vec <- df[order(df$inventory, decreasing=T)[1:90],]$coID,
                  top100_certainty_vec <- df[order(df$inventory, decreasing=T)[1:100],]$coID,
                  top110_certainty_vec <- df[order(df$inventory, decreasing=T)[1:110],]$coID,
                  top120_certainty_vec <- df[order(df$inventory, decreasing=T)[1:120],]$coID,
                  top150_certainty_vec <- df[order(df$inventory, decreasing=T)[1:150],]$coID,
                  top200_certainty_vec <- df[order(df$inventory, decreasing=T)[1:200],]$coID)


#NB: Must define cert_vec in global environment before passing to minCV() function below. For common scop
#    CF. https://stackoverflow.com/questions/28601959/error-object-not-found-in-nested-functions






#Generalized Lavallee-Hidiroglou Method -- applying Kozak's (2004) Algorithm#############################
#   CF. https://www.rdocumentation.org/packages/stratification/versions/2.2-5/topics/strata.LH


#Function Optimizing the number of sampled strata and the best performing certainty stratum from the vect


#NB:    Passing the vector c(0.5, 0, 0.5) to the 'alloc' argument of the stratification::strata.LH() func
#       Passing 'none' to the 'model' argument assumes that Y=X, i.e. that the variable being stratified


minCV <- function(lo_Ls, hi_Ls, cert_vec){
  CV.list<-numeric()
  min.CV <- 999
  min.i <- 99
  min.j <- 99

  for (i in lo_Ls:hi_Ls){
    for (j in 1:length(cert_vec)){
      k <- tryCatch(stratification::strata.LH(x=df$`inventory`, n=500, Ls=i, alloc=c(0.5,0,0.5),
                              certain=cert_vec[j], model="none", algo="Kozak")$RRMSE,
                    warning=function(w){
```

```
                      message("Warning: LH Algo failed. Some sampled strata < mininum Nh")
                      NA})
      if(is.na(k)){
        next
      }
      cat('Coefficient of Variation (RRMSE): ', k, '\n')
      cat('Selected Number of Strata: ', i, '\n')
      cat('Index of Certainty Stratum: ', j, '\n')

      CV.list <- append(CV.list, k)
      cat('All Coefficients of Variation: \n', CV.list, '\n')

      min_val <- min(CV.list, na.rm=TRUE)
      if (min.CV > min_val){
        min.CV <- min_val
        min.i <- i
        min.j <- j
      }
    }
  }
  result_list <- list('Minimum_CV'=min.CV, 'Best_LS'=min.i, 'Best_CertStratum'=min.j)
  return(result_list)
}



#Implement Stratification Optimization###################################################################


best_results <- minCV(5, 10, cert_vec)
best_results$Minimum_CV
best_results$Best_LS
best_results$Best_CertStratum



#Best Stratification Configuration: Lavallee-Hidiroglou (1988) Method and Kozak's (2004) Algorithm#######


Optim_Config <- stratification::strata.LH(x=df$`inventory`, n=500, Ls=10, alloc=c(0.5,0,0.5),
                             certain=cert_vec[9], model='none', algo='Kozak')
```

```
print(Optim_Config)
plot(Optim_Config)



#Performance Comparison: Gunning and Horgan (2004) Geometric Stratification Method

##########################################################################################################

Optim_Geo <- stratification::strata.geo(x=df$`inventory`, n=500, Ls = 8, alloc = c(0.5, 0, 0.5),
                        certain=cert_vec[9], model="none")


print(Optim_Geo)
plot(Optim_Geo)



##########################################################################################################

#Apply Optimum Stratification to Sampling and Estimation####################################################

#Preparing the data frame###################################################################################

#Identify boundary separating highest-valued certainty stratum from other strata
options(digits=15)
min(df[order(df$inventory, decreasing=T)[1:110],]$inventory)
#Returns 1372514.035

#Assign Strata
df$Stratum <- NA
df$Stratum[df$`inventory` < 36703.82] <- 1
df$Stratum[(df$`inventory` < 71655.62) & (df$`inventory` >= 36703.82)] <- 2
```

```
df$Stratum[(df$`inventory` < 106848.40) & (df$`inventory` >= 71655.62)] <- 3
df$Stratum[(df$`inventory` < 151104.80) & (df$`inventory` >= 106848.40)] <- 4
df$Stratum[(df$`inventory` < 209954.79) & (df$`inventory` >= 151104.80)] <- 5
df$Stratum[(df$`inventory` < 295186.09) & (df$`inventory` >= 209954.79)] <- 6
df$Stratum[(df$`inventory` < 413551.11) & (df$`inventory` >= 295186.09)] <- 7
df$Stratum[(df$`inventory` < 614205.40) & (df$`inventory` >= 413551.11)] <- 8
df$Stratum[(df$`inventory` < 974662.93) & (df$`inventory` >= 614205.40)] <- 9
df$Stratum[(df$`inventory` < 1372514.04) & (df$`inventory` >= 974662.93)] <- 10
df$Stratum[(df$`inventory` < 105379553.91) & (df$`inventory` >= 1372514.04)] <- 11


#Check if any NAs in Stratum column
any(is.na(df$Stratum))


#Assign column for Finite Population Correction
df$Num <- NA
df$Num[df$Stratum == 1] <- 4346
df$Num[df$Stratum == 2] <- 1047
df$Num[df$Stratum == 3] <- 1341
df$Num[df$Stratum == 4] <- 851
df$Num[df$Stratum == 5] <- 723
df$Num[df$Stratum == 6] <- 424
df$Num[df$Stratum == 7] <- 361
df$Num[df$Stratum == 8] <- 248
df$Num[df$Stratum == 9] <- 176
df$Num[df$Stratum == 10] <- 135
df$Num[df$Stratum == 11] <- 110


#Check if any NAs in df
any(is.na(df))


#Specify vector of population stratum sizes, appending certainty stratum
Nh <- append(Optim_Config$Nh, 110)


#Specify vector of population stratum standard deviations, appending 1 for certainty stratum
Sh <- append(sqrt(Optim_Config$varh), 1)


#NB:  Appending 1 as the certainty stratum standard deviation may result in an underestimation of varian
```

```
#       See results of sampling procedures #1-10 below, which serve as a performance comparison, assignin
#       The results of procedures #12-16 accept the stratum sample sizes determined by LH-Optimization, a
'certainty' stratum.




#Neyman Allocation to determine strata sizes in sample of size n = 500
PracTools::strAlloc(n.tot=500, Nh=Nh, Sh=Sh, alloc='neyman')
#Returns sample stratum sizes of (100, 38, 43, 39, 41, 39, 43, 50, 60, 47, 0)


#Run the allocation again for a sample of size 499, reserving 1 unit for the certainty stratum
Nh <- Optim_Config$Nh
Sh <- sqrt(Optim_Config$varh)
PracTools::strAlloc(n.tot=499, Nh=Nh, Sh=Sh, alloc='neyman')
#Returns sample stratum sizes of (101, 37, 43, 39, 41, 38, 43, 50, 60, 47)


#Adjusted Neyman Allocation, including certainty stratum
Ney_Alloc <- c(101, 37, 43, 39, 41, 38, 43, 50, 60, 47, 1)








###########################################################################################################


#PASS 1 OF STRATIFIED SAMPLING ESTIMATION of MEAN and TOTAL -- IMPLEMENTING LONELY PSU ADJUSTMENT
###########################################################################################################


#Set Random Sampling Seed
set.seed(222)


#Sampling using Neyman Allocation
Ney_Samp_1 <- sampling::strata(df, stratanames='Stratum', size=Ney_Alloc, method="srswor")
```

```r
#Drop duplicated column
Ney_Samp_1 <- Ney_Samp_1[, !duplicated(colnames(Ney_Samp_1))]


#Rename ID column to match original data frame, for merging
colnames(Ney_Samp_1)[2] <- "coID"
Ney_Samp_1_merged <- merge(x = df, y = Ney_Samp_1)


#sort by Stratum
Ney_Samp_1_merged <- Ney_Samp_1_merged[order(Ney_Samp_1_merged$Stratum),]


#Neyman stratified design object
mydesign_1 <- survey::svydesign(id = ~1, strata = ~Stratum, data = Ney_Samp_1_merged, fpc = ~Num)


#Adjust for single-PSU certainty stratum
options(survey.lonely.psu = "certainty")


#Neyman stratified mean estimate and SE
survey::svymean(~`inventory`, design = mydesign_1)
#Returns:                 mean          SE
#         inventory 169756.4553944  3452.97697


#Confidence interval of the mean estimate
confint(survey::svymean(~`inventory`, design = mydesign_1))
#Returns:                 2.5 %         97.5 %
#         inventory 162988.744885283  176524.165903571


#DOES NOT CONTAIN TRUE VALUE of 179774.106017914################


#Neyman stratified estimate of total and SE
survey::svytotal(~`inventory`, design = mydesign_1)
#Returns:                total          SE
#         inventory  1657162517.56  33707961.22347
```

```r
#Confidence interval of the estimate of total
confint(survey::svytotal(~`inventory`, design = mydesign_1))
#Returns:                2.5 %          97.5 %
#         inventory  1591096127.57013  1723228907.55066


#DOES NOT CONTAIN TRUE VALUE of 1754954822.94687#########








#PASS 2 OF STRATIFIED SAMPLING ESTIMATION of MEAN and TOTAL -- IMPLEMENTING LONELY PSU ADJUSTMENT
#########################################################################################################


#Set Random Sampling Seed
set.seed(223)


#Sampling using Neyman Allocation
Ney_Samp_2 <- sampling::strata(df, stratanames='Stratum', size=Ney_Alloc, method="srswor")


#Drop duplicated column
Ney_Samp_2 <- Ney_Samp_2[, !duplicated(colnames(Ney_Samp_2))]


#Rename ID column to match original data frame, for merging
colnames(Ney_Samp_2)[2] <- "coID"
Ney_Samp_2_merged <- merge(x = df, y = Ney_Samp_2)


#sort by Stratum
Ney_Samp_2_merged <- Ney_Samp_2_merged[order(Ney_Samp_2_merged$Stratum),]


#Neyman stratified design object
mydesign_2 <- survey::svydesign(id = ~1, strata = ~Stratum, data = Ney_Samp_2_merged, fpc = ~Num)
```

```r
#Adjust for single-PSU certainty stratum
options(survey.lonely.psu = "certainty")


#Neyman stratified mean estimate and SE
survey::svymean(~`inventory`, design = mydesign_2)
#Returns:                 mean         SE
#         inventory  171029.233252   3328.92537


#Confidence interval of the mean estimate
confint(survey::svymean(~`inventory`, design = mydesign_2))
#Returns:                2.5 %           97.5 %
#         inventory  164504.659423391  177553.807080663


#DOES NOT CONTAIN TRUE VALUE of 179774.106017914################


#Neyman stratified estimate of total and SE
survey::svytotal(~`inventory`, design = mydesign_2)
#Returns:                total          SE
#         inventory  1669587375.006  32496969.44308


#Confidence interval of the estimate of total
confint(survey::svytotal(~`inventory`, design = mydesign_2))
#Returns:               2.5 %           97.5 %
#         inventory  1605894485.29115  1733280264.72143


#DOES NOT CONTAIN TRUE VALUE of 1754954822.94687################





#PASS 3 OF STRATIFIED SAMPLING ESTIMATION of MEAN and TOTAL -- IMPLEMENTING LONELY PSU ADJUSTMENT
#########################################################################################################
```

```r
#Set Random Sampling Seed
set.seed(224)

#Sampling using Neyman Allocation
Ney_Samp_3 <- sampling::strata(df, stratanames='Stratum', size=Ney_Alloc, method="srswor")

#Drop duplicated column
Ney_Samp_3 <- Ney_Samp_3[, !duplicated(colnames(Ney_Samp_3))]

#Rename ID column to match original data frame, for merging
colnames(Ney_Samp_3)[2] <- "coID"
Ney_Samp_3_merged <- merge(x = df, y = Ney_Samp_3)

#sort by Stratum
Ney_Samp_3_merged <- Ney_Samp_3_merged[order(Ney_Samp_3_merged$Stratum),]

#Neyman stratified design object
mydesign_3 <- survey::svydesign(id = ~1, strata = ~Stratum, data = Ney_Samp_3_merged, fpc = ~Num)

#Adjust for single-PSU certainty stratum
options(survey.lonely.psu = "certainty")

#Neyman stratified mean estimate and SE
survey::svymean(~`inventory`, design = mydesign_3)
#Returns:               mean         SE
#         inventory  170941.7207444  3432.95716

#Confidence interval of the mean estimate
confint(survey::svymean(~`inventory`, design = mydesign_3))
#Returns:               2.5 %          97.5 %
#         inventory  164213.248356339  177670.193132409
```

```
#DOES NOT CONTAIN TRUE VALUE of 179774.106017914###############


#Neyman stratified estimate of total and SE
survey::svytotal(~`inventory`, design = mydesign_3)
#Returns:               total             SE
#         inventory  1668733077.907  33512527.76587


#Confidence interval of the estimate of total
confint(survey::svytotal(~`inventory`, design = mydesign_3))
#Returns:               2.5 %           97.5 %
#         inventory  1603049730.45458  1734416425.35857


#DOES NOT CONTAIN TRUE VALUE of 1754954822.94687###############







#PASS 4 OF STRATIFIED SAMPLING ESTIMATION of MEAN and TOTAL -- IMPLEMENTING LONELY PSU ADJUSTMENT
########################################################################################################


#Set Random Sampling Seed
set.seed(225)


#Sampling using Neyman Allocation
Ney_Samp_4 <- sampling::strata(df, stratanames='Stratum', size=Ney_Alloc, method="srswor")


#Drop duplicated column
Ney_Samp_4 <- Ney_Samp_4[, !duplicated(colnames(Ney_Samp_4))]


#Rename ID column to match original data frame, for merging
colnames(Ney_Samp_4)[2] <- "coID"
Ney_Samp_4_merged <- merge(x = df, y = Ney_Samp_4)
```

```r
#sort by Stratum
Ney_Samp_4_merged <- Ney_Samp_4_merged[order(Ney_Samp_4_merged$Stratum),]

#Neyman stratified design object
mydesign_4 <- survey::svydesign(id = ~1, strata = ~Stratum, data = Ney_Samp_4_merged, fpc = ~Num)

#Adjust for single-PSU certainty stratum
options(survey.lonely.psu = "certainty")

#Neyman stratified mean estimate and SE
survey::svymean(~`inventory`, design = mydesign_4)
#Returns:                 mean          SE
#         inventory  180661.3660204  4638.56949

#Confidence interval of the mean estimate
confint(survey::svymean(~`inventory`, design = mydesign_4))
#Returns:                 2.5 %            97.5 %
#         inventory  171569.936881626  189752.795159095

#CONTAINS TRUE VALUE of 179774.106017914################

#Neyman stratified estimate of total and SE
survey::svytotal(~`inventory`, design = mydesign_4)
#Returns:                 total             SE
#         inventory  1763616255.091  45281715.35415

#Confidence interval of the estimate of total
confint(survey::svytotal(~`inventory`, design = mydesign_4))
#Returns:                 2.5 %             97.5 %
#         inventory  1674865723.83843  1852366786.34309

#CONTAINS TRUE VALUE of 1754954822.94687################
```

```
#PASS 5 OF STRATIFIED SAMPLING ESTIMATION of MEAN and TOTAL -- IMPLEMENTING LONELY PSU ADJUSTMENT
#########################################################################################################


#Set Random Sampling Seed
set.seed(226)


#Sampling using Neyman Allocation
Ney_Samp_5 <- sampling::strata(df, stratanames='Stratum', size=Ney_Alloc, method="srswor")


#Drop duplicated column
Ney_Samp_5 <- Ney_Samp_5[, !duplicated(colnames(Ney_Samp_5))]


#Rename ID column to match original data frame, for merging
colnames(Ney_Samp_5)[2] <- "coID"
Ney_Samp_5_merged <- merge(x = df, y = Ney_Samp_5)


#sort by Stratum
Ney_Samp_5_merged <- Ney_Samp_5_merged[order(Ney_Samp_5_merged$Stratum),]


#Neyman stratified design object
mydesign_5 <- survey::svydesign(id = ~1, strata = ~Stratum, data = Ney_Samp_5_merged, fpc = ~Num)


#Adjust for single-PSU certainty stratum
options(survey.lonely.psu = "certainty")


#Neyman stratified mean estimate and SE
survey::svymean(~`inventory`, design = mydesign_5)
#Returns:               mean             SE
```

```
#         inventory  188972.1127207  4634.39056


#Confidence interval of the mean estimate
confint(survey::svymean(~`inventory`, design = mydesign_5))
#Returns:               2.5 %             97.5 %
# q       inventory  179888.874126264  198055.351315211


#DOES NOT CONTAIN TRUE VALUE of 179774.106017914###############


#Neyman stratified estimate of total and SE
survey::svytotal(~`inventory`, design = mydesign_5)
#Returns:                total            SE
#         inventory  1844745764.38   45240920.6795


#Confidence interval of the estimate of total
confint(survey::svytotal(~`inventory`, design = mydesign_5))
#Returns:                  2.5 %          97.5 %
#         inventory  1756075189.22059  1933416339.53909


#DOES NOT CONTAIN TRUE VALUE of 1754954822.94687#########









##########################################################################################################


#PASS 6 OF STRATIFIED SAMPLING ESTIMATION of MEAN and TOTAL -- NO LONELY PSU CERTAINTY STRATUM ADJUSTMENT
##########################################################################################################


#Run the allocation again for a sample of size 498, reserving 2 units for the certainty stratum
```

```r
Nh <- Optim_Config$Nh
Sh <- sqrt(Optim_Config$varh)
PracTools::strAlloc(n.tot=498, Nh=Nh, Sh=Sh, alloc='neyman')
#returns sample stratum sizes of (101, 37, 43, 39, 41, 38, 42, 50, 60, 47)


#adjusted Neyman Allocation, including certainty stratum
Ney_Alloc_2 <- c(101, 37, 43, 39, 41, 38, 43, 50, 60, 47, 2)







#Set Random Sampling Seed
set.seed(227)


#Sampling using Neyman Allocation
Ney_Samp_6 <- sampling::strata(df, stratanames='Stratum', size=Ney_Alloc_2, method="srswor")


#Drop duplicated column
Ney_Samp_6 <- Ney_Samp_6[, !duplicated(colnames(Ney_Samp_6))]


#Rename ID column to match original data frame, for merging
colnames(Ney_Samp_6)[2] <- "coID"
Ney_Samp_6_merged <- merge(x = df, y = Ney_Samp_6)


#sort by Stratum
Ney_Samp_6_merged <- Ney_Samp_6_merged[order(Ney_Samp_6_merged$Stratum),]


#Neyman stratified design object
mydesign_6 <- survey::svydesign(id = ~1, strata = ~Stratum, data = Ney_Samp_6_merged, fpc = ~Num)


#Neyman stratified mean estimate and SE
survey::svymean(~`inventory`, design = mydesign_6)
#Returns:                mean           SE
#         inventory  185135.8284971  4675.96438
```

```
#Confidence interval of the mean estimate
confint(survey::svymean(~`inventory`, design = mydesign_6))
#Returns:                  2.5 %          97.5 %
#         inventory  175971.106721974  194300.550272168


#CONTAINS TRUE VALUE of 179774.10601791 4################


#Neyman stratified estimate of total and SE
survey::svytotal(~`inventory`, design = mydesign_6)
#Returns:                  total             SE
#         inventory   1807295957.788   45646764.26414


#Confidence interval of the estimate of total
confint(survey::svytotal(~`inventory`, design = mydesign_6))
#Returns:                  2.5 %             97.5 %
#         inventory   1717829943.81991   1896761971.7569


#CONTAINS TRUE VALUE of 1754954822.94687################






#PASS 7 OF STRATIFIED SAMPLING ESTIMATION of MEAN and TOTAL -- NO LONELY PSU CERTAINTY STRATUM ADJUSTMEN
##########################################################################################################




#Set Random Sampling Seed
set.seed(228)


#Sampling using Neyman Allocation
```

```
Ney_Samp_7 <- sampling::strata(df, stratanames='Stratum', size=Ney_Alloc_2, method="srswor")

#Drop duplicated column
Ney_Samp_7 <- Ney_Samp_7[, !duplicated(colnames(Ney_Samp_7))]

#Rename ID column to match original data frame, for merging
colnames(Ney_Samp_7)[2] <- "coID"
Ney_Samp_7_merged <- merge(x = df, y = Ney_Samp_7)

#sort by Stratum
Ney_Samp_7_merged <- Ney_Samp_7_merged[order(Ney_Samp_7_merged$Stratum),]

#Neyman stratified design object
mydesign_7 <- survey::svydesign(id = ~1, strata = ~Stratum, data = Ney_Samp_7_merged, fpc = ~Num)

#Neyman stratified mean estimate and SE
survey::svymean(~`inventory`, design = mydesign_7)
#Returns:                  mean               SE
#         inventory   179848.508501     4643.21721

#Confidence interval of the mean estimate
confint(survey::svymean(~`inventory`, design = mydesign_7))
#Returns:                  2.5 %              97.5 %
#         inventory   170747.969987538   188949.047014456

#CONTAINS TRUE VALUE of 179774.106017914################

#Neyman stratified estimate of total and SE
survey::svytotal(~`inventory`, design = mydesign_7)
#Returns:                 total              SE
#         inventory   1755681139.987   45327086.45115

#Confidence interval of the estimate of total
confint(survey::svytotal(~`inventory`, design = mydesign_7))
```

```r
#Returns:                   2.5 %             97.5 %
#         inventory   1666841683.01835   1844520596.95512


#CONTAINS TRUE VALUE of 1754954822.94687#################






#PASS 8 OF STRATIFIED SAMPLING ESTIMATION of MEAN and TOTAL -- NO LONELY PSU CERTAINTY STRATUM ADJUSTMENT
###########################################################################################################



#Set Random Sampling Seed
set.seed(229)


#Sampling using Neyman Allocation
Ney_Samp_8 <- sampling::strata(df, stratanames='Stratum', size=Ney_Alloc_2, method="srswor")


#Drop duplicated column
Ney_Samp_8 <- Ney_Samp_8[, !duplicated(colnames(Ney_Samp_8))]


#Rename ID column to match original data frame, for merging
colnames(Ney_Samp_8)[2] <- "coID"
Ney_Samp_8_merged <- merge(x = df, y = Ney_Samp_8)


#sort by Stratum
Ney_Samp_8_merged <- Ney_Samp_8_merged[order(Ney_Samp_8_merged$Stratum),]


#Neyman stratified design object
mydesign_8 <- survey::svydesign(id = ~1, strata = ~Stratum, data = Ney_Samp_8_merged, fpc = ~Num)
```

```
#Neyman stratified mean estimate and SE
survey::svymean(~`inventory`, design = mydesign_8)
#Returns:                   mean         SE
#         inventory   184075.5593905  4733.82151


#Confidence interval of the mean estimate
confint(survey::svymean(~`inventory`, design = mydesign_8))
#Returns:                  2.5 %              97.5 %
#         inventory   174797.439716593   193353.679064313


#CONTAINS TRUE VALUE of 179774.106017914################


#Neyman stratified estimate of total and SE
survey::svytotal(~`inventory`, design = mydesign_8)
#Returns:                 total             SE
#         inventory   1796945610.77    46211565.60562


#Confidence interval of the estimate of total
confint(survey::svytotal(~`inventory`, design = mydesign_8))
#Returns:                 2.5 %           97.5 %
#         inventory  1706372606.51338  1887518615.02582


#CONTAINS TRUE VALUE of 1754954822.94687#################







#PASS 9 OF STRATIFIED SAMPLING ESTIMATION of MEAN and TOTAL -- NO LONELY PSU CERTAINTY STRATUM ADJUSTMEN
###########################################################################################################
```

```r
#Set Random Sampling Seed
set.seed(230)

#Sampling using Neyman Allocation
Ney_Samp_9 <- sampling::strata(df, stratanames='Stratum', size=Ney_Alloc_2, method="srswor")

#Drop duplicated column
Ney_Samp_9 <- Ney_Samp_9[, !duplicated(colnames(Ney_Samp_9))]

#Rename ID column to match original data frame, for merging
colnames(Ney_Samp_9)[2] <- "coID"
Ney_Samp_9_merged <- merge(x = df, y = Ney_Samp_9)

#sort by Stratum
Ney_Samp_9_merged <- Ney_Samp_9_merged[order(Ney_Samp_9_merged$Stratum),]

#Neyman stratified design object
mydesign_9 <- survey::svydesign(id = ~1, strata = ~Stratum, data = Ney_Samp_9_merged, fpc = ~Num)

#Neyman stratified mean estimate and SE
survey::svymean(~`inventory`, design = mydesign_9)
#Returns:                  mean          SE
#         inventory   182583.5832636   5040.97741

#Confidence interval of the mean estimate
confint(survey::svymean(~`inventory`, design = mydesign_9))
#Returns:                  2.5 %             97.5 %
#         inventory  172703.449097735   192463.717429423

#CONTAINS TRUE VALUE of 179774.106017914###############

#Neyman stratified estimate of total and SE
survey::svytotal(~`inventory`, design = mydesign_9)
#Returns:                 total              SE
```

```
#         inventory  1782380939.819   49210021.45333


#Confidence interval of the estimate of total
confint(survey::svytotal(~`inventory`, design = mydesign_9))
#Returns:                 2.5 %            97.5 %
#         inventory  1685931070.09209   1878830809.54603


#CONTAINS TRUE VALUE of 1754954822.94687#################









#PASS 10 OF STRATIFIED SAMPLING ESTIMATION of MEAN and TOTAL -- NO LONELY PSU CERTAINTY STRATUM ADJUSTMEN
##########################################################################################################




#Set Random Sampling Seed
set.seed(231)


#Sampling using Neyman Allocation
Ney_Samp_10 <- sampling::strata(df, stratanames='Stratum', size=Ney_Alloc_2, method="srswor")


#Drop duplicated column
Ney_Samp_10 <- Ney_Samp_10[, !duplicated(colnames(Ney_Samp_10))]


#Rename ID column to match original data frame, for merging
colnames(Ney_Samp_10)[2] <- "coID"
Ney_Samp_10_merged <- merge(x = df, y = Ney_Samp_10)


#sort by Stratum
Ney_Samp_10_merged <- Ney_Samp_10_merged[order(Ney_Samp_10_merged$Stratum),]
```

```r
#Neyman stratified design object
mydesign_10 <- survey::svydesign(id = ~1, strata = ~Stratum, data = Ney_Samp_10_merged, fpc = ~Num)


#Neyman stratified mean estimate and SE
survey::svymean(~`inventory`, design = mydesign_10)
#Returns:                 mean           SE
#         inventory  170758.0515275  3455.03629


#Confidence interval of the mean estimate
confint(survey::svymean(~`inventory`, design = mydesign_10))
#Returns:                  2.5 %             97.5 %
#         inventory  163986.304831325   177529.798223672


#DOES NOT CONTAIN TRUE VALUE of 179774.106017914###########


#Neyman stratified estimate of total and SE
survey::svytotal(~`inventory`, design = mydesign_10)
#Returns:                  total            SE
#         inventory   1666940099.011   33728064.2754


#Confidence interval of the estimate of total
confint(survey::svytotal(~`inventory`, design = mydesign_10))
#Returns:                  2.5 %            97.5 %
#         inventory   1600834307.7634   1733045890.25948


#DOES NOT CONTAIN TRUE VALUE of 1754954822.94687############
```

```
##################################################################################################################

#PASS 11: PERFORMANCE COMPARISON with UNSTRATIFIED SIMPLE RANDOM SAMPLING
##################################################################################################################

#Set Random Sampling Seed
set.seed(1)

#SRS of Size 500
srs_df <- df[sample(nrow(df), 500), ]

#SRS design object
mydesign_srs <- survey::svydesign(id = ~1, data = srs_df)

#SRS mean estimate and SE
survey::svymean(~`inventory`, design = mydesign_srs)
#Returns:                     mean           SE
#         inventory   314438.5186579    203292.38226

#confidence interval of SRS mean estimate
confint(survey::svymean(~`inventory`, design = mydesign_srs))
#Returns:                     2.5 %         97.5 %
#         inventory  -84007.2288961298  712884.266211934

#HUGE UNINFORMATIVE CONFIDENCE INTERVAL CONTAINS TRUE VALUE of 179774.106017914###

#SRS estimate of total and SE
survey::svytotal(~`inventory`, design = mydesign_srs)
#Returns:                   total             SE
#         inventory  157219259.329      101646191.1282
```

```
#Confidence interval of SRS estimate of total
confint(survey::svytotal(~`inventory`, design = mydesign_srs))
#Returns:                   2.5 %          97.5 %
#         inventory  -42003614.4480649   356442133.105967


#HUGE UNINFORMATIVE CONFIDENCE INTERVAL STILL DOES NOT CONTAIN TRUE VALUE of 1754954822.94687###







##################################################################################################################



#PASS 12 OF STRATIFIED SAMPLING ESTIMATION of MEAN and TOTAL -- ASSUMING SAMPLE STRATUM SIZES ASSIGNED BY
##################################################################################################################




#adjusted Neyman Allocation, assuming values provided by the LH optimization
Ney_Alloc_3 <- append(Optim_Config$nh, 110)


#Set Random Sampling Seed
set.seed(232)


#Sampling using Neyman Allocation
Ney_Samp_12 <- sampling::strata(df, stratanames='Stratum', size=Ney_Alloc_3, method="srswor")


#Drop duplicated column
Ney_Samp_12 <- Ney_Samp_12[, !duplicated(colnames(Ney_Samp_12))]
```

```r
#Rename ID column to match original data frame, for merging
colnames(Ney_Samp_12)[2] <- "coID"
Ney_Samp_12_merged <- merge(x = df, y = Ney_Samp_12)


#sort by Stratum
Ney_Samp_12_merged <- Ney_Samp_12_merged[order(Ney_Samp_12_merged$Stratum),]


#Neyman stratified design object
mydesign_12 <- survey::svydesign(id = ~1, strata = ~Stratum, data = Ney_Samp_12_merged, fpc = ~Num)


#Neyman stratified mean estimate and SE
survey::svymean(~`inventory`, design = mydesign_12)
#Returns:                 mean             SE
#         inventory  173299.0523965  7965.95442


#Confidence interval of the mean estimate
confint(survey::svymean(~`inventory`, design = mydesign_12))
#Returns:                 2.5 %             97.5 %
#         inventory  157686.068638111  188912.036154827


#CONTAINS TRUE VALUE of 179774.106017914################


#Neyman stratified estimate of total and SE
survey::svytotal(~`inventory`, design = mydesign_12)
#Returns:                 total             SE
#         inventory   1691745349.494   77763647.01153


#Confidence interval of the estimate of total
confint(survey::svytotal(~`inventory`, design = mydesign_12))
#Returns:                 2.5 %             97.5 %
#         inventory  1539331402.04524   1844159296.94342


#CONTAINS TRUE VALUE of 1754954822.94687################
```

```
#PASS 13 OF STRATIFIED SAMPLING ESTIMATION of MEAN and TOTAL -- ASSUMING SAMPLE STRATUM SIZES ASSIGNED BY
################################################################################################


#adjusted Neyman Allocation, assuming values provided by the LH optimization
Ney_Alloc_3 <- append(Optim_Config$nh, 110)

#Set Random Sampling Seed
set.seed(233)

#Sampling using Neyman Allocation
Ney_Samp_13 <- sampling::strata(df, stratanames='Stratum', size=Ney_Alloc_3, method="srswor")

#Drop duplicated column
Ney_Samp_13 <- Ney_Samp_13[, !duplicated(colnames(Ney_Samp_13))]

#Rename ID column to match original data frame, for merging
colnames(Ney_Samp_13)[2] <- "coID"
Ney_Samp_13_merged <- merge(x = df, y = Ney_Samp_13)

#sort by Stratum
Ney_Samp_13_merged <- Ney_Samp_13_merged[order(Ney_Samp_13_merged$Stratum),]

#Neyman stratified design object
mydesign_13 <- survey::svydesign(id = ~1, strata = ~Stratum, data = Ney_Samp_13_merged, fpc = ~Num)
```

```
#Neyman stratified mean estimate and SE
survey::svymean(~`inventory`, design = mydesign_13)
#Returns:                  mean         SE
#         inventory  170343.7363795   7660.27815


#Confidence interval of the mean estimate
confint(survey::svymean(~`inventory`, design = mydesign_13))
#Returns:                 2.5 %              97.5 %
#         inventory  155329.867097901   185357.605661186


#CONTAINS TRUE VALUE of 179774.106017914################


#Neyman stratified estimate of total and SE
survey::svytotal(~`inventory`, design = mydesign_13)
#Returns:                total              SE
#         inventory  1662895554.537   74779635.28079


#Confidence interval of the estimate of total
confint(survey::svytotal(~`inventory`, design = mydesign_13))
#Returns:                2.5 %              97.5 %
#         inventory  1516330162.60971  1809460946.4645


#CONTAINS TRUE VALUE of 1754954822.94687#################









#PASS 14 OF STRATIFIED SAMPLING ESTIMATION of MEAN and TOTAL -- ASSUMING SAMPLE STRATUM SIZES ASSIGNED BY
##########################################################################################################
```

```r
#adjusted Neyman Allocation, assuming values provided by the LH optimization
Ney_Alloc_3 <- append(Optim_Config$nh, 110)

#Set Random Sampling Seed
set.seed(234)

#Sampling using Neyman Allocation
Ney_Samp_14 <- sampling::strata(df, stratanames='Stratum', size=Ney_Alloc_3, method="srswor")

#Drop duplicated column
Ney_Samp_14 <- Ney_Samp_14[, !duplicated(colnames(Ney_Samp_14))]

#Rename ID column to match original data frame, for merging
colnames(Ney_Samp_14)[2] <- "coID"
Ney_Samp_14_merged <- merge(x = df, y = Ney_Samp_14)

#sort by Stratum
Ney_Samp_14_merged <- Ney_Samp_14_merged[order(Ney_Samp_14_merged$Stratum),]

#Neyman stratified design object
mydesign_14 <- survey::svydesign(id = ~1, strata = ~Stratum, data = Ney_Samp_14_merged, fpc = ~Num)

#Neyman stratified mean estimate and SE
survey::svymean(~`inventory`, design = mydesign_14)
#Returns:                 mean          SE
#         inventory   175205.17016   7810.02253

#Confidence interval of the mean estimate
confint(survey::svymean(~`inventory`, design = mydesign_14))
#Returns:                 2.5 %             97.5 %
#         inventory  159897.807284972  190512.533034961
```

```
#CONTAINS TRUE VALUE of 179774.106017914###############


#Neyman stratified estimate of total and SE
survey::svytotal(~`inventory`, design = mydesign_14)
#Returns:                total            SE
#         inventory  1710352871.102  76241439.92664


#Confidence interval of the estimate of total
confint(survey::svytotal(~`inventory`, design = mydesign_14))
#Returns:                 2.5 %            97.5 %
#         inventory  1560922394.7159   1859783347.48729


#CONTAINS TRUE VALUE of 1754954822.94687################







#PASS 15 OF STRATIFIED SAMPLING ESTIMATION of MEAN and TOTAL -- ASSUMING SAMPLE STRATUM SIZES ASSIGNED BY
#########################################################################################################



#adjusted Neyman Allocation, assuming values provided by the LH optimization
Ney_Alloc_3 <- append(Optim_Config$nh, 110)


#Set Random Sampling Seed
set.seed(235)


#Sampling using Neyman Allocation
Ney_Samp_15 <- sampling::strata(df, stratanames='Stratum', size=Ney_Alloc_3, method="srswor")
```

```
#Drop duplicated column
Ney_Samp_15 <- Ney_Samp_15[, !duplicated(colnames(Ney_Samp_15))]


#Rename ID column to match original data frame, for merging
colnames(Ney_Samp_15)[2] <- "coID"
Ney_Samp_15_merged <- merge(x = df, y = Ney_Samp_15)


#sort by Stratum
Ney_Samp_15_merged <- Ney_Samp_15_merged[order(Ney_Samp_15_merged$Stratum),]


#Neyman stratified design object
mydesign_15 <- survey::svydesign(id = ~1, strata = ~Stratum, data = Ney_Samp_15_merged, fpc = ~Num)


#Neyman stratified mean estimate and SE
survey::svymean(~`inventory`, design = mydesign_15)
#Returns:                 mean           SE
#         inventory  174342.7305529  8034.61217


#Confidence interval of the mean estimate
confint(survey::svymean(~`inventory`, design = mydesign_15))
#Returns:                 2.5 %              97.5 %
#         inventory  158595.180070016   190090.281035698


#CONTAINS TRUE VALUE of 179774.106017914################


#Neyman stratified estimate of total and SE
survey::svytotal(~`inventory`, design = mydesign_15)
#Returns:                 total              SE
#         inventory  1701933735.657   78433884.00301


#Confidence interval of the estimate of total
confint(survey::svytotal(~`inventory`, design = mydesign_15))
#Returns:                 2.5 %              97.5 %
#         inventory   1548206147.8435   1855661323.47048
```

```r
#CONTAINS TRUE VALUE of 1754954822.94687################







#PASS 16 OF STRATIFIED SAMPLING ESTIMATION of MEAN and TOTAL -- ASSUMING SAMPLE STRATUM SIZES ASSIGNED BY
##############################################################################################################



#adjusted Neyman Allocation, assuming values provided by the LH optimization
Ney_Alloc_3 <- append(Optim_Config$nh, 110)


#Set Random Sampling Seed
set.seed(236)


#Sampling using Neyman Allocation
Ney_Samp_16 <- sampling::strata(df, stratanames='Stratum', size=Ney_Alloc_3, method="srswor")


#Drop duplicated column
Ney_Samp_16 <- Ney_Samp_16[, !duplicated(colnames(Ney_Samp_16))]


#Rename ID column to match original data frame, for merging
colnames(Ney_Samp_16)[2] <- "coID"
Ney_Samp_16_merged <- merge(x = df, y = Ney_Samp_16)


#sort by Stratum
Ney_Samp_16_merged <- Ney_Samp_16_merged[order(Ney_Samp_16_merged$Stratum),]
```

```r
#Neyman stratified design object
mydesign_16 <- survey::svydesign(id = ~1, strata = ~Stratum, data = Ney_Samp_16_merged, fpc = ~Num)

#Neyman stratified mean estimate and SE
survey::svymean(~`inventory`, design = mydesign_16)
#Returns:                  mean           SE
#         inventory  170959.9091492  7684.90266

#Confidence interval of the mean estimate
confint(survey::svymean(~`inventory`, design = mydesign_16))
#Returns:                  2.5 %            97.5 %
#         inventory  155897.77670773   186022.04159064

#CONTAINS TRUE VALUE of 179774.106017914###############

#Neyman stratified estimate of total and SE
survey::svytotal(~`inventory`, design = mydesign_16)
#Returns:                  total             SE
#         inventory   1668910633.114    75020019.78265

#Confidence interval of the estimate of total
confint(survey::svytotal(~`inventory`, design = mydesign_16))
#Returns:                 2.5 %                97.5 %
#         inventory  1521874096.22086   1815947170.00782

#CONTAINS TRUE VALUE of 1754954822.94687################
```

```
###############################################################################################

#SUMMARY OF RESULTS##########################################################################

#Mean of the Total and Mean Estimates and their Standard Errors (across Samples assuming the Sizes alloca

m.est_of_mean <- mean(173299.05, 170343.74, 175205.17, 174342.73, 170959.91)
#Returns: 173299.05

m.est_of_mean_SE <- mean(7965.95, 7660.28, 7810.02, 8034.61, 7684.90)
#Returns: 7965.95

m.est_of_total <- mean(1691745349.49, 1662895554.54, 1710352871.10, 1701933735.66, 1668910633.11)
#Returns: 1691745349.49

m.est_of_total_SE <- mean(77763647.01, 74779635.28, 76241439.93, 78433884.00, 75020019.78)
#Returns: 77763647.01




#VISUALIZE TABLE OF RESULTS##################################################################

val_df <- setNames(data.frame(matrix(ncol = 8, nrow = 16)),
                   c('Mean_Estimate','Mean_SE','Mean_95%_CI','Includes_True_Mean','Total_Estimate','Total

rownames(val_df) <- c(paste('Stratified_Test',1:5), paste('Stratified_No_Lonely_PSU',1:5), 'SRS', paste(

val_df[,1] <- c(169756.46, 171029.23, 170941.72, 180661.37, 188972.11,
                185135.83, 179848.51, 184075.56, 182583.58, 170758.05,
```

```
                314438.52,
                173299.05, 170343.74, 175205.17, 174342.73, 170959.91)


val_df[,2] <- c(3452.98, 3328.93, 3432.96, 4638.57, 4634.39,
                4675.96, 4643.22, 4733.82, 5040.98, 3455.04,
                203292.38,
                7965.95, 7660.28, 7810.02, 8034.61, 7684.90)


val_df[,3] <- c('[162988.75, 176524.17]','[164504.66, 177553.81]','[164213.25, 177670.19]','[171569.94, 1
                '[175971.11, 194300.55]','[170747.97, 188949.05]','[174797.44, 193353.68]','[172703.45, 1
                '[-84007.23, 712884.27]',
                '[157686.07, 188912.04]','[155329.87, 185357.61]','[159897.81, 190512.53]','[158595.18, 1


val_df[,4] <- c('NO','NO','NO','YES','NO',
                'YES','YES','YES','YES','NO',
                'YES*',
                'YES','YES','YES','YES','YES')


val_df[,5] <- c(1657162517.56, 1669587375.01, 1668733077.91, 1763616255.09, 1844745764.38,
                1807295957.79, 1755681139.99, 1796945610.77, 1782380939.82, 1666940099.01,
                157219259.33,
                1691745349.49, 1662895554.54, 1710352871.10, 1701933735.66, 1668910633.11)


val_df[,6] <- c(33707961.22, 32496969.44, 33512527.77, 45281715.35, 45240920.68,
                45646764.26, 45327086.45, 46211565.61, 49210021.45, 33728064.28,
                101646191.13,
                77763647.01, 74779635.28, 76241439.93, 78433884.00, 75020019.78)


val_df[,7] <- c('[1591096127.57, 1723228907.55]','[1605894485.29, 1733280264.72]','[1603049730.45, 17344
1852366786.34]','[1756075189.22, 1933416339.54]',
                '[1717829943.82, 1896761971.76]','[1666841683.02, 1844520596.96]','[1706372606.51, 18875
1878830809.55]','[1600834307.76. 1733045890.26]',
                '[-42003614.45, 356442133.11]',
                '[1539331402.05, 1844159296.94]','[1516330162.61, 1809460946.46]','[1560922394.72, 18597
1855661323.47]','[1521874096.22, 1815947170.01]')
```

```r
val_df[,8] <- c('NO','NO','NO','YES','NO',
                'YES','YES','YES','YES','NO',
                'NO',
                'YES','YES','YES','YES','YES')


#Create grid of results values
val_tbl <- tableGrob(val_df)


#Helper function to find cells in grid
find_cell <- function(table, row, col, name="core-fg"){
  l <- table$layout
  which(l$t==row & l$l==col & l$name==name)
}




#Coloring 'YES' cells resulting from the LH Optimized Stratification Method
ind <- find_cell(val_tbl, 13, 5, "core-bg")
val_tbl$grobs[ind][[1]][["gp"]] <- gpar(fill="darkolivegreen1", col = "darkolivegreen4", lwd=5)


ind2 <- find_cell(val_tbl, 14, 5, "core-bg")
val_tbl$grobs[ind2][[1]][["gp"]] <- gpar(fill="darkolivegreen1", col = "darkolivegreen4", lwd=5)


ind3 <- find_cell(val_tbl, 15, 5, "core-bg")
val_tbl$grobs[ind3][[1]][["gp"]] <- gpar(fill="darkolivegreen1", col = "darkolivegreen4", lwd=5)


ind4 <- find_cell(val_tbl, 16, 5, "core-bg")
val_tbl$grobs[ind4][[1]][["gp"]] <- gpar(fill="darkolivegreen1", col = "darkolivegreen4", lwd=5)


ind5 <- find_cell(val_tbl, 17, 5, "core-bg")
val_tbl$grobs[ind5][[1]][["gp"]] <- gpar(fill="darkolivegreen1", col = "darkolivegreen4", lwd=5)




ind6 <- find_cell(val_tbl, 13, 9, "core-bg")
```

```
val_tbl$grobs[ind6][[1]][["gp"]] <- gpar(fill="darkolivegreen1", col = "darkolivegreen4", lwd=5)

ind7 <- find_cell(val_tbl, 14, 9, "core-bg")
val_tbl$grobs[ind7][[1]][["gp"]] <- gpar(fill="darkolivegreen1", col = "darkolivegreen4", lwd=5)

ind8 <- find_cell(val_tbl, 15, 9, "core-bg")
val_tbl$grobs[ind8][[1]][["gp"]] <- gpar(fill="darkolivegreen1", col = "darkolivegreen4", lwd=5)

ind9 <- find_cell(val_tbl, 16, 9, "core-bg")
val_tbl$grobs[ind9][[1]][["gp"]] <- gpar(fill="darkolivegreen1", col = "darkolivegreen4", lwd=5)

ind10 <- find_cell(val_tbl, 17, 9, "core-bg")
val_tbl$grobs[ind10][[1]][["gp"]] <- gpar(fill="darkolivegreen1", col = "darkolivegreen4", lwd=5)



#Coloring 'NO' cells resulting from the SRS method and from strafication methods allocating 1 or 2 reco
ind11 <- find_cell(val_tbl, 2, 5, "core-bg")
val_tbl$grobs[ind11][[1]][["gp"]] <- gpar(fill="coral1", col = "coral4", lwd=5)

ind12 <- find_cell(val_tbl, 3, 5, "core-bg")
val_tbl$grobs[ind12][[1]][["gp"]] <- gpar(fill="coral1", col = "coral4", lwd=5)

ind13 <- find_cell(val_tbl, 4, 5, "core-bg")
val_tbl$grobs[ind13][[1]][["gp"]] <- gpar(fill="coral1", col = "coral4", lwd=5)

ind14 <- find_cell(val_tbl, 6, 5, "core-bg")
val_tbl$grobs[ind14][[1]][["gp"]] <- gpar(fill="coral1", col = "coral4", lwd=5)

ind15 <- find_cell(val_tbl, 11, 5, "core-bg")
val_tbl$grobs[ind15][[1]][["gp"]] <- gpar(fill="coral1", col = "coral4", lwd=5)
```

```
ind16 <- find_cell(val_tbl, 2, 9, "core-bg")
val_tbl$grobs[ind16][[1]][["gp"]] <- gpar(fill="coral1", col = "coral4", lwd=5)

ind17 <- find_cell(val_tbl, 3, 9, "core-bg")
val_tbl$grobs[ind17][[1]][["gp"]] <- gpar(fill="coral1", col = "coral4", lwd=5)

ind18 <- find_cell(val_tbl, 4, 9, "core-bg")
val_tbl$grobs[ind18][[1]][["gp"]] <- gpar(fill="coral1", col = "coral4", lwd=5)

ind19 <- find_cell(val_tbl, 6, 9, "core-bg")
val_tbl$grobs[ind19][[1]][["gp"]] <- gpar(fill="coral1", col = "coral4", lwd=5)

ind20 <- find_cell(val_tbl, 11, 9, "core-bg")
val_tbl$grobs[ind20][[1]][["gp"]] <- gpar(fill="coral1", col = "coral4", lwd=5)

ind21 <- find_cell(val_tbl, 12, 9, "core-bg")
val_tbl$grobs[ind21][[1]][["gp"]] <- gpar(fill="coral1", col = "coral4", lwd=5)



#Coloring 'YES' cells resulting from strafication methods allocating 1 or 2 records to the upper certaint
ind22 <- find_cell(val_tbl, 5, 5, "core-bg")
val_tbl$grobs[ind22][[1]][["gp"]] <- gpar(fill="azure1", col = "azure4", lwd=5)

ind23 <- find_cell(val_tbl, 7, 5, "core-bg")
val_tbl$grobs[ind23][[1]][["gp"]] <- gpar(fill="azure1", col = "azure4", lwd=5)

ind24 <- find_cell(val_tbl, 8, 5, "core-bg")
val_tbl$grobs[ind24][[1]][["gp"]] <- gpar(fill="azure1", col = "azure4", lwd=5)

ind25 <- find_cell(val_tbl, 9, 5, "core-bg")
val_tbl$grobs[ind25][[1]][["gp"]] <- gpar(fill="azure1", col = "azure4", lwd=5)
```

```r
ind26 <- find_cell(val_tbl, 10, 5, "core-bg")
val_tbl$grobs[ind26][[1]][["gp"]] <- gpar(fill="azure1", col = "azure4", lwd=5)



ind27 <- find_cell(val_tbl, 5, 9, "core-bg")
val_tbl$grobs[ind27][[1]][["gp"]] <- gpar(fill="azure1", col = "azure4", lwd=5)

ind28 <- find_cell(val_tbl, 7, 9, "core-bg")
val_tbl$grobs[ind28][[1]][["gp"]] <- gpar(fill="azure1", col = "azure4", lwd=5)

ind29 <- find_cell(val_tbl, 8, 9, "core-bg")
val_tbl$grobs[ind29][[1]][["gp"]] <- gpar(fill="azure1", col = "azure4", lwd=5)

ind30 <- find_cell(val_tbl, 9, 9, "core-bg")
val_tbl$grobs[ind30][[1]][["gp"]] <- gpar(fill="azure1", col = "azure4", lwd=5)

ind31 <- find_cell(val_tbl, 10, 9, "core-bg")
val_tbl$grobs[ind31][[1]][["gp"]] <- gpar(fill="azure1", col = "azure4", lwd=5)



#Coloring the 'YES' cell resulting from the uninformative SRS method
ind32 <- find_cell(val_tbl, 12, 5, "core-bg")
val_tbl$grobs[ind32][[1]][["gp"]] <- gpar(fill="darkgrey", col = "black", lwd=5)



grid.draw(val_tbl)



#Create grid without coloring cells
#grid.arrange(val_tbl)
```