MSDS 6370 Sampling Project

CitiBike Trip Duration Mean Estimation

Team members:

Christopher Boomhower, Alex Frye

Date: 08/02/2017

Introduction

As interests in data science continue to accelerate, more and more organizations are joining the open data movement, releasing datasets to the open public. However, this may not always be the case and manual effort may be required to obtain the data by outsiders. In other instances, data may be made available but its sheer volume may make it difficult for some individuals to gather the insights necessary given their computational resources available.

With these challenges in mind, we choose to assess survey sampling estimation methods while estimating mean CitiBike trip durations in NYC. The data utilized consists of CitiBike trip history data (CitiBike) collected and released by NYC Bike Share, LLC and Jersey Bike Share, LLC under Citi Bike's NYCBS Data Use Policy (CitiBike). Citi Bike is America's largest bike share program, with 10,000 bikes and 600 stations across Manhattan, Brooklyn, Queens, and Jersey City... 55 neighborhoods in all. As such, our data set's trip history includes all rental transactions conducted within the NYC CitiBike system from July 1st, 2013 to February 28th, 2014. These transactions amount to 5,562,293 trips within this time frame.

For this particular dataset, the data is made available. However, since this may not always be the case, it represents a valid test case for applying various survey methods to estimate a population parameter. In our case, we first use Simple Random Sampling (SRS), Proportional Allocation, and Neyman Allocation to estimate ***trip duration mean***, calculate the Design Effect (deff) for each complex method, obtain the new sample counts per the deff, and then proceed to compare each method's performance. After this, we transact 5 iterations of each of these sampling methodologies to derive confidence intervals for our estimation and compare these to the population true mean.

Outlier Removal & Stratification Design Analysis

In analyzing a Box Plot on trip duration values, we find extreme outliers present (Figure 2). With durations reaching up to 72 days in the most extreme instance, our team decided to rule out any observation with a duration greater than a 24-hour period. The likelihood of an individual sleeping overnight after their trip with the bike still checked out is much higher after the 24-hour period. This fact easily skews the results of this value, potentially harming any analysis done. We proceed with removing a total of 457 of the original ~5.5 million observations based on trip duration greater than 24 hours (86,400 seconds).

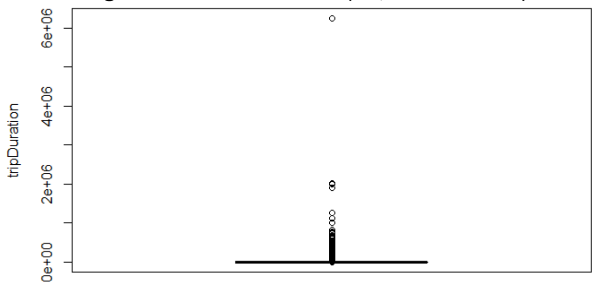


Figure 2 - Boxplot TripDuration with Outliers

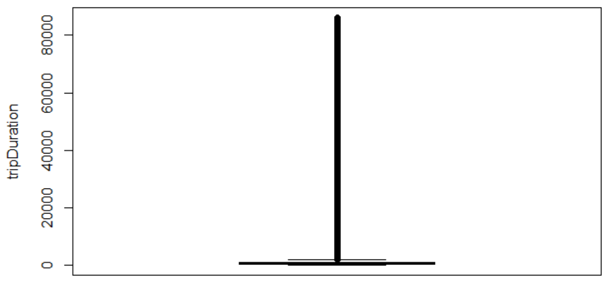


Figure 2 - Boxplot TripDuration after Outlier Removal

Although the data is still heavily left-skewed (Figure 2) after outlier removal, we move forward with this dataset for our analysis and begin to decide on criteria to define strata within our dataset. First, we choose to stratify by the Day of Week (Figure 4) – as one may suspect riders may have different usage tendencies during business days vs. weekend rides. Unfortunately, we do not quite see the variability across strata we want, so we take this one step further to stratify by Day of Week and Time of Day for a total of 35 strata (Figure 4). This produces far more interesting results, as we can clearly see that rider tendencies change when they start their trip in the morning vs evening, and other variations alike. Similarly, we can see at the same times day-to-day that slight variations in trip duration exist as well, such as those between Wednesday and Saturday afternoons. The number of long trips (>40000 seconds) is much higher on Saturdays than that of Wednesday afternoons, possibly indicating a difference in usage purpose (e.g. leisure vs lunch trips) during the working day. We are happy with this stratification approach, and move on to our sampling design tasks.

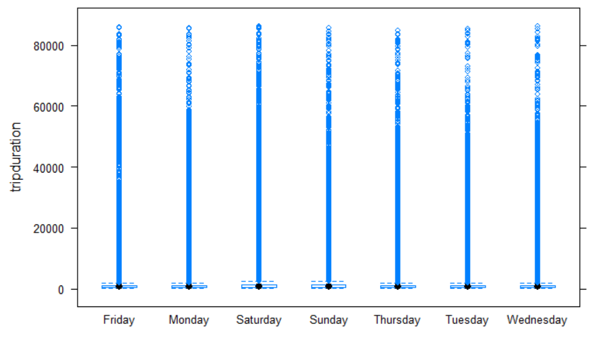


Figure 4 - Trip Duration by Day Boxplot

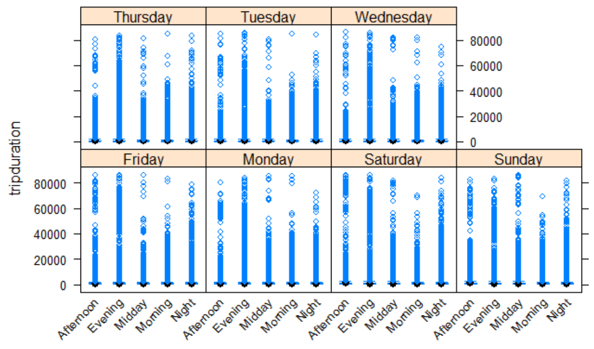


Figure 4 – Trip Duration by Day & Time of Day Boxplot

Task 1 – Sampling Design (Simple Random Sample)

First, we estimate the number of samples needed for a simple random sample (SRS) design. Although our goal is to perform a stratified sample for our estimate, we first need to compute a SRS design estimate to assist in calculation of Design Effect for the stratified design. Utilizing a margin of error (MOE) of 15 seconds, and 95% confidence threshold, we compute sample size $n_{0,srs}$ below.

$$n_{0,srs} = \frac{(Z_{\alpha/2}S)^2}{(moe)^2} = \frac{1.96^2 \times 1400.1478^2}{(15)^2} = 33471.6752 \approx 33472$$

Because the sample size is less than 10% of the original population, we may ignore the fpc adjustment in this calculation, leaving us with a sample size of 33472. With our sample size identified, we take a SRS of 33472 observations from the original population, less outliers.

With our SRS performed, we apply R's *svydesign()* & *svyMean()* procedures for trip duration mean estimation. Executing these procedures on our sampled data produces a mean estimate of 872.5 seconds and a standard error of 8.2137. As our true population mean of the original data less outliers is 860.978, we have a delta of an estimate 11.522 seconds larger than the true mean. This estimate will serve as a baseline for computing Design Effect in future sample design iterations.

Task 1 – Stratified Sampling Design (Proportional Allocation)

Our first stratified sampling method is Proportional Allocation. This method is chosen to ensure sample sizes across strata consisting of Day of Week and Time of Day remain consistent with the original population. We begin this process by computing strata sample sizes to produce a total sample size equal to $n_{0,srs}$ from our SRS design. Below, in Table 1, are the first 10 strata and steps taken to calculate a final rounded sample size.

Table 1. Proportional Sample Count Derivations (First 10 Strata)

DayOfWeek	TimeOfDay	Freq	N	p	SampSizeh	SampSizehRounded
<fctr>	<fctr>	<int>	<int>	<dbl>	<dbl>	<dbl>
Friday	Afternoon	163217	5561859	0.029345764	982.1894	982
Monday	Afternoon	134681	5561859	0.024215105	810.4560	810
Saturday	Afternoon	192714	5561859	0.034649206	1159.7062	1160
Sunday	Afternoon	176500	5561859	0.031733994	1062.1282	1062
Thursday	Afternoon	131989	5561859	0.023731094	794.2552	794
Tuesday	Afternoon	134233	5561859	0.024134556	807.7599	808
Wednesday	Afternoon	136051	5561859	0.024461426	818.7008	819
Friday	Evening	268323	5561859	0.048243402	1614.7312	1615
Monday	Evening	307180	5561859	0.055229735	1848.5777	1849
Saturday	Evening	201134	5561859	0.036163089	1210.3789	1210

Although not easily identified through the results above, the original rounded calculations produced a total sample size of 33474, 2 observations higher than our $n_{0,srs}$ of 33472. This is mitigated by adjusting the sample size calculation of all strata by a value of .072 prior to rounding. This adjustment is enough to force two strata sample size values, closest to .5 decimal range, to round down. With the adjustment factor in place, our total sample size matches $n_{0,srs}$. Next, we proceed to take a SRS without replacement from each stratum of the sizes calculated in the previous step, resulting in our stratified proportional sample dataset.

With our proportional samples taken, we again apply R’s *svydesign()* & *svyMean()* procedures for trip duration mean estimation. Executing these procedures on our sampled data produces a mean estimate of 868.4 seconds and a standard error of 8.1176. As our true population mean of the original data less outliers is 860.978, we have a delta of an estimate 7.422 seconds larger than the true mean.

After computing an estimate with the same sample size as was done for the SRS design, we may identify the Design Effect for our Proportional Allocation sample design. We do this by using SRS and Proportional SE values as follows, which results in a deff value of 0.9883.

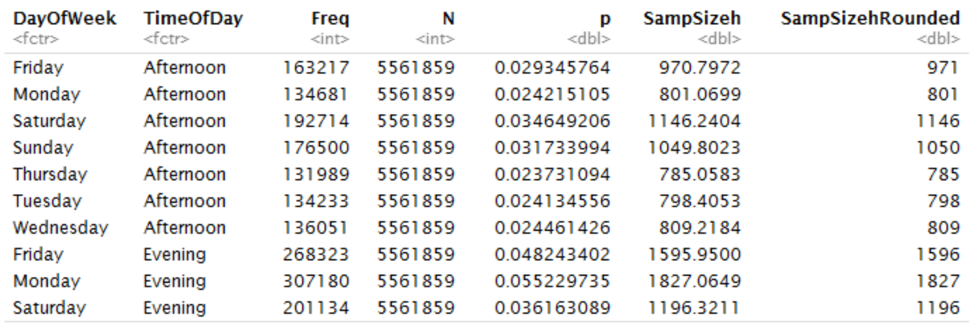
$$Deff_{complex} = \frac{V(\bar{y}_{complex})}{V(\bar{y}_{srs})} = \frac{8.1176}{8.2137} = 0.9883$$

Due to a design effect < 1, we may infer that Proportional Allocation is slightly more precise than SRS for this design. This deff value allows us to compute an appropriate sample size for Proportional Allocation to, with 95% confidence, produce a sample mean estimate within 15 seconds of the true mean.

$$n_{0,complex} = n_{0,srs} \times deff_{complex} = 33472 \times 0.9883 = 33080.25 \approx 33081$$

With our new sample size calculated, we proceed to compute our new sample strata sizes as was done before. The first 10 strata and their respective calculations may be found below in Table 2.

Table 2. Post-Deff Proportional Sample Count Derivations (First 10 Strata)



DayOfWeek	TimeOfDay	Freq	N	p	SampSizeh	SampSizehRounded
<fctr>	<fctr>	<int>	<int>	<dbl>	<dbl>	<dbl>
Friday	Afternoon	163217	5561859	0.029345764	970.7972	971
Monday	Afternoon	134681	5561859	0.024215105	801.0699	801
Saturday	Afternoon	192714	5561859	0.034649206	1146.2404	1146
Sunday	Afternoon	176500	5561859	0.031733994	1049.8023	1050
Thursday	Afternoon	131989	5561859	0.023731094	785.0583	785
Tuesday	Afternoon	134233	5561859	0.024134556	798.4053	798
Wednesday	Afternoon	136051	5561859	0.024461426	809.2184	809
Friday	Evening	268323	5561859	0.048243402	1595.9500	1596
Monday	Evening	307180	5561859	0.055229735	1827.0649	1827
Saturday	Evening	201134	5561859	0.036163089	1196.3211	1196

As can be seen from this output, our sample sizes in each stratum have decreased to produce our new total sample size. Once again, we proceed to take a SRS without replacement from each stratum of the sizes calculated in the previous step, resulting in our stratified proportional sample dataset.

With our post-deff proportional sample taken, we again apply R’s *svydesign()* & *svyMean()* procedures for trip duration mean estimation. Executing these procedures on our sampled data produces a mean estimate of 859.57 seconds and a standard error of 7.3067. As our true population mean of the original data less outliers is 860.978 seconds, we have a delta of an estimate 1.408 seconds less than the true mean.

<u>Task 1 – Stratified Sampling Design (Neyman Allocation)</u>

As our second sampling method, we choose to implement Neyman Allocation in order to account for *tripduration* variance by stratum. As was performed during Proportional Allocation, the strata sample sizes are computed such that the total number of samples equals our $n_{0,srs}$ calculated previously. The primary difference between our Proportional Allocation sampling calculations and Neyman Allocation computations is that our weight is derived from the product of stratum size and stratum standard deviation $\left(\frac{\text{Nh}\times \text{Sh}}{\text{Total NhSh}}\right)$. Neyman sample count derivation steps are portrayed for the first 10 strata in Table 3.

Table 3. Neyman Sample Count Derivations (First 10 Strata)

DayOfWeek <fctr>	TimeOfDay <fctr>	Freq <int>	stdDevs <dbl>	NhSh <dbl>	NhSh.ratio <dbl>	sampsRaw <dbl>	Neyman.samples <dbl>
Friday	Afternoon	163217	1331.090	217256438	0.028378647	949.8901	950
Monday	Afternoon	134681	1226.894	165239313	0.021584024	722.4605	722
Saturday	Afternoon	192714	1548.635	298443576	0.038983540	1304.8570	1305
Sunday	Afternoon	176500	1426.135	251712866	0.032879443	1100.5407	1101
Thursday	Afternoon	131989	1255.365	165694375	0.021643466	724.4501	724
Tuesday	Afternoon	134233	1186.993	159333697	0.020812616	696.6399	697
Wednesday	Afternoon	136051	1191.676	162128689	0.021177706	708.8602	709
Friday	Evening	268323	1574.462	422464410	0.055183490	1847.1018	1847
Monday	Evening	307180	1555.102	477696166	0.062398018	2088.5864	2089
Saturday	Evening	201134	1675.171	336933810	0.044011242	1473.1443	1473

By rounding the calculated strata sample counts in Table 3, we now have definitive Neyman sample sizes for each stratum. Comparing the sum of these values against $n_{0,srs}$ reveals rounding during calculation caused the sum to become two samples larger. Since our Neyman sample size is larger than the minimum calculated via $n_{0,srs}$, no further action would normally be required as we are ensured our Neyman sample sizes will obtain for us an estimate with no more than 15 seconds margin of error (with 95% confidence). However, because our intent is to also calculate Design Effect, we choose to manually adjust our total sample count to match $n_{0,srs}$. This is done by identifying the two strata with the smallest decimal value over 0.5 (*Tuesday Evening* and *Sunday Afternoon*) and rounding these strata sample sizes down.

Our next task is to sample each stratum per our derived Neyman sample counts. We implement simple random sampling within each stratum, without replacement, to accomplish this. After doing so, we merge our original population stratum size counts with our sample data for comparison using R's *survey* library *svydesign()* function for mean estimation and standard error calculation.

With our Neyman sample set complete and population strata sizes merged, we are ready to apply R's *svydesign()* & *svyMean()* procedures for trip duration mean estimation. Running these functions on our samples produces a trip duration mean estimation of 870.75 seconds with 8.4397 standard error. Again, our true population mean is 860.978 seconds, resulting in a delta of 9.772 seconds.

As was performed for Proportional Allocation, we consider it important to calculate the Neyman Allocation Design Effect to acquire the correct total sample size necessary to produce results like SRS. Calculating deff using SRS and Neyman Allocation SE values as follows results in a deff value of 1.0275.

$$Deff_{complex} = \frac{V(\bar{y}_{complex})}{V(\bar{y}_{srs})} = \frac{8.4397}{8.2137} = 1.0275$$

Being that the deff value of 1.0275 is approximately 1.0 (only 2.8% larger), it is apparent that Neyman Allocation produces an estimate which is approximately the same as SRS and only slightly less precise (deff value may change slightly with each new random sample set but floats around 1.0). Since the number reported leads us to believe a sample size 2.8% larger than $n_{0,srs}$ may be required for Neyman to obtain the same precision, we multiply $n_{0,srs}$ by 1.0275 to obtain our new total sample size.

$$n_{0,complex} = n_{0,srs} \times Deff_{complex} = 33472 \times 1.0275 = 34393.13 \approx 34394$$

With our new Neyman sample total defined, we proceed to compute our Neyman Allocation sample sizes amongst strata. This is performed as was done before Design Effect was calculated. The new strata sample size derivations are shown in Table 4 (First 10 strata are shown).

Table 4. Post-Deff Neyman Sample Count Derivations (First 10 Strata)

DayOfWeek <fctr>	TimeOfDay <fctr>	Freq <int>	stdDevs <dbl>	NhSh <dbl>	NhSh.ratio <dbl>	sampsRaw <dbl>	Neyman.samples <dbl>
Friday	Afternoon	163217	1331.090	217256438	0.028378647	976.0552	976
Monday	Afternoon	134681	1226.894	165239313	0.021584024	742.3609	742
Saturday	Afternoon	192714	1548.635	298443576	0.038983540	1340.7999	1341
Sunday	Afternoon	176500	1426.135	251712866	0.032879443	1130.8556	1131
Thursday	Afternoon	131989	1255.365	165694375	0.021643466	744.4054	744
Tuesday	Afternoon	134233	1186.993	159333697	0.020812616	715.8291	716
Wednesday	Afternoon	136051	1191.676	162128689	0.021177706	728.3860	728
Friday	Evening	268323	1574.462	422464410	0.055183490	1897.9810	1898
Monday	Evening	307180	1555.102	477696166	0.062398018	2146.1174	2146
Saturday	Evening	201134	1675.171	336933810	0.044011242	1513.7227	1514

When again checking our *Neyman.samples* total against the Post-deff sample size calculated, we find that rounding error has again affected our total, this time resulting in a total one sample short (34393 instead of 34394). We identify the *Thursday Midday* stratum for rounding up and do so since it contains the largest decimal value less than 0.5.

Finally, with new Neyman Allocation strata sample sizes calculated, we sample each stratum per its respective sample count and estimate the mean as was done previously. Doing so produces an estimate of 858.362 seconds with a standard error of 7.193.

Task 1 – Comparison

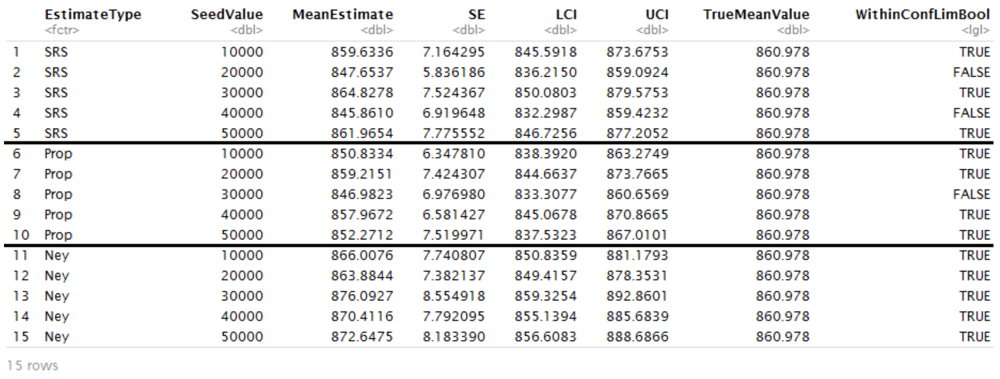
Based on the performance of all three sampling methods, there are some considerations we must review before selecting a winning method. Namely, since we want to use deff values to define sample size, we consider only post-deff adjusted sampling outcomes in our comparisons. With that in mind, factors such as estimate accuracy, standard error, and required sample size will be considered.

SRS mean estimation of 872.5 seconds is 11.5 seconds over the true population mean, compared to only 1.4 seconds under for Proportional Allocation and 2.6 under for Neyman Allocation. Then SRS, Proportional, and Neyman SE values are 8.214, 7.307, and 7.193 respectively. Based on the fact that SRS total sample count must be at least 33472 to obtain no more than 15 seconds margin of error and that Proportional Allocation must sample only 33081 CitiBike trips (391 less than SRS) whereas Neyman Allocation must sample 34394 (922 more than SRS) to obtain similar results, we choose Proportional Allocation as the best sampling design for these data. It produces the most accurate estimate of the three methods with only marginally higher SE than Neyman (And with 1313 fewer samples compared to Neyman).

Task 2 & Final Conclusions – Compute 5 Estimate Confidence Intervals and Compare to Actual Mean

To further assess our three sample designs & estimate for mean trip duration, we compute the confidence interval of 5 different random seeds for each design. In Table 5 are the results of these tests further supporting our previous findings.

Table 5. Actual Mean Comparison across 5 unique random seeds



	EstimateType <fctr>	SeedValue <dbl>	MeanEstimate <dbl>	SE <dbl>	LCI <dbl>	UCI <dbl>	TrueMeanValue <dbl>	WithinConfLimBool <lgl>
1	SRS	10000	859.6336	7.164295	845.5918	873.6753	860.978	TRUE
2	SRS	20000	847.6537	5.836186	836.2150	859.0924	860.978	FALSE
3	SRS	30000	864.8278	7.524367	850.0803	879.5753	860.978	TRUE
4	SRS	40000	845.8610	6.919648	832.2987	859.4232	860.978	FALSE
5	SRS	50000	861.9654	7.775552	846.7256	877.2052	860.978	TRUE
6	Prop	10000	850.8334	6.347810	838.3920	863.2749	860.978	TRUE
7	Prop	20000	859.2151	7.424307	844.6637	873.7665	860.978	TRUE
8	Prop	30000	846.9823	6.976980	833.3077	860.6569	860.978	FALSE
9	Prop	40000	857.9672	6.581427	845.0678	870.8665	860.978	TRUE
10	Prop	50000	852.2712	7.519971	837.5323	867.0101	860.978	TRUE
11	Ney	10000	866.0076	7.740807	850.8359	881.1793	860.978	TRUE
12	Ney	20000	863.8844	7.382137	849.4157	878.3531	860.978	TRUE
13	Ney	30000	876.0927	8.554918	859.3254	892.8601	860.978	TRUE
14	Ney	40000	870.4116	7.792095	855.1394	885.6839	860.978	TRUE
15	Ney	50000	872.6475	8.183390	856.6083	888.6866	860.978	TRUE

15 rows

Once again, we have used the post-deff adjusted sample sizes for each iteration and respective estimation design method. As expected, SRS estimation produced the most confidence intervals which did not include the true mean; 2 of 5 tests executed did not contain true mean value. Proportional Allocation was second, with 1 of 5 tests not containing the true mean value. Although there was one iteration in this test, the Upper Confidence Limit only missed the true mean value by 0.3211 seconds. Finally, Neyman Allocation produces results with 100% of Confidence Intervals containing the true mean value. Although Neyman Allocation produces the best results in our 5 iterations, we must once again consider factors such as standard error and required sample size. Neyman Allocation has the benefit of a larger sample size but contains a larger SE on average across the 5 test iterations in comparison to the Proportional Allocation iterations. Additionally, when analyzing our estimate for the missed Proportional Allocation iteration, we can compute the delta between the estimate and True Mean Value ($860.978 - 846.9823 = 13.9957\ seconds$). This delta falls within our 15 second MOE defined previously, further supporting Proportional Allocation as the design method of choice for future sample estimates of trip duration.

References

CitiBike. *Data License Agreement*. 2017. Website. 19 7 2017. <https://www.citibikenyc.com/data-sharing-policy>.

—. *System Data*. 2017. Website. 19 07 2017. <https://www.citibikenyc.com/system-data>.

Code Appendix

```r
require(survey)
require(dplyr)
require(lattice)

setwd("../Analysis/Data")
if(!file.exists("popData.rds")){
    pop.data <- NULL
    for(i in 1:length(list.files())){
        data.frag <- read.csv(paste("dataset",i,".csv",sep = ""), header = FALSE)
        pop.data <- rbind(pop.data, data.frag)
    }
    colnames(pop.data) <- c("tripduration",
                            "starttime",
                            "stoptime",
                            "start_station_id",
                            "start_station_name",
                            "start_station_latitude",
                            "start_station_longitude",
                            "end_station_id",
                            "end_station_name",
                            "end_station_latitude",
                            "end_station_longitude",
                            "bikeid",
                            "usertype",
                            "birth year",
                            "gender",
                            "LinearDistance",
                            "DayOfWeek",
                            "TimeOfDay",
                            "HolidayFlag",
                            "PRCP",
                            "SNOW",
                            "TAVE",
                            "TMAX",
                            "TMIN")
    saveRDS(pop.data, "popData.rds")
    rm(data.frag)
} else pop.data <- readRDS("popData.rds")

    # BoxPlot tripDuration - Heavy Outliers!
boxplot(pop.data$tripduration, ylab = "tripDuration")

    # How Many Greater than 24 hours?
print(nrow(pop.data[pop.data$tripduration>86400,]))

    # Remove > 24 Hours
pop.data <- pop.data[pop.data$tripduration<86400,c("tripduration", "DayOfWeek", "TimeOfDay")]

boxplot(pop.data$tripduration, ylab = "tripDuration")

bwplot(tripduration ~ DayOfWeek, data = pop.data)
bwplot(tripduration ~ TimeOfDay | DayOfWeek, data = pop.data, scales=list(x=list(rot=45)))

trueMean <- mean(pop.data$tripduration)

print(paste0("The true mean of the population less outliers is: ", trueMean))

   # Compute the sample size for a SRS
stdev = sd(pop.data$tripduration)
MOE   = 15 #seconds
N = nrow(pop.data)
print(stdev)

n0srs = ceiling((1.96^2*stdev^2)/(MOE^2))

print(paste0('With a sample size only ',round(n0srs/N*100,4), '% of the original population, w
e ignore fpc.'))

print(paste0('Sample Size needed for an estimate of the mean trip duration within 15 seconds,
with 95% confidence, is ' , n0srs, '.'))

SrsMeanEstimate<-function(Seed, SampSize, printOutput= TRUE){
set.seed(Seed)

pop.data.SRSSampled = sample_n(pop.data,SampSize)

if(printOutput == TRUE){
  print(nrow(pop.data.SRSSampled))
  print(bwplot(tripduration ~ DayOfWeek, data = pop.data.SRSSampled))
  print(bwplot(tripduration ~ TimeOfDay | DayOfWeek, data = pop.data.SRSSampled, scales=list(x
=list(rot=45))))
}

mydesign <- svydesign(id = ~1, data = pop.data.SRSSampled)

srsMean = svymean(~tripduration, design = mydesign)
srsSE = SE(srsMean)
srsCI = confint(srsMean)

rm(pop.data.SRSSampled)
rm(mydesign)

return(list(as.numeric(srsMean[1]),
            as.numeric(srsSE),
            as.numeric(srsCI[1]),
            as.numeric(srsCI[2])
           )
      )
}

srsMean <- SrsMeanEstimate(n0srs, n0srs)

print(paste('The Mean Estimate =', srsMean[[1]]))

print(paste('The Standard Error =', srsMean[[2]]))



PropMeanEstimate<-function(Seed, SampSize, SampSizeAdj, printOutput= TRUE){

set.seed(Seed)

  # Identify Frequency of DayOfWeek Stratum
PropFreq <- as.data.frame(table(pop.data[,c("DayOfWeek", "TimeOfDay")]))
names(PropFreq)[1] = 'DayOfWeek'
PropFreq

PropFreq$N = nrow(pop.data)
PropFreq$p = PropFreq$Freq/PropFreq$N
PropFreq$SampSizeh = (PropFreq$p * SampSize)-SampSizeAdj  #adjustment of SampSizeAdj in order
to round down the closest to .5 stratum due to an original rounded sample size 2 higher than n
0srs
PropFreq$SampSizehRounded = round(PropFreq$SampSizeh)


pop.data.PropSampled <- NULL

for (i in 1:nrow(PropFreq)){
  pop.data.PropSampled<-rbind(pop.data.PropSampled,
                            sample_n(pop.data[(pop.data$DayOfWeek == PropFreq[i,"DayOfWeek"]
                                                 & pop.data$TimeOfDay ==   PropFreq[i,"TimeOf
Day"])
                                               ,]
                                     ,PropFreq[i,"SampSizehRounded"]
                                     )
                            )

}

if(printOutput == TRUE){
  print(PropFreq)
  print(nrow(pop.data.PropSampled))
  print(bwplot(tripduration ~ DayOfWeek, data = pop.data.PropSampled))
  print(bwplot(tripduration ~ TimeOfDay | DayOfWeek, data = pop.data.PropSampled, scales=list(
x=list(rot=45))))
}

mydesign <- svydesign(id = ~1, strata = ~paste(DayOfWeek,TimeOfDay), data = pop.data.PropSampl
ed)

propMean = svymean(~tripduration, design = mydesign)
propSE = SE(propMean)
propCI = confint(propMean)

rm(pop.data.PropSampled)
rm(mydesign)
propCI = confint(propMean)
return(list(as.numeric(propMean[1]),
            as.numeric(propSE),
            as.numeric(propCI[1]),
            as.numeric(propCI[2])
           )
      )
}

propMean <- PropMeanEstimate(n0srs, n0srs, .072)

print(paste('The Mean Estimate =', propMean[[1]]))

print(paste('The Standard Error =', propMean[[2]]))
```

```r
deffProp = as.numeric(propMean[[2]]/srsMean[[2]])
deffProp



n0prop = n0srs*deffProp
n0prop

n0prop = ceiling(n0prop)
n0prop



propMean <- PropMeanEstimate(n0srs, n0prop, -.01)

print(paste('The Mean Estimate =', propMean[[1]]))

print(paste('The Standard Error =', propMean[[2]]))



  # Identify Frequency of DayOfWeek Stratum
NeyFreq <- as.data.frame(table(pop.data[,c("DayOfWeek", "TimeOfDay")]))
names(NeyFreq)[1] = 'DayOfWeek'

stdDevs <- tapply(pop.data$tripduration, paste(pop.data$TimeOfDay, pop.data$DayOfWeek), sd)
NhSh <- NeyFreq$Freq * stdDevs
NhSh.ratio <- NhSh/sum(NhSh)
sampsRaw <- round(n0srs)*NhSh.ratio
Neyman.samples <- round(sampsRaw)

data.frame(NeyFreq,
           stdDevs = as.vector(stdDevs),
           NhSh = as.vector(NhSh),
           NhSh.ratio = as.vector(NhSh.ratio),
           sampsRaw = as.vector(sampsRaw),
           Neyman.samples = as.vector(Neyman.samples))



paste("n0srs = ", n0srs)

paste("Neyman Samples Total = ", sum(Neyman.samples))

paste("Sunday Afternoon before change: ", Neyman.samples[4])

Neyman.samples[4] = Neyman.samples[4] - 1
paste("Sunday Afternoon after change: ", Neyman.samples[4])

paste("Tuesday Evening before change: ", Neyman.samples[13])

Neyman.samples[13] = Neyman.samples[13] - 1
paste("Tuesday Evening after change: ", Neyman.samples[13])

paste("Neyman Samples Total = ", sum(Neyman.samples))



pop.data.Neyman <- data.frame(DayOfWeek = pop.data$DayOfWeek, TimeOfDay = pop.data$TimeOfDay,
tripduration = pop.data$tripduration)
pop.data.Neyman$N <- NA
```

```r
for(i in 1:nrow(NeyFreq)){
    pop.data.Neyman$N[paste(pop.data.Neyman$TimeOfDay, pop.data.Neyman$DayOfWeek) ==
                          attributes(Neyman.samples[i])] <- Neyman.samples[i]
}

head(pop.data.Neyman)

tail(pop.data.Neyman)



set.seed(n0srs)
pop.data.NeySampled <- NULL

for (i in 1:nrow(NeyFreq)){
  pop.data.NeySampled<-rbind(pop.data.NeySampled,
                            sample_n(pop.data.Neyman[paste(pop.data.Neyman$TimeOfDay,
                                                           pop.data.Neyman$DayOfWeek) ==
                                                         attributes(Neyman.samples[i]),],
                                     Neyman.samples[i]))

}

NeySamp <- merge(pop.data.NeySampled, NeyFreq, by = c("TimeOfDay","DayOfWeek"))



  # Create SurveyDesign
mydesign <- svydesign(id = ~1, strata = ~paste(DayOfWeek,TimeOfDay), data = NeySamp) # Dropped
since no correction needed: , fpc = ~Freq)

#mydesign <- postStratify(design = mydesign, strata = ~DayOfWeek, population = ToDFreq)

NeyMean <- svymean(~tripduration, design = mydesign)
NeyMean



deffNey = as.numeric(SE(NeyMean)/srsMean[[2]])
deffNey



n0Ney = n0srs*deffNey
n0Ney

n0Ney = ceiling(n0Ney)
n0Ney

sample.Neyman <- function(seed.value){
    set.seed(seed.value)

      # Identify Frequency of DayOfWeek and TimeOfDay Stratum
    NeyFreq <- as.data.frame(table(pop.data[,c("DayOfWeek", "TimeOfDay")]))
    names(NeyFreq)[1] = 'DayOfWeek'

    stdDevs <- tapply(pop.data$tripduration, paste(pop.data$TimeOfDay, pop.data$DayOfWeek), sd
)
    NhSh <- NeyFreq$Freq * stdDevs
    NhSh.ratio <- NhSh/sum(NhSh)
    sampsRaw <- round(n0Ney)*NhSh.ratio
    Neyman.samples <- round(sampsRaw)
    return(list(data.frame(NeyFreq,
                stdDevs = as.vector(stdDevs),
                NhSh = as.vector(NhSh),
                NhSh.ratio = as.vector(NhSh.ratio),
                sampsRaw = as.vector(sampsRaw),
                Neyman.samples = as.vector(Neyman.samples)),
            NeyFreq, Neyman.samples))
}

Neyman.survey <- function(seed.value, Freq, Neyman.samples){
    set.seed(seed.value)

    pop.data.Neyman <- data.frame(DayOfWeek = pop.data$DayOfWeek, TimeOfDay = pop.data$TimeOfD
ay,
                                  tripduration = pop.data$tripduration)
    pop.data.Neyman$N <- NA

    for(i in 1:nrow(Freq)){
    pop.data.Neyman$N[paste(pop.data.Neyman$TimeOfDay, pop.data.Neyman$DayOfWeek) ==
                          attributes(Neyman.samples[i])] <- Neyman.samples[i]
    }

    pop.data.NeySampled <- NULL

    for (i in 1:nrow(Freq)){
      pop.data.NeySampled<-rbind(pop.data.NeySampled,
                                 sample_n(pop.data.Neyman[paste(pop.data.Neyman$TimeOfDay,
                                                               pop.data.Neyman$DayOfWeek) ==
                                                             attributes(Neyman.samples[i]),],
                                          Neyman.samples[i]))

    }

    NeySamp <- merge(pop.data.NeySampled, Freq, by = c("TimeOfDay","DayOfWeek"))

      # Create SurveyDesign
    mydesign <- svydesign(id = ~1, strata = ~paste(DayOfWeek,TimeOfDay), data = NeySamp)

    NeyMean <- svymean(~tripduration, design = mydesign)

    NeySE = SE(NeyMean)

    NeyCI <- confint(svymean(~tripduration, design = mydesign))

    return(list(as.numeric(NeyMean[1]),
                as.numeric(NeySE),
                as.numeric(NeyCI[1]),
                as.numeric(NeyCI[2])))
}

output <- sample.Neyman(n0srs)
NeymanSamps <- output[[1]]
NeyFreq <- output[[2]]
Neyman.samples <- output[[3]]

NeymanSamps

rm(output)
```

```r
paste("n0Ney = ", n0Ney)

paste("Post-Deff Neyman Samples Total = ", sum(Neyman.samples))

paste("Thursday Midday before change: ", Neyman.samples[19])

Neyman.samples[19] = Neyman.samples[19] + 1
paste("Thursday Midday after change: ", Neyman.samples[19])



NeyMean <- Neyman.survey(n0srs, NeyFreq, Neyman.samples)

print(paste('The Mean Estimate =', NeyMean[[1]]))

print(paste('The Standard Error =', NeyMean[[2]]))



SeedList <- c(10000, 20000, 30000, 40000, 50000)

df<- NULL

  #SRS Seed Executions
for (seed in SeedList){
  srsEstimate <- SrsMeanEstimate(seed, n0srs, FALSE)
  srsEstimate <- data.frame('SRS', seed, srsEstimate)
  names(srsEstimate) <- c("EstimateType","SeedValue", "MeanEstimate", "SE", "LCI", "UCI")
  df<- rbind(df,
             srsEstimate
            )
}



  #Prop Seed Executions
for (seed in SeedList){
  PropEstimate <- PropMeanEstimate(seed, n0prop, -.01, FALSE)
  PropEstimate <- data.frame('Prop', seed, PropEstimate)
  names(PropEstimate) <- c("EstimateType","SeedValue", "MeanEstimate", "SE", "LCI", "UCI")
  df<- rbind(df,
             PropEstimate
            )
}



#Ney Seed Executions
for (seed in SeedList){
 NeyEstimate <- Neyman.survey(seed, NeyFreq, Neyman.samples)
 NeyEstimate <- data.frame('Ney', seed, NeyEstimate)
 names(NeyEstimate) <- c("EstimateType","SeedValue", "MeanEstimate", "SE", "LCI", "UCI")
 df<- rbind(df,
            NeyEstimate
           )
}



  #Add True Mean Value, in-line with estimates
df$TrueMeanValue <- trueMean
```

```r
  #Add Bool Value for whether the Conf Limit contains the True Mean Value
df$WithinConfLimBool <- df$LCI <= df$TrueMeanValue & df$UCI >= df$TrueMeanValue

  #Print Results
print(df)
```