Stratified Random Sampling Simulation and Parameter Estimation

Maggie

This report presents a simulation study on the performance of different estimators in stratified random sampling under the influence of outliers, including the Neyman Sample Mean Estimator, the Wang-Xu Hybrid Estimator, and the Proposed Hybrid Estimator. The study evaluates bias, variance, and mean squared error (MSE) across multiple scenarios with varying levels of outliers and different numbers of strata to determine the most robust estimation method.

Simulation Design
A finite population consisting of $100,000$ units was generated and divided into H strata. The data within each stratum was drawn from a normal distribution $N(\mu_h, \sigma_h^2)$, where μ_h and σ_h^2 varied across strata to simulate heterogeneity. A fixed proportion of each stratum was replaced with extreme values (outliers) generated from a heavy-tailed $t - distribution$ with 1 degree of freedom , shifted by the mean of the respective stratum. Outliers were introduced randomly across all strata to assess their impact on estimation.

Data was simulated with **5% and 10% outliers** and across **3, 5, and 8 strata**, resulting in six different simulation cases based on the proportion of outliers and the number of strata. Each case was simulated **100 times** to generate a sufficient number of sample statistics for each estimator. The study aimed to compute the **Mean Squared Error (MSE), Bias, and Variance** for each case, allowing a comparison of the following estimators under the influence of outliers and evaluating how stratum sizes affect their estimates.

- Neyman Sample Mean Estimator: $\hat{Y} = \sum_{h=1}^{H} P_h \bar{y_h}$
- Wang Xu Hybrid Estimator: $\hat{Y}_{trimmed} = \sum_{h=1}^{H} P_h \left[w\bar{y} + (1-w)T_h\right]$
- Proposed Hybrid Estimator: $\hat{Y}_{weighted} = \sum_{h=1}^{H} P_h \left[w\bar{y} + (1-w)W_h\right]$

Sampling cases:
Case 1: Outliers proportion = 5% Number of strata = 3 Number of simulations = 100
Case 2: Outliers proportion = 5% Number of strata = 5 Number of simulations = 100
Case 3: Outliers proportion = 5% Number of strata = 8 Number of simulations = 100
Case 4: Outliers proportion = 10% Number of strata = 3 Number of simulations = 100
Case 5: Outliers proportion = 10% Number of strata = 5 Number of simulations = 100
Case 6: Outliers proportion = 10% Number of strata = 8 Number of simulations = 100

Below is a tabular representation of five out of the **100** simulations for each case. The naming convention follows the pattern: **Case-1-2**, where **Case-1** refers to the first case, and **2** indicates the second simulation within that case.

Table 1: Estimated Parameters from Simulation

	Neyman_estimates	Wang_Xu_Hybrid_estimates	Proposed_Hybrid_estimates
case-1-1	1.675540	1.530402	1.530746
case-1-2	1.552211	1.531041	1.532803
case-1-3	1.431684	1.535462	1.535797
case-1-4	1.522013	1.541000	1.542513
case-1-5	1.677263	1.537351	1.535815

	Neyman_estimates	Wang_Xu_Hybrid_estimates	Proposed_Hybrid_estimates
case-2-1	4.727031	5.411823	5.410567
case-2-2	5.312279	5.385065	5.382796
case-2-3	5.346904	5.377466	5.377178
case-2-4	5.361737	5.383362	5.382944
case-2-5	5.398415	5.389211	5.387028
case-3-1	7.302779	7.640346	7.641205
case-3-2	7.693708	7.681240	7.675827
case-3-3	7.695182	7.681252	7.680439
case-3-4	7.636400	7.657651	7.655277
case-3-5	7.682307	7.663499	7.659547
case-4-1	1.476276	1.511213	1.512891
case-4-2	2.318188	1.536751	1.538773
case-4-3	1.851490	1.543556	1.541862
case-4-4	2.196156	1.520166	1.517468
case-4-5	1.572293	1.542053	1.542508
case-5-1	5.379672	5.393857	5.390236
case-5-2	5.444140	5.406370	5.406800
case-5-3	5.471715	5.387075	5.379906
case-5-4	5.460752	5.422810	5.420034
case-5-5	5.304758	5.385921	5.388025
case-6-1	7.479171	7.659409	7.657121
case-6-2	7.734387	7.622998	7.621461
case-6-3	9.806518	7.649664	7.650925
case-6-4	7.670216	7.671487	7.665782
case-6-5	8.657707	7.627235	7.623592

After conducting the simulations, I computed the **Mean Squared Error (MSE), Variance, and Bias** for each estimator across all listed cases. The results are presented below, providing a comparative analysis of the estimators' performance under different conditions.

Table 2: Neyman Estimator

	Case-1	Case-2	Case-3	Case-4	Case-5	Case-6
Varience	0.052849	0.854925	16.505676	4.450224	0.488581	0.933919
Biase	0.006925	0.066531	0.597781	-0.285573	0.045389	-0.153237
MS Error	0.052897	0.859351	16.863018	4.531776	0.490641	0.957400

Table 3: Wang Xu Hybrid Estimator

	Case-1	Case-2	Case-3	Case-4	Case-5	Case-6
Varience	0.000095	0.000327	0.000494	0.00010	0.000276	0.000419
Biase	-0.000674	-0.000054	-0.001657	-0.00026	0.003181	0.004885
MS Error	0.000095	0.000327	0.000496	0.00010	0.000286	0.000443

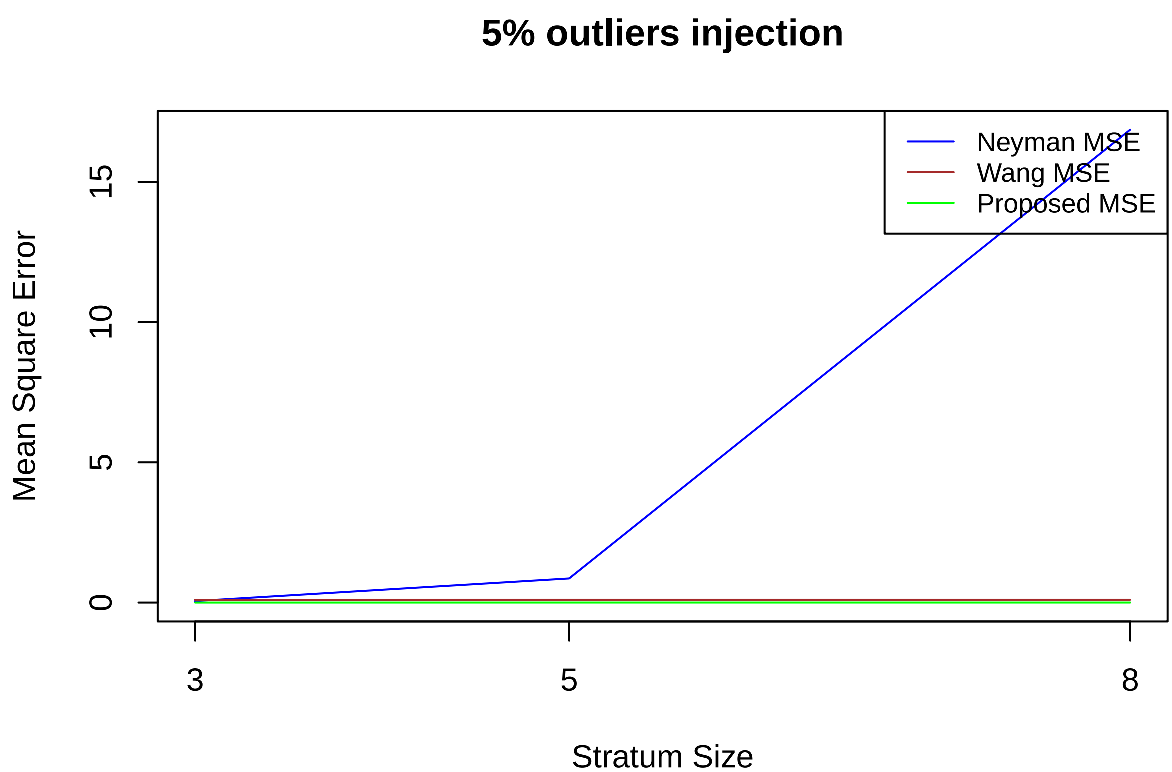
Table 4: Proposed Hybrid Estimator

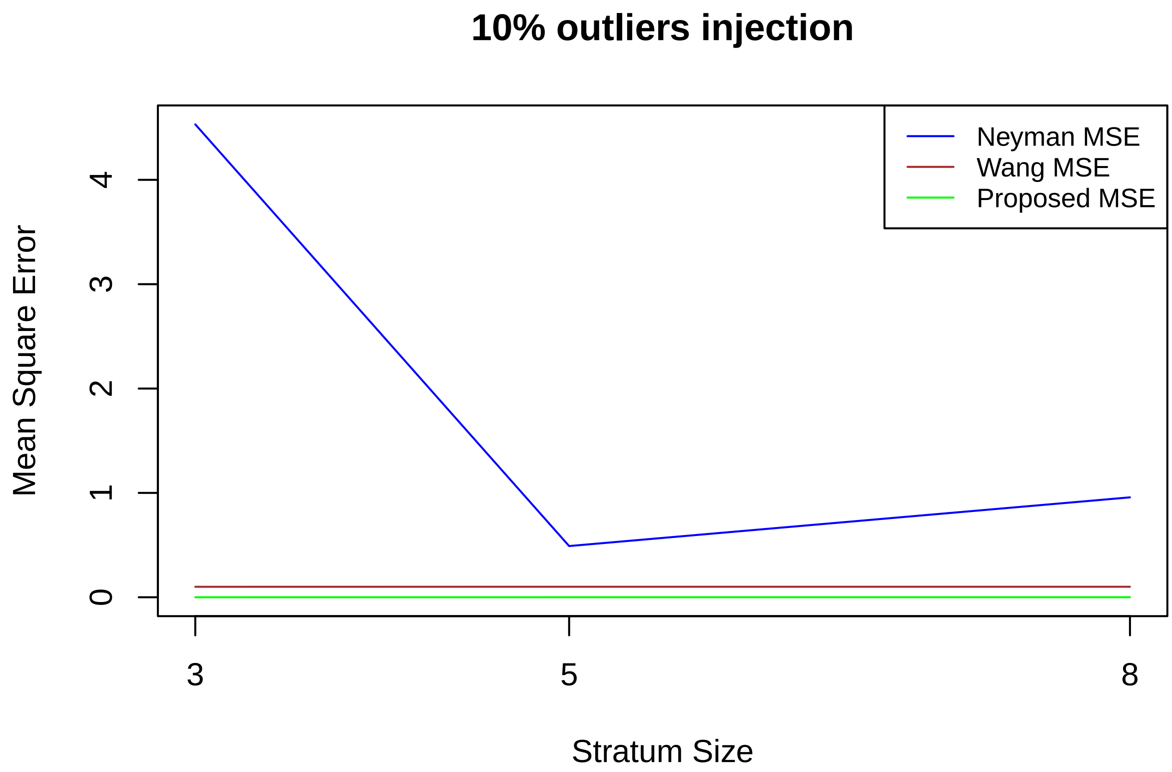
	Case-1	Case-2	Case-3	Case-4	Case-5	Case-6
Varience	0.000096	0.000333	0.000511	0.000110	0.000280	0.000437
Biase	-0.000736	-0.000051	-0.001724	-0.000268	0.003373	0.004898
MS Error	0.000096	0.000333	0.000514	0.000110	0.000291	0.000461

Table 5: Estimated Parameters- one time Simulation

	Case-1	Case-2	Case-3	Case-4	Case-5	Case-6
Neyman_estimates	1.550633	5.352097	7.624483	1.722806	5.316595	7.718108
Wang_Xu_Hybrid_estimates	1.515134	5.387096	7.682260	1.532333	5.397232	7.654469
Proposed_Hybrid_estimates	1.513301	5.389660	7.678370	1.533518	5.396322	7.657172

The results indicate that the **Neyman estimator** is significantly affected by outliers, as evidenced by its extreme variance across different cases. In contrast, the **hybrid estimators** (including the **Proposed Hybrid Estimator** and the **Trimmed Hybrid Estimator**) demonstrate greater robustness to outliers. The lower variance and mean squared error (MSE) of these hybrid estimators suggest that they provide more stable and reliable estimates under the influence of outliers, making them preferable in scenarios where data contamination is a concern.





Note:
In the graphs presented, the **Wang-Zu estimator's Mean Square Error (MSE) values** were **vertically shifted** by a constant (**0.1**) **solely for visualization purposes**. This adjustment was necessary because the **Neyman estimator**, being highly sensitive to outliers, caused a significant stretch of the Y-axis scale, thereby compressing the **Wang-Zu** and **Proposed Hybrid estimators** into overlapping lines. As a result, it became difficult to visually distinguish between the two robust estimators.

To address this, the **Wang-Zu MSE values** were shifted **upward** to separate the lines and make them visible. The **Proposed Hybrid estimator** was plotted using its original, unshifted MSE values.

This vertical shift was applied **only to the Wang-Zu estimator** to improve clarity, and **does not affect the interpretation or conclusions** about the relative stable performance of the estimators under the influence of outliers..

Code chunk:

```
stratified_simulation = function(n_strata,
                                 key_parameters,
                                 inject_propotion = 0.1,
                                 t_df = 1,
                                 sample_size = 50000,
                                 sample_set = "m"){

  # initializing key containers and values
  strata_set = list()
  strata_mixed_set = list()
  strata_outliers_set = list()
```

```r
  stratum_index_replace_set = list()

  # Generate random stratified normal distributions
  for(i in 1:n_strata){
    strata_set[[paste0("stratum-",i)]] = rnorm(key_parameters[[i]][1],
                                               key_parameters[[i]][2],
                                               key_parameters[[i]][3])
  }

  # Generate outliers set from a t distribution shifted by the mean of the respective stratum
  for(i in 1:n_strata){
    stratum_size = key_parameters[[i]][1]
    stratum_outliers_count = inject_propotion * stratum_size
    stratum_mean = mean(strata_set[[paste0("stratum-",i)]])

    strata_outliers_set[[paste0("outlier_set-",i)]]
    = rt(stratum_outliers_count, t_df) + stratum_mean
    stratum_index_replace_set[[paste0("index_set", i)]]
    = sample(1:stratum_size, stratum_outliers_count)
  }

  # Inject outliers into the simulated stratum
  for(i in  1:n_strata){
    strata_mixed_set[[paste0("stratummixed-",i)]]
    = strata_set[[paste0("stratum-",i)]][-stratum_index_replace_set[[paste0("index_set", i)]]]
    strata_mixed_set[[paste0("stratummixed-",i)]]
    = c(strata_mixed_set[[paste0("stratummixed-",i)]], strata_outliers_set[[paste0("outlier_set-",i)]])
  }


  #  Sampling process

  if(sample_set == "m"){
    set_determinent = strata_mixed_set
  } else if(sample_set == "u") {
    set_determinent = strata_set
  }

  stratified_sample = c()
  for(i in 1:n_strata){
    stratum_sample_size = (key_parameters[[i]][1] / 100000) * sample_size
    stratified_sample = c(stratified_sample, sample(set_determinent[[i]], stratum_sample_size))
  }
  # The function returns  the stratified sample, the stratum set with and with out outliers
  return(list(stratified_sample, strata_set, strata_mixed_set))

}
#__________________________ Estimators section _________________________________

# The Neyman Estimator
neyman_estimator = function(strata){
  strata_propotion_set = (sapply(strata, length)) / 100000
  strata_mean_set = sapply(strata, mean)
```

```r
  neyman_estimate = sum(strata_propotion_set * strata_mean_set)

  return(neyman_estimate)
}

# The weighted mean estimator for each stratum
weighted_mean = function(strata_nooutliers, strata_outliers){
  strata_mean_set = sapply(strata_nooutliers, mean)
  strata_sd_set = sapply(strata_nooutliers, sd)

  weight_vec = lapply(seq_along(strata_outliers), function(i){
    weight_sub = 1 / (1 + (abs(strata_outliers[[i]]
                             - strata_mean_set[names(strata_mean_set)[i]])
                           / strata_sd_set[names(strata_sd_set)[i]]))
    return(weight_sub)
  })

  w_mean = sapply(seq_along(weight_vec), function(i){
    return(sum(strata_outliers[[i]] * weight_vec[[i]]) / sum(weight_vec[[i]]))
  })

  return(w_mean)
}

# The trimmed mean  estimator for  each stratum
trim_mean = function(strata, trim_weight){
  trim_fun = function(x, trim_weight){
    trim_size = round(length(x) * trim_weight)
    trimmed_values = sort(x)[(trim_size + 1):(length(x) - trim_size)]
    return(trimmed_values)
  }

  t_strata_set = lapply(strata, trim_fun, trim_weight = trim_weight)
  t_mean = sapply(t_strata_set, mean)

  return(t_mean)
}


# The hybrid estimator capable of deriving computing the Neyman,
# Wang Xu hybrid, and  proposed  hybrid estimators

hybrid_estimator = function(strata_nooutliers, strata_outliers, have_outliers
                            = TRUE, trimmean = FALSE, trim_weight = 0.05){
  strata_propotion_set = (sapply(strata_nooutliers, length)) / 100000

  if(have_outliers & trimmean == FALSE){
    strata_weighted_mean_set = weighted_mean(strata_nooutliers, strata_outliers)
    hybrid_estimate = sum(strata_weighted_mean_set * strata_propotion_set)
  }
  else if(have_outliers & trimmean == TRUE){
    strata_trimmed_mean_set = trim_mean(strata_outliers, trim_weight = trim_weight)
    hybrid_estimate = sum(strata_trimmed_mean_set * strata_propotion_set)
```

```r
  }
  else{
    hybrid_estimate = neyman_estimator(strata_nooutliers)
  }

  return(hybrid_estimate)
}


#_____________________________ simulation ____________________________________

case_names = c("case-1-1", "case-1-2", "case-1-3", "case-1-4", "case-1-5",
               "case-2-1", "case-2-2", "case-2-3", "case-2-4", "case-2-5",
               "case-3-1", "case-3-2", "case-3-3", "case-3-4", "case-3-5",
               "case-4-1", "case-4-2", "case-4-3", "case-4-4", "case-4-5",
               "case-5-1", "case-5-2", "case-5-3", "case-5-4", "case-5-5",
               "case-6-1", "case-6-2", "case-6-3", "case-6-4", "case-6-5")

propotion_option = c(0.05, 0.1)
dist_option = list(list(c(32145, 1, 2), c(28734, 1.5, 3), c(39121,2,4)),
                   list(c(18064, 5,3), c(25491,5,8), c(15678,8,2), c(22315, 6.3, 5), c(18452, 3,5)),
                   list(c(12341, 12, 4), c(15321, 5, 10), c(13456, 7,10), c(10765, 8,3),
                        c(11984, 9,2), c(14213, 9,5), c(9572,4,7), c(12348,7,4)))

Neyman_estimates = c()
Wang_Xu_Hybrid_estimates =  c()
Proposed_Hybrid_estimates =  c()
summary_list_dt = list()
for (i in propotion_option) {
  Injection_propotion = i

  for (j in dist_option) {
    for(i in 1:100){
      Distribution_parameters_and_stratum_size = j
      number_of_strata = length(j)

      b = stratified_simulation(n_strata = number_of_strata,
                                key_parameters = Distribution_parameters_and_stratum_size,
                                inject_propotion = Injection_propotion)

      estimates = c(neyman_estimator(b[[3]]), hybrid_estimator(b[[2]], b[[3]]),
                    hybrid_estimator(b[[2]], b[[3]], trimmean = TRUE))
      Neyman_estimates = c(Neyman_estimates, estimates[1])
      Wang_Xu_Hybrid_estimates =  c(Wang_Xu_Hybrid_estimates, estimates[2])
      Proposed_Hybrid_estimates =  c(Proposed_Hybrid_estimates, estimates[3])

      max_vec = c(sapply(b[[2]], max), sapply(b[[3]], max))
      min_vec = c(sapply(b[[2]], min), sapply(b[[3]], min))
      mean_vec = c(sapply(b[[2]], mean), sapply(b[[3]], mean))
      sd_vec = c(sapply(b[[2]], sd), sapply(b[[3]], sd))
      row_names = c(names(b[[2]]),names(b[[3]]))

      summary_dt = data.frame(max_vec, min_vec, mean_vec, sd_vec, row.names = row_names)
```

```
      colnames(summary_dt) =  c("Max_value", "Min_Value", "Mean", "SD")
      summary_list_dt = append(summary_list_dt, list(summary_dt))
    }

  }
}


estimates_set = list(Neyman_estimates, Wang_Xu_Hybrid_estimates, Proposed_Hybrid_estimates)
var_set = c()
biased_set = c()
mu_set = c(1.53488, 5.391395,7.65506, 1.53488, 5.391395,7.65506)

for(i in estimates_set){
  ofset = 0
  for(j in 1:6){
    a = i[(1+ofset):(100 + ofset)]
    var_set = c(var_set,  mean((a-mean(a))**2))
    biased_set = c(biased_set, (mean(a) - mu_set[j]))
    ofset = ofset + 100
  }
}
mse_set = var_set + (biased_set)**2


neyman_var_set = var_set[1:6]
neyman_biased_set = biased_set[1:6]
neyman_mse_set = mse_set[1:6]

wang_var_set = var_set[7:12]
wang_biased_set = biased_set[7:12]
wang_mse_set = mse_set[7:12]

proposed_var_set = var_set[13:18]
proposed_biased_set = biased_set[13:18]
proposed_mse_set = mse_set[13:18]

neyman_dt = rbind(neyman_var_set, neyman_biased_set, neyman_mse_set)
colnames(neyman_dt) = c("Case-1", "Case-2", "Case-3", "Case-4", "Case-5", "Case-6")
rownames(neyman_dt) = c("Varience", "Biase", "MS Error")
neyman_dt

round(var_set, 6)
round(biased_set, 6)
round(mse_set, 6)

print(neyman_mse_set)
print(wang_mse_set)
print(proposed_mse_set)


for(i in 1:6){
  value = c(value, neyman_mse_set[i])
```

```r
  value = c(value, wang_mse_set[i])
  value = c(value, proposed_mse_set[i])
}


# __________________________________________Graphing ________________
library(ggplot2)
library(dplyr)


data <- data.frame(
  Category = rep(c("Case_1", "Case_2", "Case_3", "Case_4", "Case_5", "Case_6"), each = 3),
  Subcategory = rep(c("Neyman", "Wang", "Proposed"), times = 6),
  Value = value
)


data <- data %>%
  group_by(Category) %>%
  mutate(Percentage = Value / sum(Value) * 100)
data

ggplot(data, aes(x = "", y = Percentage, fill = Subcategory)) +
  geom_bar(stat = "identity", width = 1, color = "white") +
  coord_polar(theta = "y") +
  facet_wrap(~ Category) +
  scale_fill_manual(values = c("#1b9e77", "#d95f02", "#7570b3")) +
  labs(
    title = "Comparison of Estimators Across Cases",
    fill = "Estimator"
  ) +
  theme_void() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    legend.position = "bottom"
  )
```