R_Code for the Simulation

Maggie

2025-04-02

```
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()
  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
```

```r
  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)
  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)
  }
```

```r
  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)
  }
  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]
```

```r
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])
  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(
```

```r
    plot.title = element_text(hjust = 0.5, face = "bold"),
    legend.position = "bottom"
  )
```