Sim Parallel Poisson Gamma

Baoxing Liu

2025-11-07

```r
# ============================================================
# Parallel Simulation: Comparing SRS vs Clustered Kernel Estimator
# Parallelized over B using parLapply (Windows Compatible)
# ============================================================

library(tidyverse)
library(parallel)

# ----------------------------
# 1. Discrete Triangular Kernel
# ----------------------------
discrete_triang_kernel <- function(xi, x, h, a) {
  num <- (a + 1)^h - abs(xi - x)^h
  num[num < 0] <- 0
  P_ah <- (2 * a + 1) * (a + 1)^h - 2 * sum((0:a)^h)
  return(num / P_ah)
}

discrete_triang_estimator <- function(xi, x, h, a) {
  mean(discrete_triang_kernel(xi, x, h, a))
}

# ----------------------------
# 2. Leave-One-Cluster-Out Estimator
# ----------------------------
loco_triang_estimator <- function(x, data, cluster_to_exclude, h, a) {
  train_data <- data[data$cluster != cluster_to_exclude, ]
  mean(discrete_triang_kernel(train_data$X, x, h, a))
}

# ----------------------------
# 3. Cross-Validation Objective
# ----------------------------
cv_objective <- function(h, data, a) {
  clusters <- unique(data$cluster)
  n <- nrow(data)
  sum1 <- 0
  sum2 <- 0

  for (c in clusters) {
    Cc_indices <- which(data$cluster == c)
    mc <- length(Cc_indices)
```

```r
    for (i in Cc_indices) {
      sum1 <- sum1 + loco_triang_estimator(data$X[i], data, c, h, a)
    }

    x_points <- min(data$X):max(data$X)
    f_squared_sum <- sum(sapply(x_points, function(x) {
      loco_triang_estimator(x, data, c, h, a)^2
    }))

    sum2 <- sum2 + mc * f_squared_sum
  }

  cv_val <- -2/n * sum1 + 1/n * sum2
  return(cv_val)
}

# ============================================================
# 4. Simulation Setup
# ============================================================
set.seed(123)

lambda <- 5
xs <- 0:30
k_grid <- c(10, 20, 30)
m <- 10
r_grid <- c(1, 3, 5, 8, 10)
rho <- lambda / (lambda + r_grid)
h_grid <- seq(0.1, 3.0, by = 0.2)
a <- 1
B <- 500   # number of Monte Carlo replicates
n_cores <- max(1, detectCores() - 1)  # leave one core free

# ============================================================
# 5. Main Simulation Loop over (k, r)
# ============================================================

mse_all <- list()
mise_all <- list()

for (k in k_grid) {
  for (l in seq_along(r_grid)) {
    r <- r_grid[l]
    f_true <- dnbinom(xs, size = r, prob = r / (r + lambda))

    # ------------------------------------------------------------
    # Parallel Monte Carlo Replicates (Windows-safe)
    # ------------------------------------------------------------
    cl <- makeCluster(n_cores)
    clusterExport(cl, varlist = c("lambda", "k", "m", "r", "xs", "a",
                                  "h_grid", "f_true", "discrete_triang_kernel",
                                  "discrete_triang_estimator",
                                  "loco_triang_estimator", "cv_objective"),
                  envir = environment())
```

```r
clusterEvalQ(cl, library(tidyverse))

res_list <- parLapply(cl, 1:B, function(b) {
  Zc <- rgamma(k, shape = r, rate = r)
  X <- unlist(lapply(Zc, function(z) rpois(m, lambda * z)))
  df_clust <- data.frame(cluster = rep(1:k, each = m), X = X)
  df_srs <- data.frame(cluster = 1:(k*m), X = X)

  cv_scores_srs <- sapply(h_grid, function(h) cv_objective(h, df_srs, a))
  cv_scores_clust <- sapply(h_grid, function(h) cv_objective(h, df_clust, a))

  best_h_srs <- h_grid[which.min(cv_scores_srs)]
  best_h_clust <- h_grid[which.min(cv_scores_clust)]

  fhat_srs_vals <-
    sapply(xs, function(x) discrete_triang_estimator(df_srs$X, x, best_h_srs, a))
  fhat_clust_vals <-
    sapply(xs, function(x) discrete_triang_estimator(df_clust$X, x, best_h_clust, a))

  se_srs <- (fhat_srs_vals - f_true)^2
  se_clust <- (fhat_clust_vals - f_true)^2
  ise_srs <- sum(se_srs)
  ise_clust <- sum(se_clust)

  list(
    se_srs = se_srs,
    se_clust = se_clust,
    ise_srs = ise_srs,
    ise_clust = ise_clust
  )
})
stopCluster(cl)

# ------------------------------------------------------------
# Combine Parallel Results
# ------------------------------------------------------------
se_srs_mat <- do.call(cbind, lapply(res_list, `[[`, "se_srs"))
se_clust_mat <- do.call(cbind, lapply(res_list, `[[`, "se_clust"))
ise_srs <- sapply(res_list, `[[`, "ise_srs")
ise_clust <- sapply(res_list, `[[`, "ise_clust")

mse_srs <- rowMeans(se_srs_mat)
mse_clust <- rowMeans(se_clust_mat)

mse_df <- data.frame(
  x = xs,
  MSE_SRS = mse_srs,
  MSE_Clustered = mse_clust,
  r = r,
  rho = round(rho[1], 3),
  k = k
) %>%
  pivot_longer(cols = c(MSE_SRS, MSE_Clustered),
```

```
                    names_to = "Method", values_to = "MSE")

    mse_all[[paste0("r", r, "_k", k)]] <- mse_df

    mise_tbl <- tibble(
      r = r,
      rho = round(rho[l], 3),
      k = k,
      Method = c("SRS", "Clustered"),
      MISE = c(mean(ise_srs), mean(ise_clust)),
      SD_ISE = c(sd(ise_srs), sd(ise_clust))
    )
    mise_all[[paste0("r", r, "_k", k)]] <- mise_tbl

    cat("Finished r =", r, ", k =", k, "\n")
  }
}
```

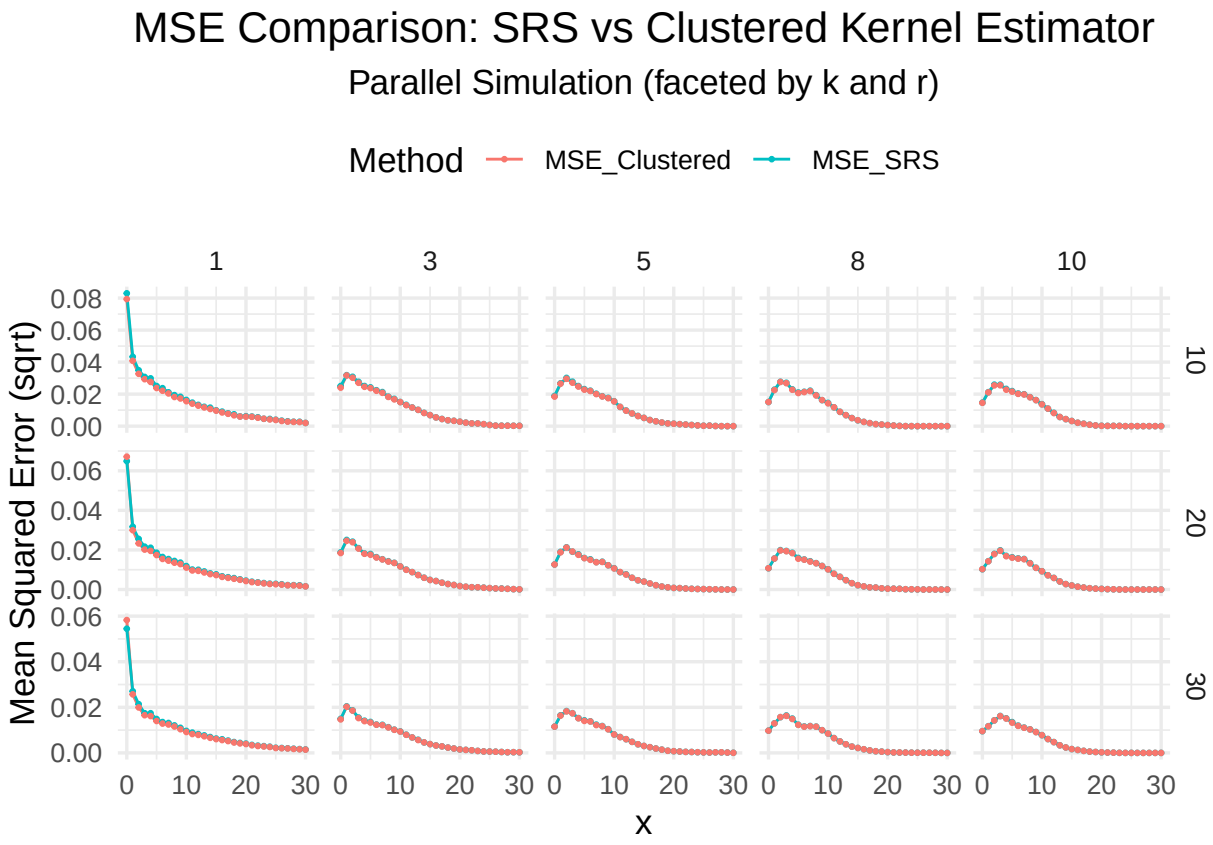
```
Finished r = 1 , k = 10
Finished r = 3 , k = 10
Finished r = 5 , k = 10
Finished r = 8 , k = 10
Finished r = 10 , k = 10
Finished r = 1 , k = 20
Finished r = 3 , k = 20
Finished r = 5 , k = 20
Finished r = 8 , k = 20
Finished r = 10 , k = 20
Finished r = 1 , k = 30
Finished r = 3 , k = 30
Finished r = 5 , k = 30
Finished r = 8 , k = 30
Finished r = 10 , k = 30
```

```
# ================================================================
# 6. Combine and Save Results
# ================================================================
mse_all_df <- bind_rows(mse_all)
mise_all_df <- bind_rows(mise_all)

write.csv(mse_all_df, "mse_results.csv", row.names = FALSE)
write.csv(mise_all_df, "mise_results.csv", row.names = FALSE)

# ================================================================
# 7. Faceted Plot for MSE
# ================================================================
ggplot(mse_all_df, aes(x = x, y = sqrt(MSE), color = Method)) +
  geom_line() + geom_point(size = 0.5) +
  facet_grid(k ~ r, scales = "free_y") +
  theme_minimal(base_size = 13) +
  labs(
    title = "MSE Comparison: SRS vs Clustered Kernel Estimator",
    subtitle = "Parallel Simulation (faceted by k and r)",
```

```
    x = "x",
    y = "Mean Squared Error (sqrt)"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5),
    plot.subtitle = element_text(hjust = 0.5),
    legend.position = "top"
  )
```




```
# ============================================================
# 8. Table for MISE
# ============================================================
mise_all_df %>%
  arrange(k, r, Method) %>%
  knitr::kable(caption = "Summary of MISE Results (Parallelized Simulation)")
```

Table 1: Summary of MISE Results (Parallelized Simulation)

r	rho	k	Method	MISE	SD_ISE
1	0.833	10	Clustered	0.0141036	0.0115133
1	0.833	10	SRS	0.0156460	0.0119597
3	0.625	10	Clustered	0.0067713	0.0050893
3	0.625	10	SRS	0.0070146	0.0053062
5	0.500	10	Clustered	0.0059582	0.0046810

r	rho	k	Method	MISE	SD_ISE
5	0.500	10	SRS	0.0061222	0.0049576
8	0.385	10	Clustered	0.0052471	0.0043660
8	0.385	10	SRS	0.0053352	0.0044335
10	0.333	10	Clustered	0.0047637	0.0039301
10	0.333	10	SRS	0.0048834	0.0041438
1	0.833	20	Clustered	0.0085525	0.0061142
1	0.833	20	SRS	0.0088341	0.0061654
3	0.625	20	Clustered	0.0039299	0.0033960
3	0.625	20	SRS	0.0040285	0.0034640
5	0.500	20	Clustered	0.0029886	0.0024160
5	0.500	20	SRS	0.0030303	0.0024304
8	0.385	20	Clustered	0.0027000	0.0020744
8	0.385	20	SRS	0.0027465	0.0020820
10	0.333	20	Clustered	0.0025309	0.0022076
10	0.333	20	SRS	0.0025719	0.0022170
1	0.833	30	Clustered	0.0062594	0.0046575
1	0.833	30	SRS	0.0061811	0.0045077
3	0.625	30	Clustered	0.0024024	0.0019643
3	0.625	30	SRS	0.0024548	0.0019773
5	0.500	30	Clustered	0.0022544	0.0017863
5	0.500	30	SRS	0.0022889	0.0018133
8	0.385	30	Clustered	0.0018171	0.0014473
8	0.385	30	SRS	0.0018354	0.0014646
10	0.333	30	Clustered	0.0016871	0.0013599
10	0.333	30	SRS	0.0017071	0.0013679