S23 NSERC USRA Simulation Studies for Underreported Infectious Disease Time Series Models

Sindi Bebeziqi (Under the Supervision of Dr. Justin Slater)

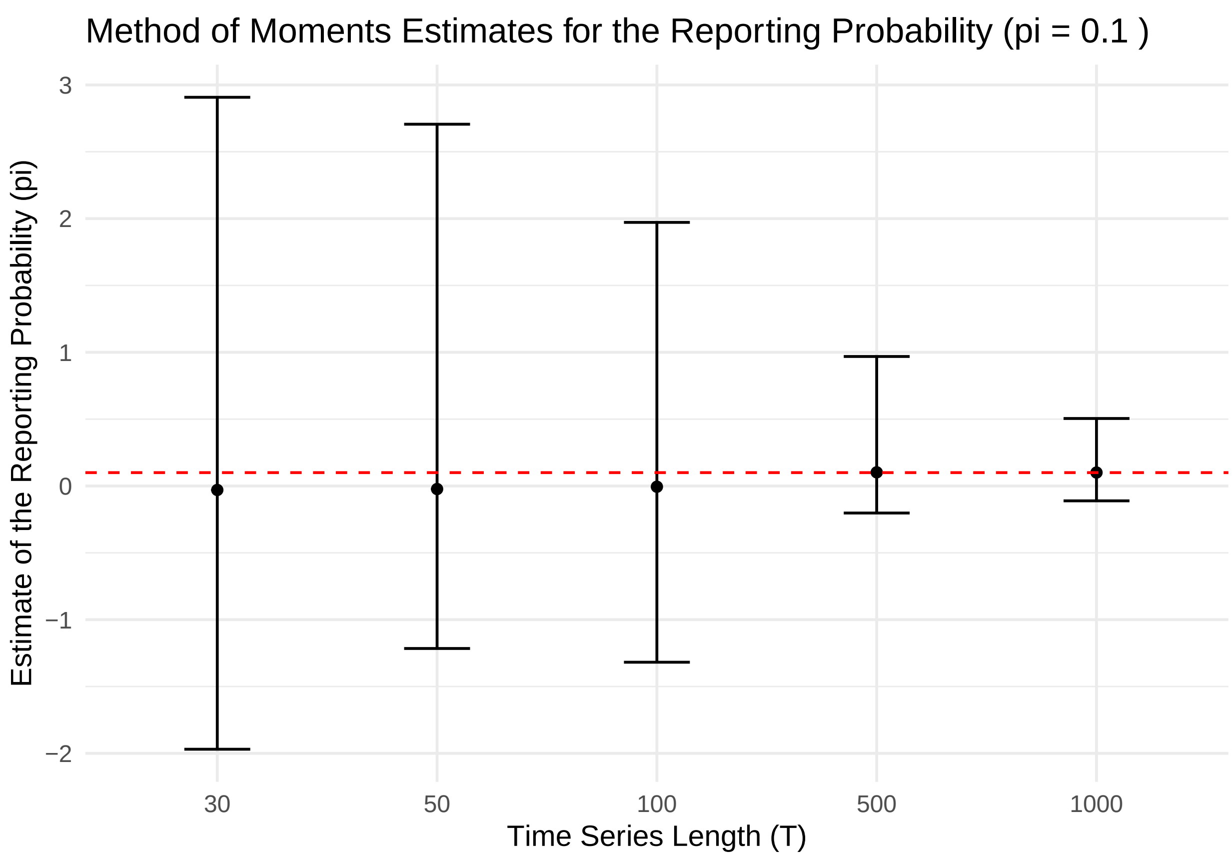
2023-08-17

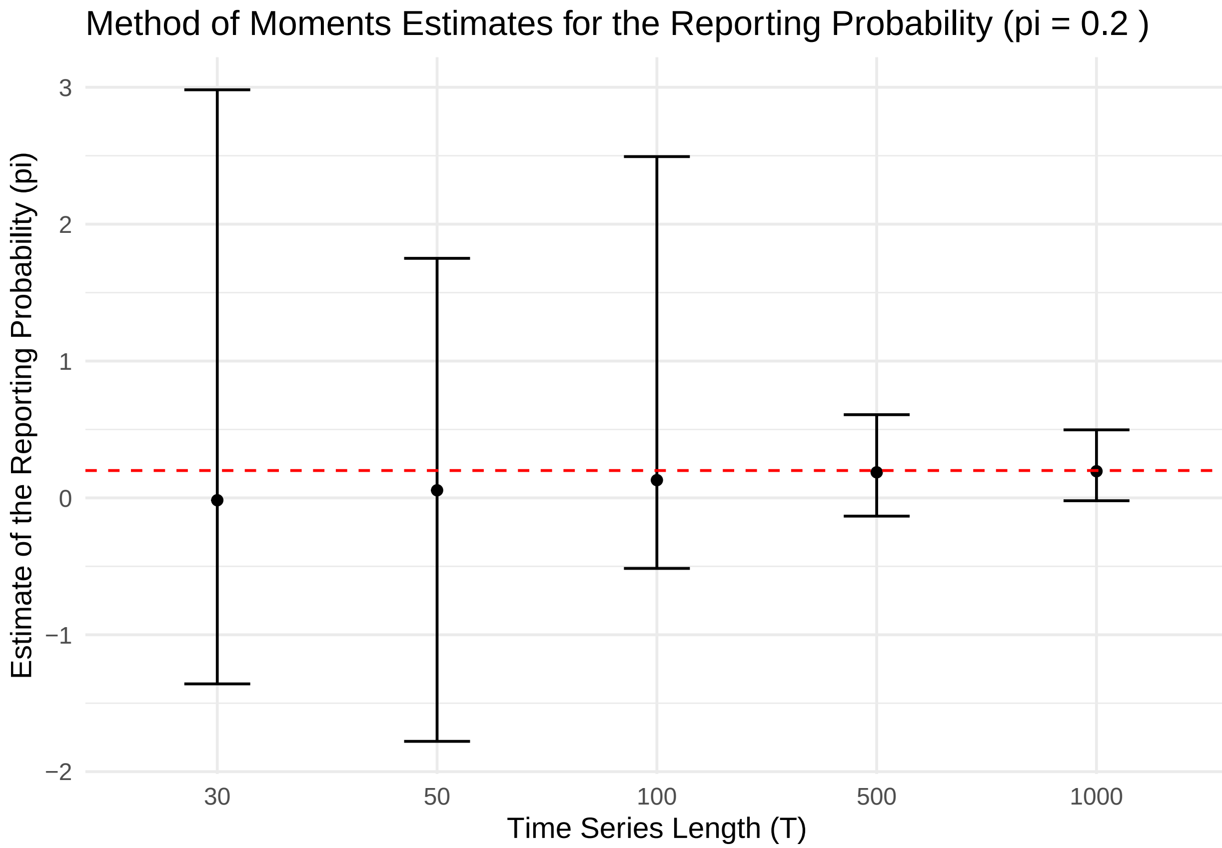
```r
library(ggplot2)
library(tinytex)
library(abind)

# Set parameter values
phi <- 0.8   # Autoregressive parameter
nu <- 5   # Mean parameter

# create sequence of pi's
pi_seq <- seq(0.1, 1, 0.1)

# initialize the objects that will be used to store all the results
df_pi <- c()
df_phi <- c()
df_nu <- c()

# for each pi...
for (pi in pi_seq) {

  # Define number of simulations and time series lengths
  num_simulations <- 1000
  T_values <- c(30, 50, 100, 500, 1000)  # Updated values of T

  # Function to compute Method of Moments estimators
  compute_estimators <- function(z) {
    rho_1 <- acf(z, lag.max = 1, plot = FALSE)$acf[2]   # Autocorrelation at lag 1

    # Handle missing or NaN values
    if (is.na(rho_1) || is.nan(rho_1)) {
      rho_1 <- 0.01
    }

    # Method of Moments estimators
    if (abs(1 - mean(z) / var(z)) < 0.001 || var(z) <= 0.001) {
      phi_hat <- 0.001  # Assign phi directly as 0.001
    } else {
      phi_hat <- (1 / rho_1) * (1 - mean(z) / var(z))
    }

    # Check if variance is close to zero
```

```r
  if (var(z) <= 0.001) {
    pi_hat <- 0.001  # Assign a small positive value to pi_hat
  } else {
    pi_hat <- 1 - var(z) / mean(z) * (1 - rho_1 / phi_hat)
  }

  nu_hat <- (1 - phi_hat) * mean(z) / pi_hat

  return(c(pi_hat, phi_hat, nu_hat))
}

# Function to simulate and estimate
simulate_and_estimate <- function(pi, phi, nu, T) {
  z <- rep(0, T)   # Initialize time series
  z[1] <- 5   # Initialize Z_1 at 5

  # Simulate time series
  for (i in 2:T) {
    z[i] <- rpois(1, lambda = nu + phi * z[i-1])
  }

  # Thin the time series using binomial thinning
  y <- rbinom(T, z, pi)

  # Check for missing or NaN values
  if (any(is.na(y)) || any(is.nan(y))) {
    return(c(NA, NA, NA))
  }

  # Compute Method of Moments estimators
  estimators <- compute_estimators(y)

  return(estimators)
}

# Initialize matrix to store Method of Moments estimates and quantiles
estimates_matrix <- matrix(0, nrow = length(T_values), ncol = 3)
quantiles_matrix <- array(0, dim = c(length(T_values), 3, 3))

# Perform simulation study for different values of T
for (j in 1:length(T_values)) {
  T <- T_values[j]
  mom_estimators <- matrix(0, nrow = num_simulations, ncol = 3)

  # Run simulations
  for (i in 1:num_simulations) {
    mom_estimators[i, ] <- simulate_and_estimate(pi, phi, nu, T)
  }

  # Calculate average error
  avg_error <- abs(colMeans(mom_estimators) - c(phi, pi, nu))

  # Check for "Inf" or "NaN" average error estimates and replace with numeric values
```

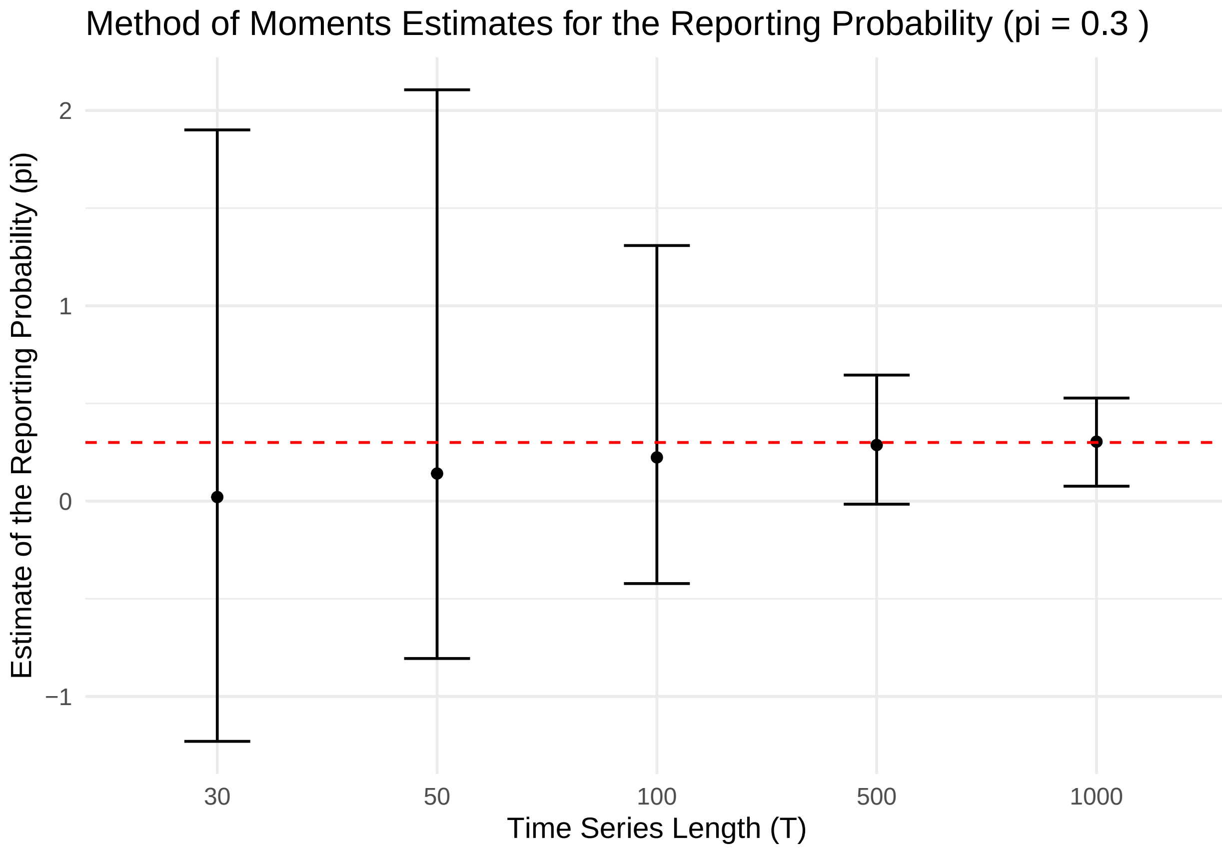
```
    avg_error[is.infinite(avg_error)] <- NA
    avg_error[is.nan(avg_error)] <- NA

    # Calculate quantiles
    valid_simulations <- complete.cases(mom_estimators)
    quantiles <- apply(mom_estimators[valid_simulations, ], 2, quantile, probs = c(0.025, 0.5, 0.975))

    # Store Method of Moments estimates and quantiles in the matrices
    estimates_matrix[j, ] <- colMeans(mom_estimators, na.rm = TRUE)
    quantiles_matrix[j, , ] <- quantiles

    # Print average error and quantiles
    #cat("T =", T, "Average Error:", avg_error, "\n")
    #cat("T =", T, "Quantiles:", quantiles, "\n")
  }

  # Create data frames for plotting
  df_pi_pi <- data.frame(T = as.character(T_values), Estimate = quantiles_matrix[, 2, 1],
                         Lower = quantiles_matrix[, 1, 1], Upper = quantiles_matrix[,3 , 1])
  df_phi_pi <- data.frame(T = as.character(T_values), Estimate = quantiles_matrix[, 2, 2],
                          Lower = quantiles_matrix[, 1, 2], Upper = quantiles_matrix[, 3, 2])
  df_nu_pi <- data.frame(T = as.character(T_values), Estimate = quantiles_matrix[, 2, 3],
                         Lower = quantiles_matrix[, 1, 3], Upper = quantiles_matrix[, 3, 3])

  # Append the results for this pi value to the corresponding data frame
  df_pi <- abind(df_pi, df_pi_pi, along = 3) # added along = 3
  df_phi <- abind(df_phi, df_phi_pi, along = 3) # along = 3
  df_nu <- abind(df_nu, df_nu_pi, along = 3) # along = 3
}

# Now we have 1 matrix for each value of pi.
# We will plot each one separately.

# Plot for estimate of pi for each value of pi
pi_plots <- list()
for (i in 1:length(pi_seq)) {
  pi_plots[[i]] <- ggplot(as.data.frame(df_pi[,,i]), aes(x = T, y = as.numeric(Estimate))) + # added as
    geom_point() +
    geom_errorbar(aes(ymin = as.numeric(Lower), ymax = as.numeric(Upper)), width = 0.3) + #as.numeric t
    geom_hline(yintercept = pi_seq[i], linetype = "dashed", color = "red") +
    labs(title = paste("Method of Moments Estimates for the Reporting Probability (pi =", pi_seq[i], ")
         x = "Time Series Length (T)", y = "Estimate of the Reporting Probability (pi)") +
    theme_minimal() +
    scale_x_discrete(limits = as.character(T_values))
}

# Plot for estimate of phi for each value of pi
phi_plots <- list()
for (i in 1:length(pi_seq)) {
  phi_plots[[i]] <- ggplot(as.data.frame(df_phi[,,i]), aes(x = T, y = as.numeric(Estimate))) +
    geom_point() +
    geom_errorbar(aes(ymin = as.numeric(Lower), ymax = as.numeric(Upper)), width = 0.3) +
    geom_hline(yintercept = phi, linetype = "dashed", color = "red") +
```

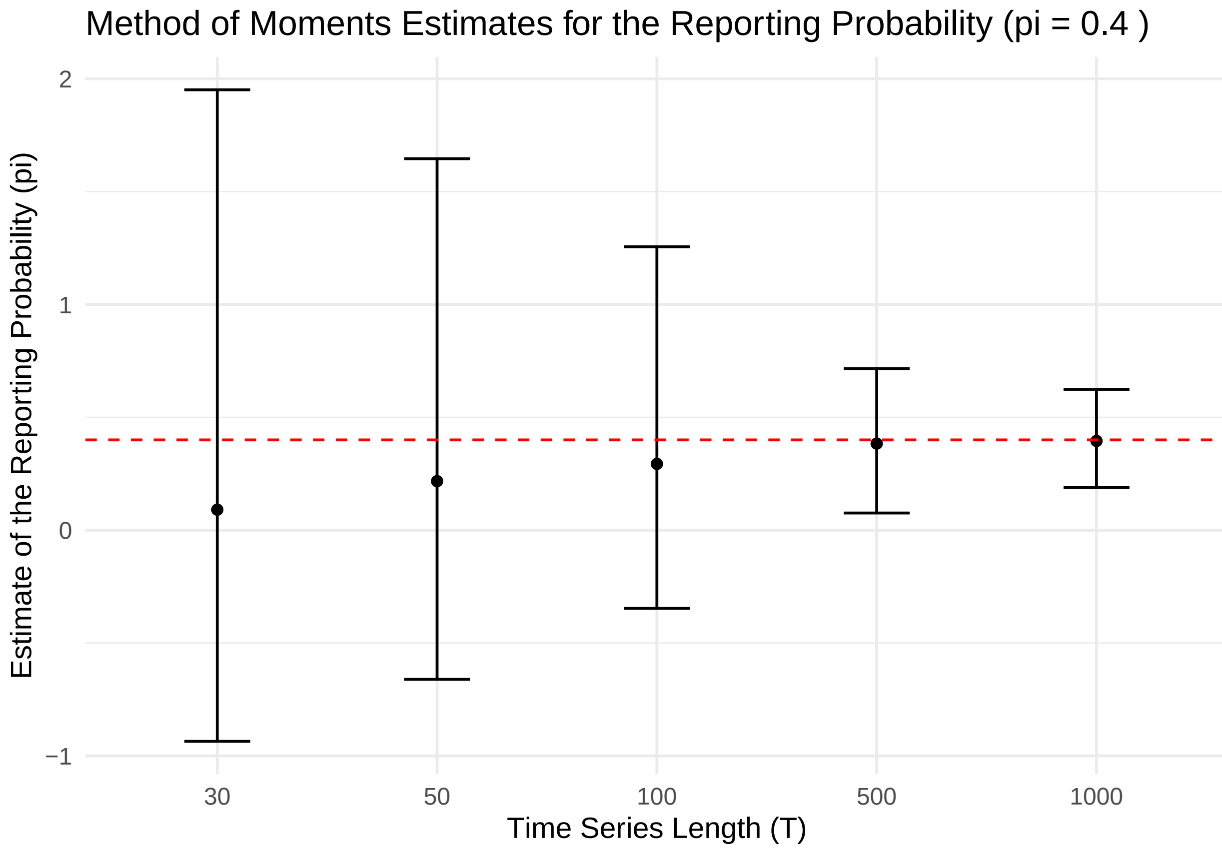
```
    labs(title = paste("Method of Moments Estimates for the Autoregressive Parameter (phi) for pi =", pi
         x = "Time Series Length (T)", y = "Estimate of the Autoregressive Parameter (phi)") +
    theme_minimal() +
    scale_x_discrete(limits = as.character(T_values))
}

# Plot for estimate of nu for each value of pi
nu_plots <- list()
for (i in 1:length(pi_seq)) {
  nu_plots[[i]] <- ggplot(as.data.frame(df_nu[,,i]), aes(x = T, y = as.numeric(Estimate))) +
    geom_point() +
    geom_errorbar(aes(ymin = as.numeric(Lower), ymax = as.numeric(Upper)), width = 0.3) +
    geom_hline(yintercept = nu, linetype = "dashed", color = "red") +
    labs(title = paste("Method of Moments Estimates for the Mean Parameter (nu) for pi =", pi_seq[i]),
         x = "Time Series Length (T)", y = "Estimate of the Mean Parameter (nu)") +
    theme_minimal() +
    scale_x_discrete(limits = as.character(T_values))
}

# Display the plots for each value of pi
for (i in 1:length(pi_plots)) {
  print(pi_plots[[i]])
}
```



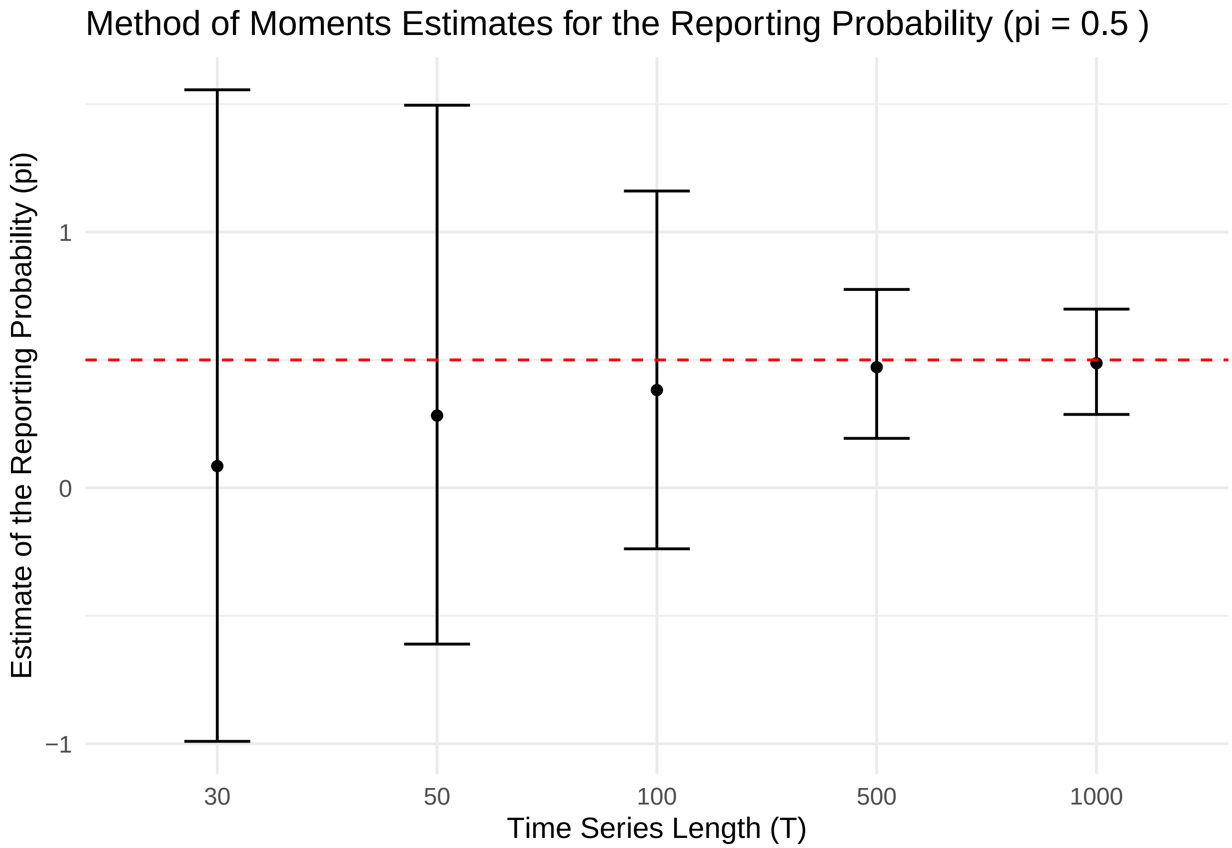


Method of Moments Estimates for the Reporting Probability (pi = 0.2)
3
2
1
0
−1
−2
Estimate of the Reporting Probability (pi)
30
50
100
500
1000
Time Series Length (T)

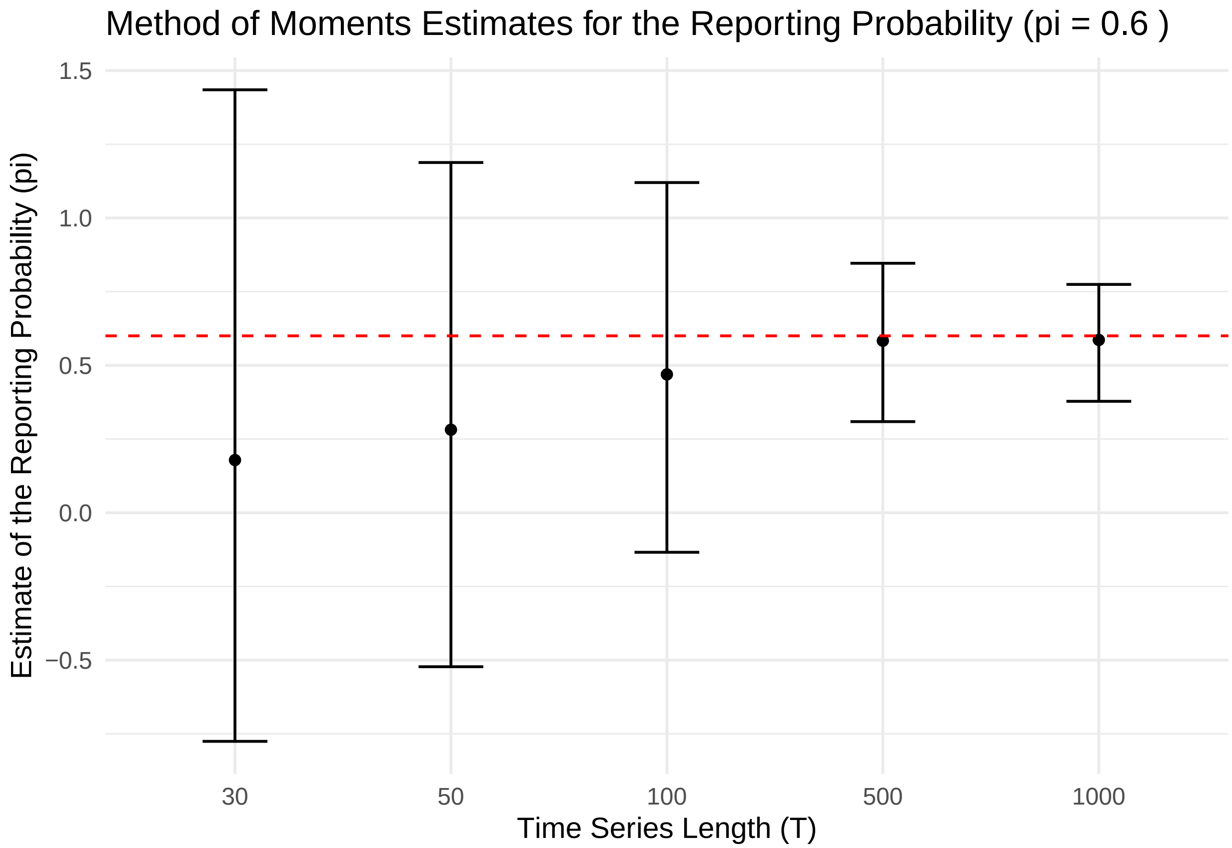

Method of Moments Estimates for the Reporting Probability (pi = 0.3)
Estimate of the Reporting Probability (pi)
2
1
0
−1
30
50
100
500
1000
Time Series Length (T)

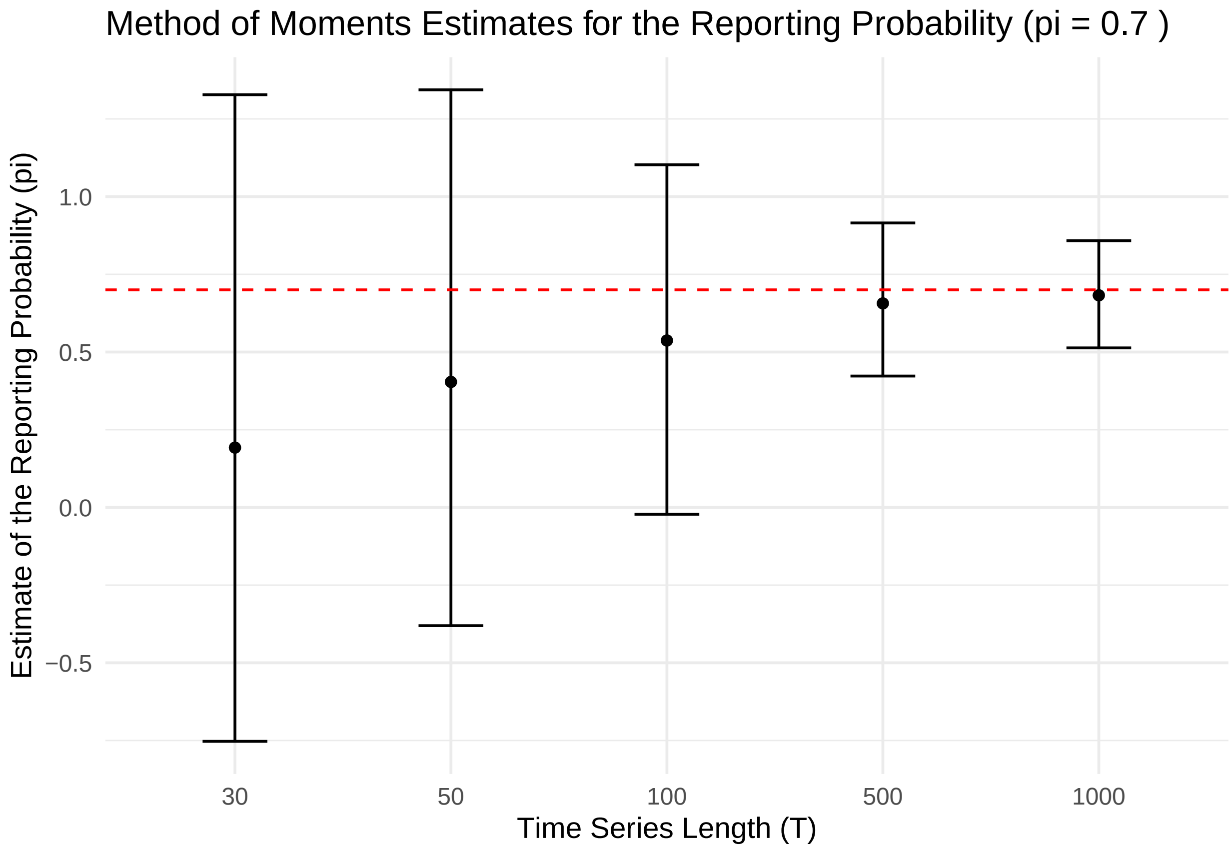


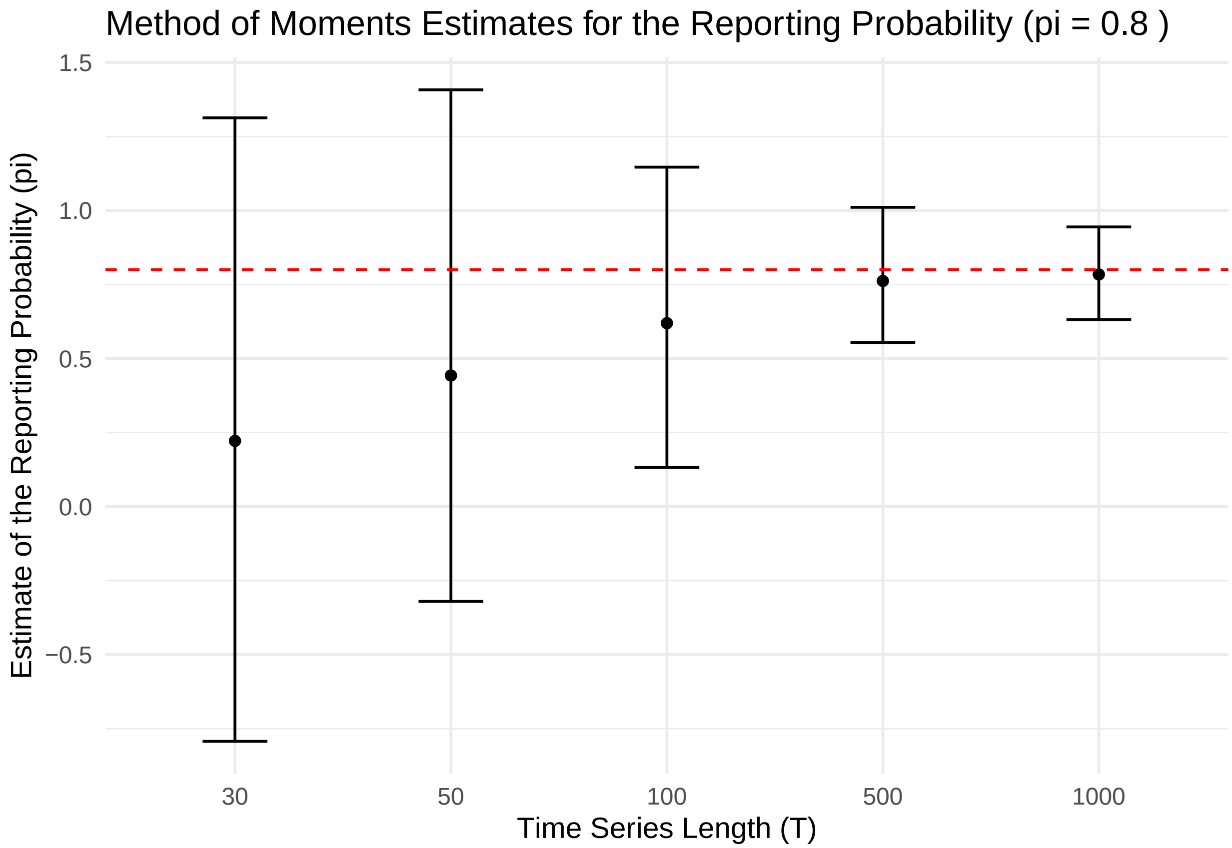
Method of Moments Estimates for the Reporting Probability (pi = 0.4)
Estimate of the Reporting Probability (pi)
2
1
0
−1
30
50
100
500
1000
Time Series Length (T)



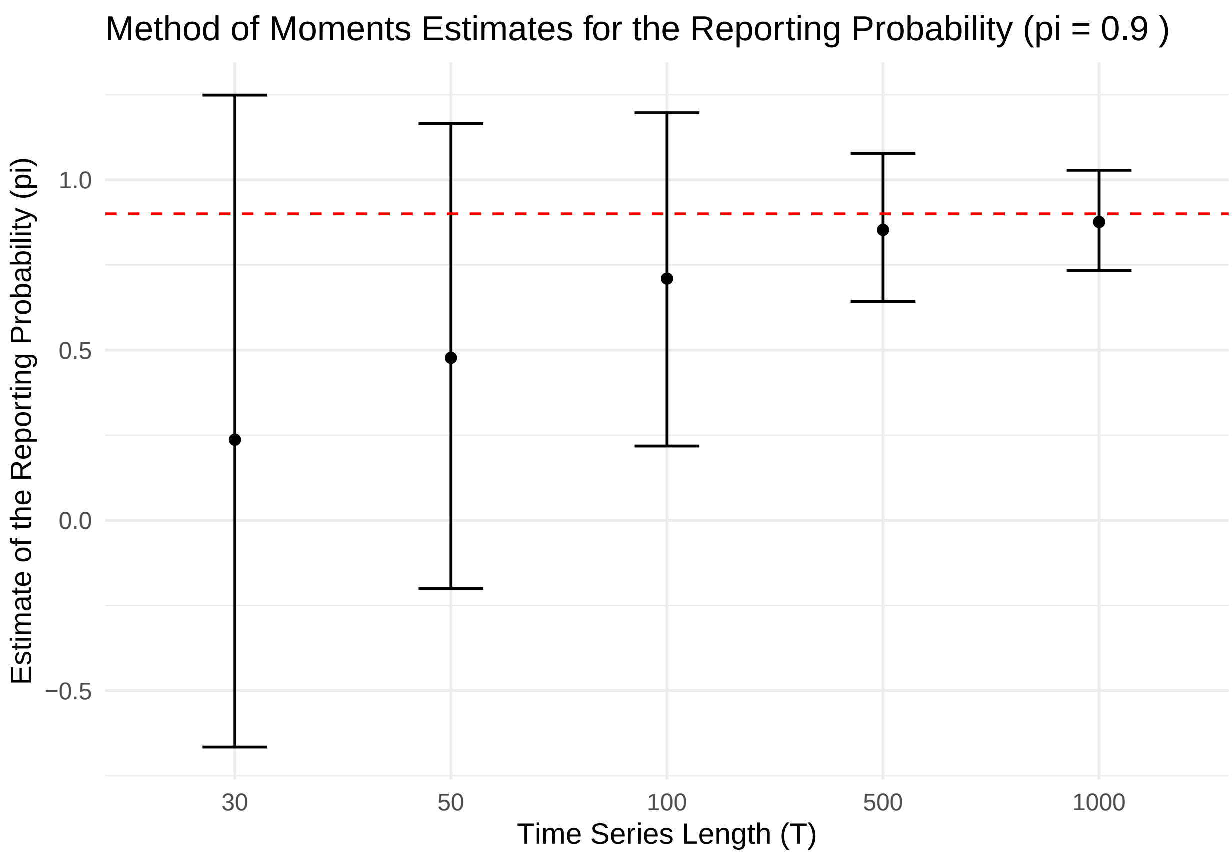
Method of Moments Estimates for the Reporting Probability (pi = 0.5)
Estimate of the Reporting Probability (pi)
1
0
−1
30
50
100
500
1000
Time Series Length (T)

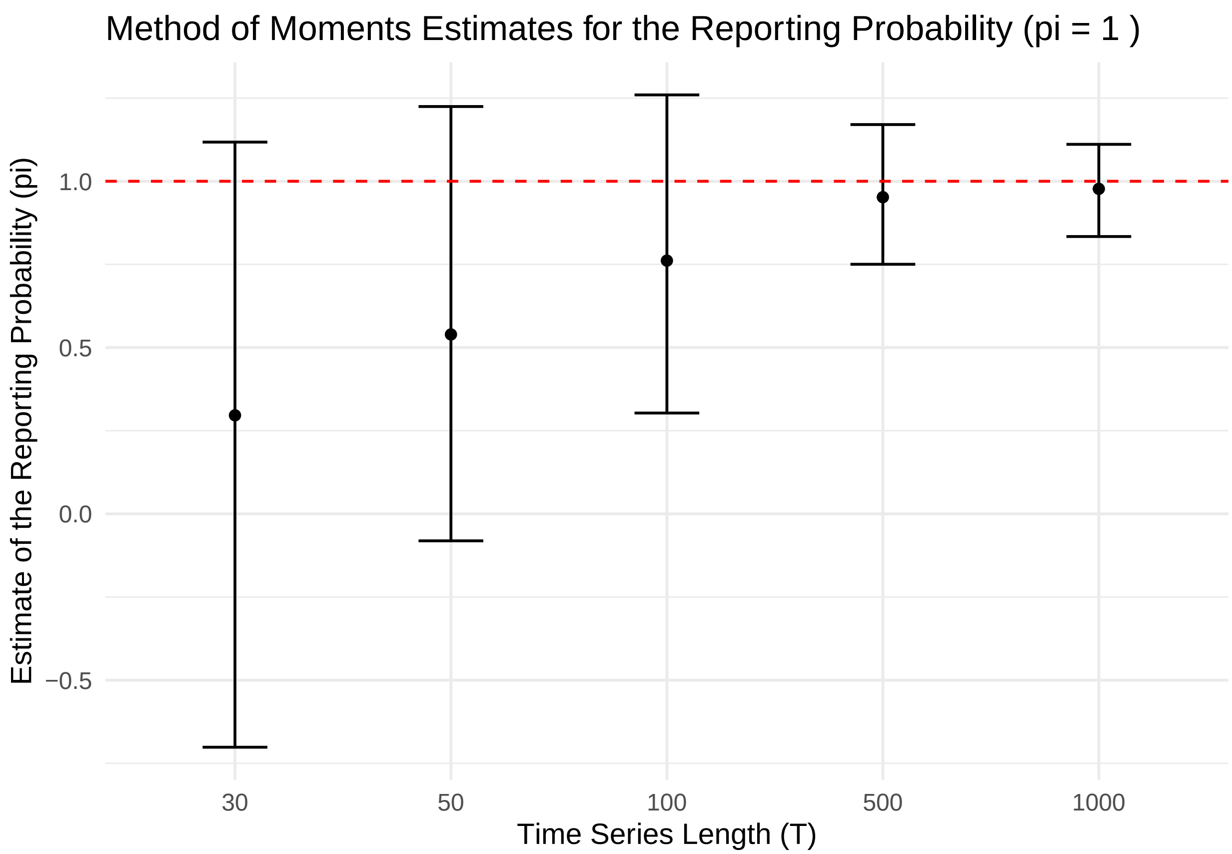

Method of Moments Estimates for the Reporting Probability (pi = 0.6)
Estimate of the Reporting Probability (pi)
1.5
1.0
0.5
0.0
−0.5
30
50
100
500
1000
Time Series Length (T)


Method of Moments Estimates for the Reporting Probability (pi = 0.7)
Estimate of the Reporting Probability (pi)
1.0
0.5
0.0
−0.5
30
50
100
500
1000
Time Series Length (T)

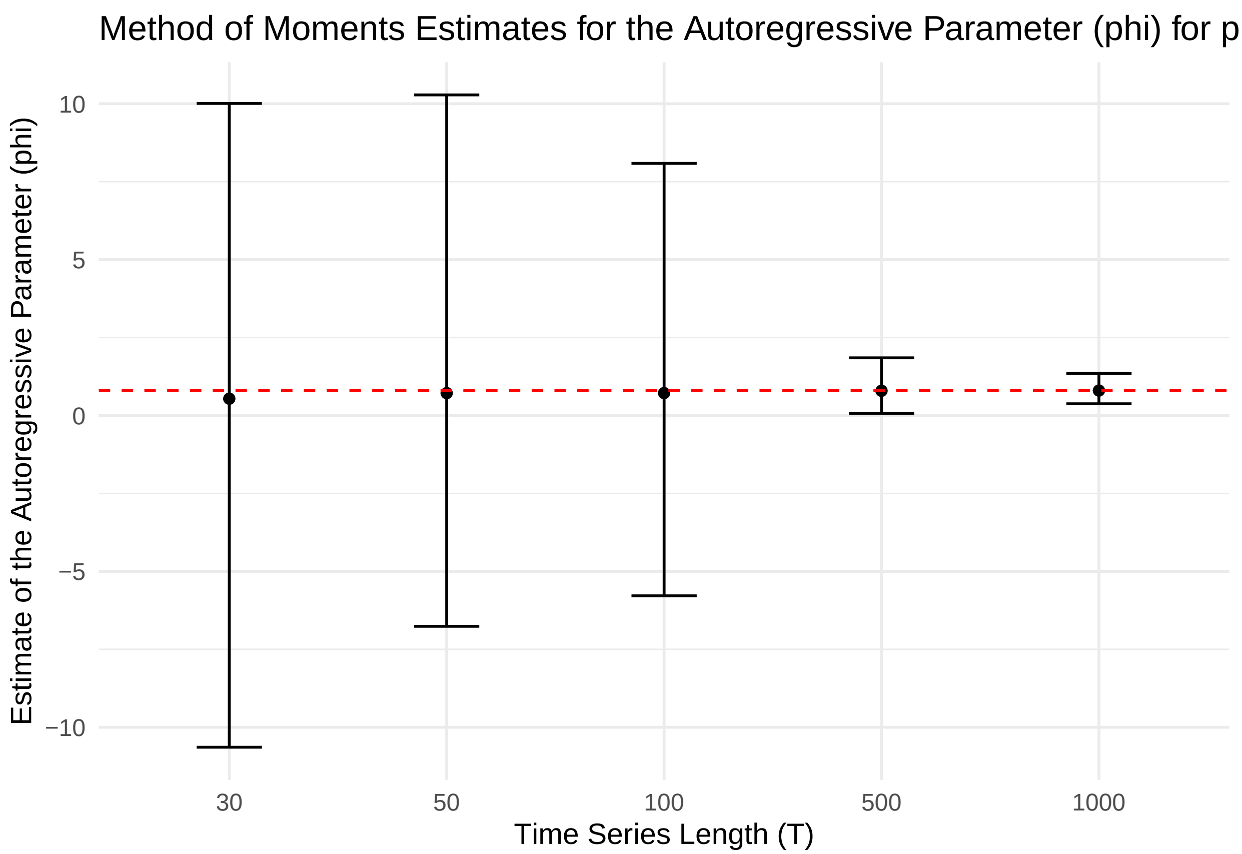


Method of Moments Estimates for the Reporting Probability (pi = 0.8)
Estimate of the Reporting Probability (pi)
1.5
1.0
0.5
0.0
−0.5
30
50
100
500
1000
Time Series Length (T)

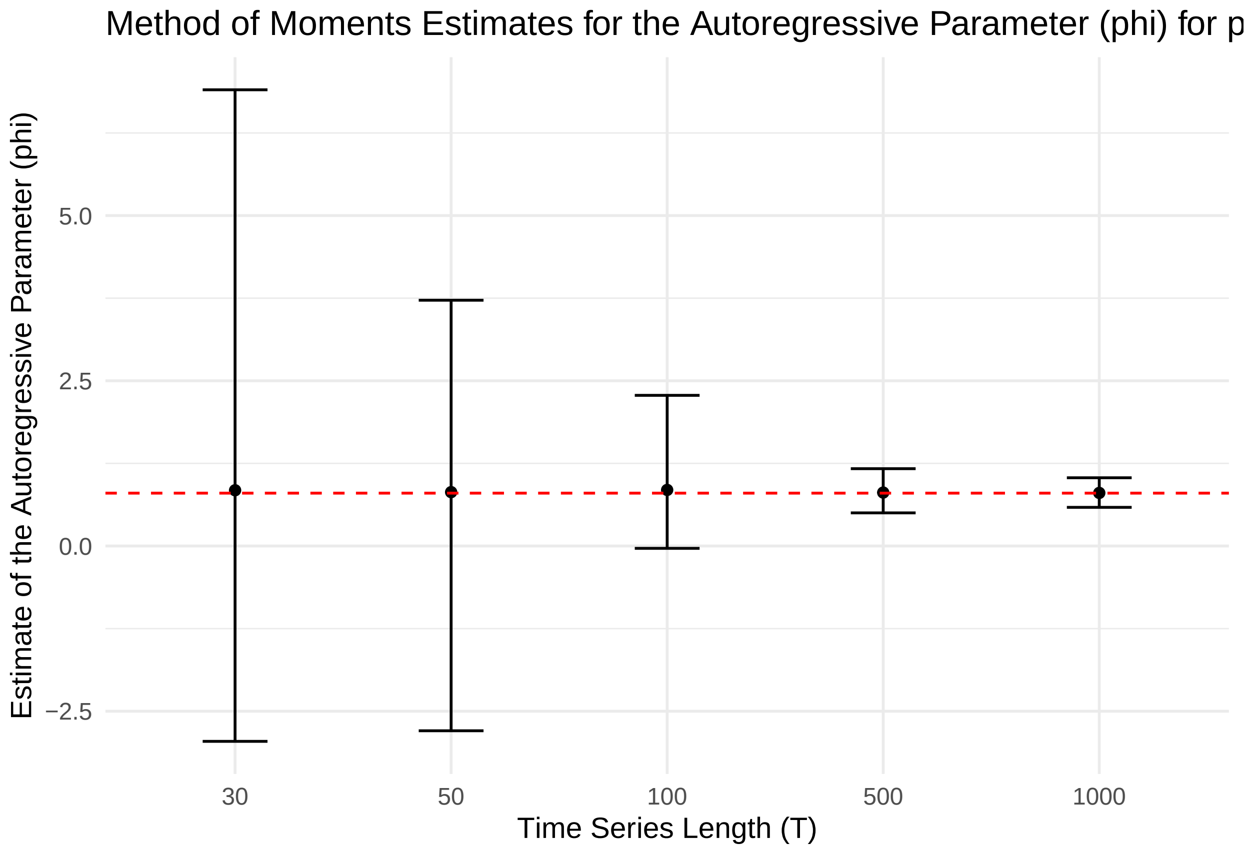

Method of Moments Estimates for the Reporting Probability (pi = 0.9)
Estimate of the Reporting Probability (pi)
1.0
0.5
0.0
−0.5
30
50
100
500
1000
Time Series Length (T)

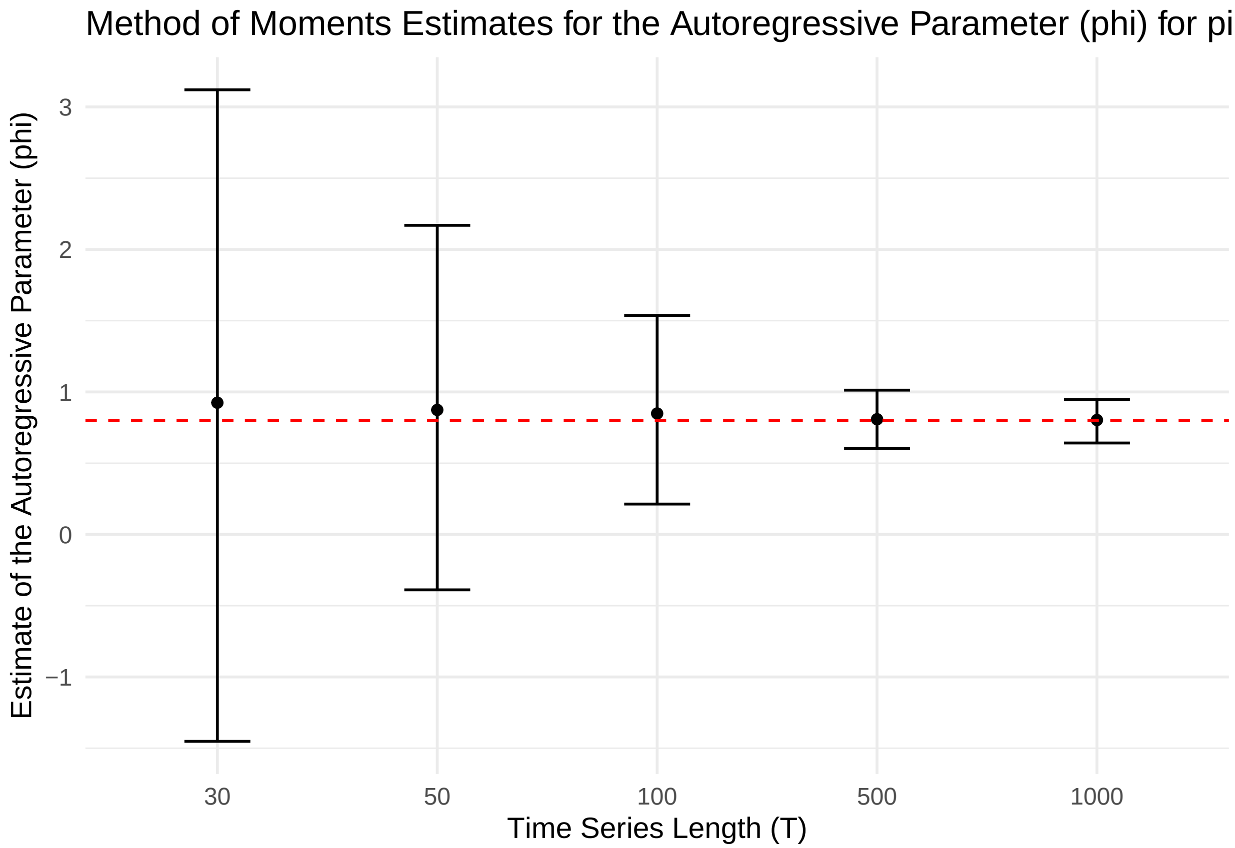


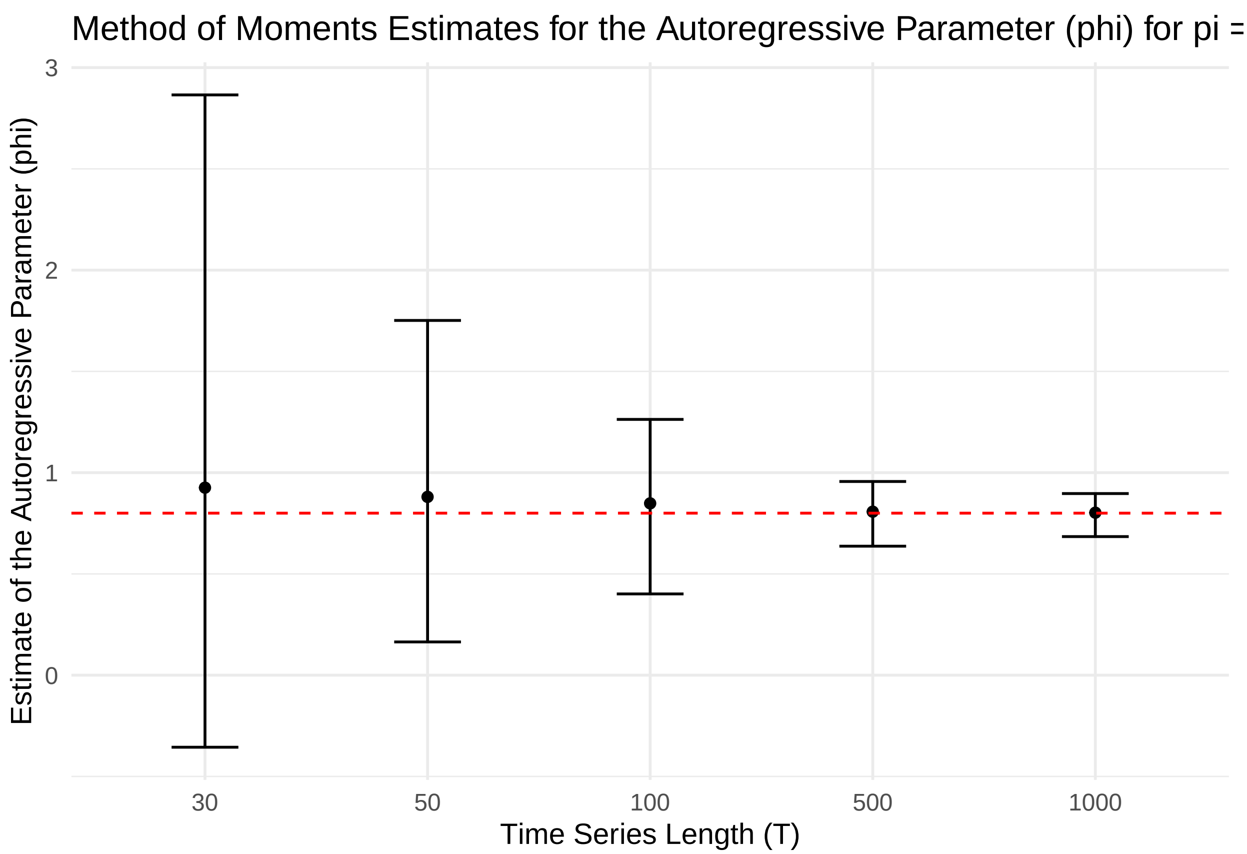

```
for (i in 1:length(phi_plots)) {
  print(phi_plots[[i]])
}
```



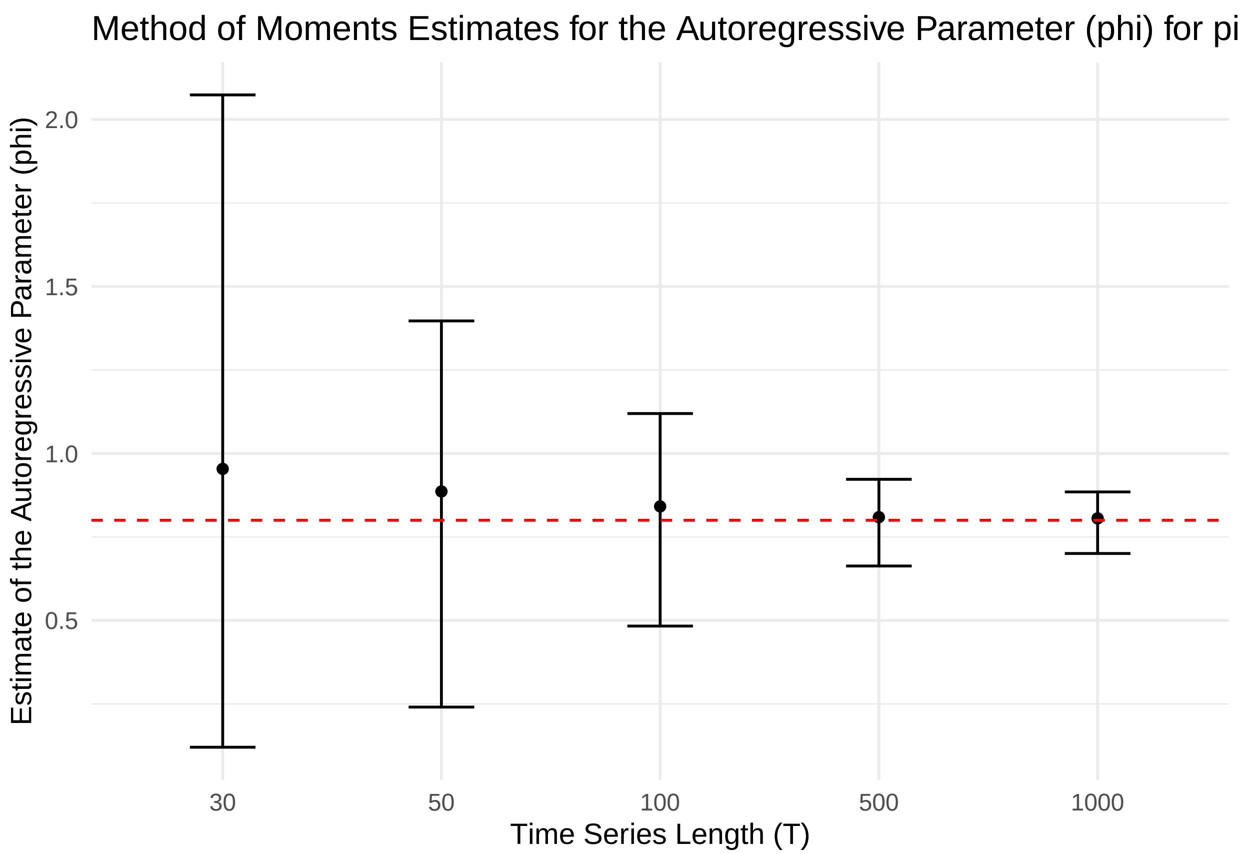
Method of Moments Estimates for the Autoregressive Parameter (phi) for pi
Estimate of the Autoregressive Parameter (phi)
10
5
0
−5
−10
30
50
100
500
1000
Time Series Length (T)

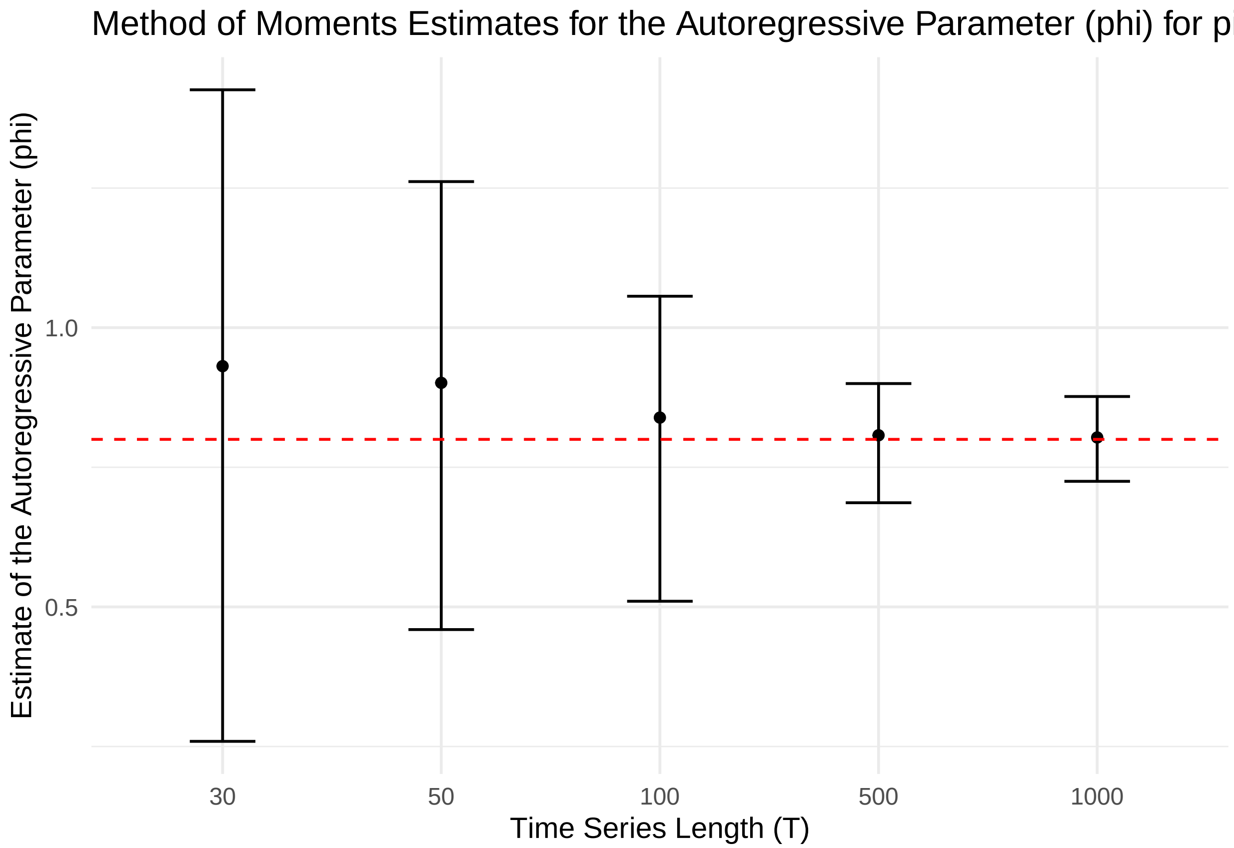

Method of Moments Estimates for the Autoregressive Parameter (phi) for p
Estimate of the Autoregressive Parameter (phi)
5.0
2.5
0.0
−2.5
30
50
100
500
1000
Time Series Length (T)

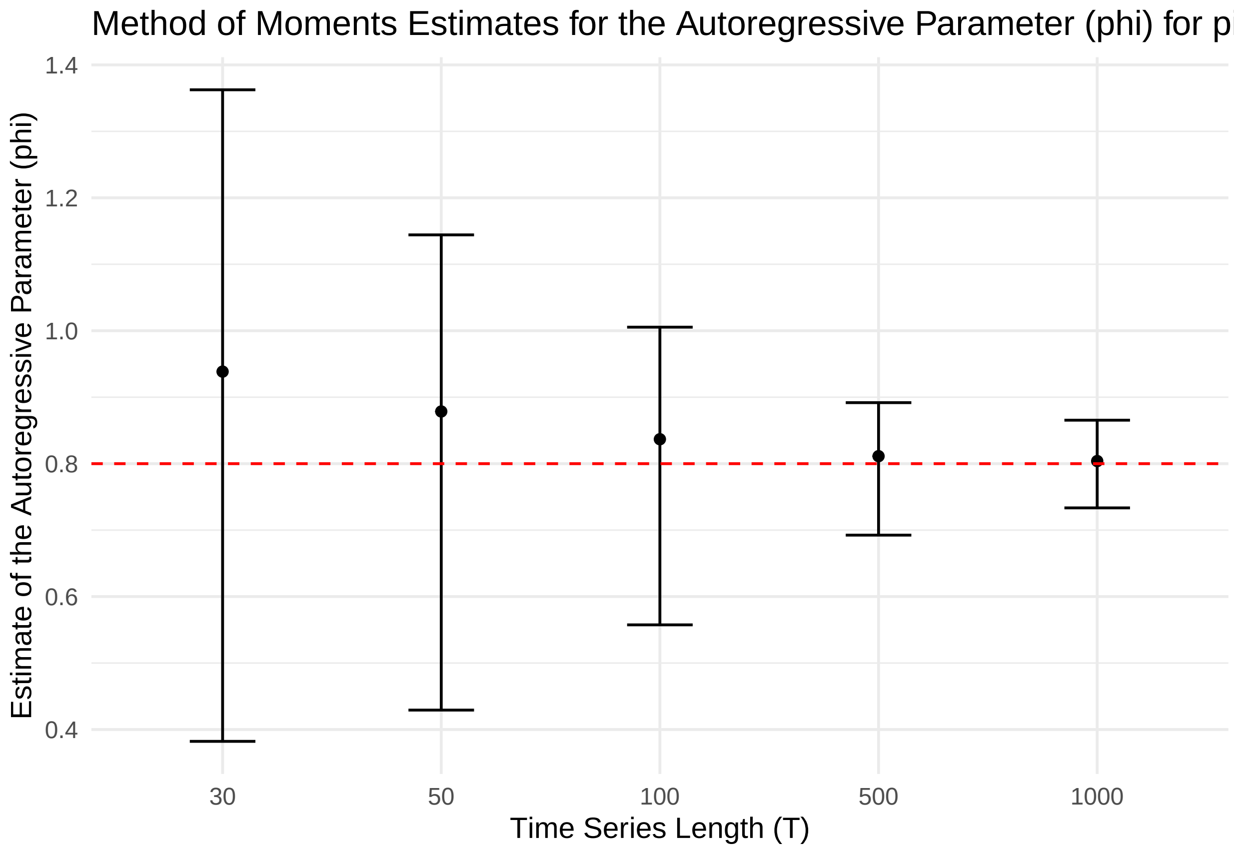

Method of Moments Estimates for the Autoregressive Parameter (phi) for pi
Estimate of the Autoregressive Parameter (phi)
3
2
1
0
−1
30
50
100
500
1000
Time Series Length (T)

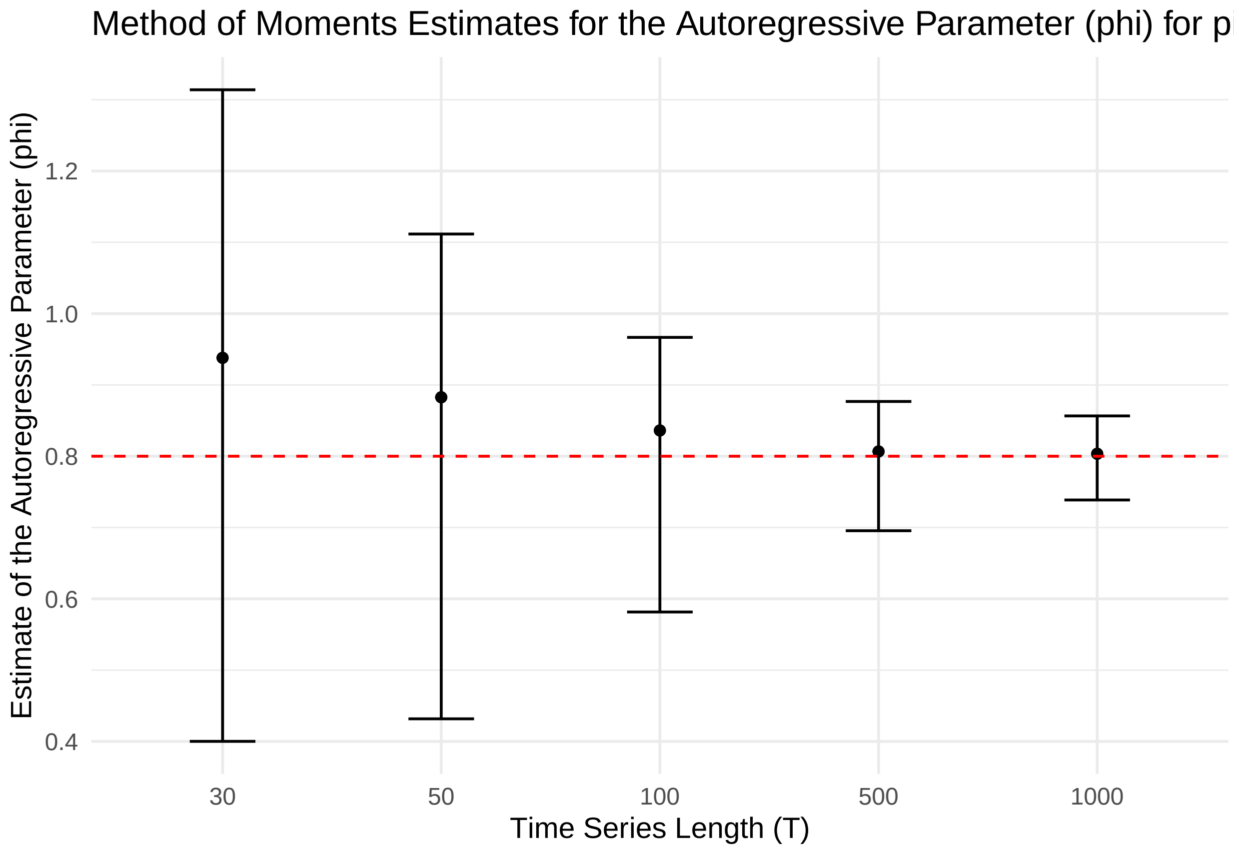


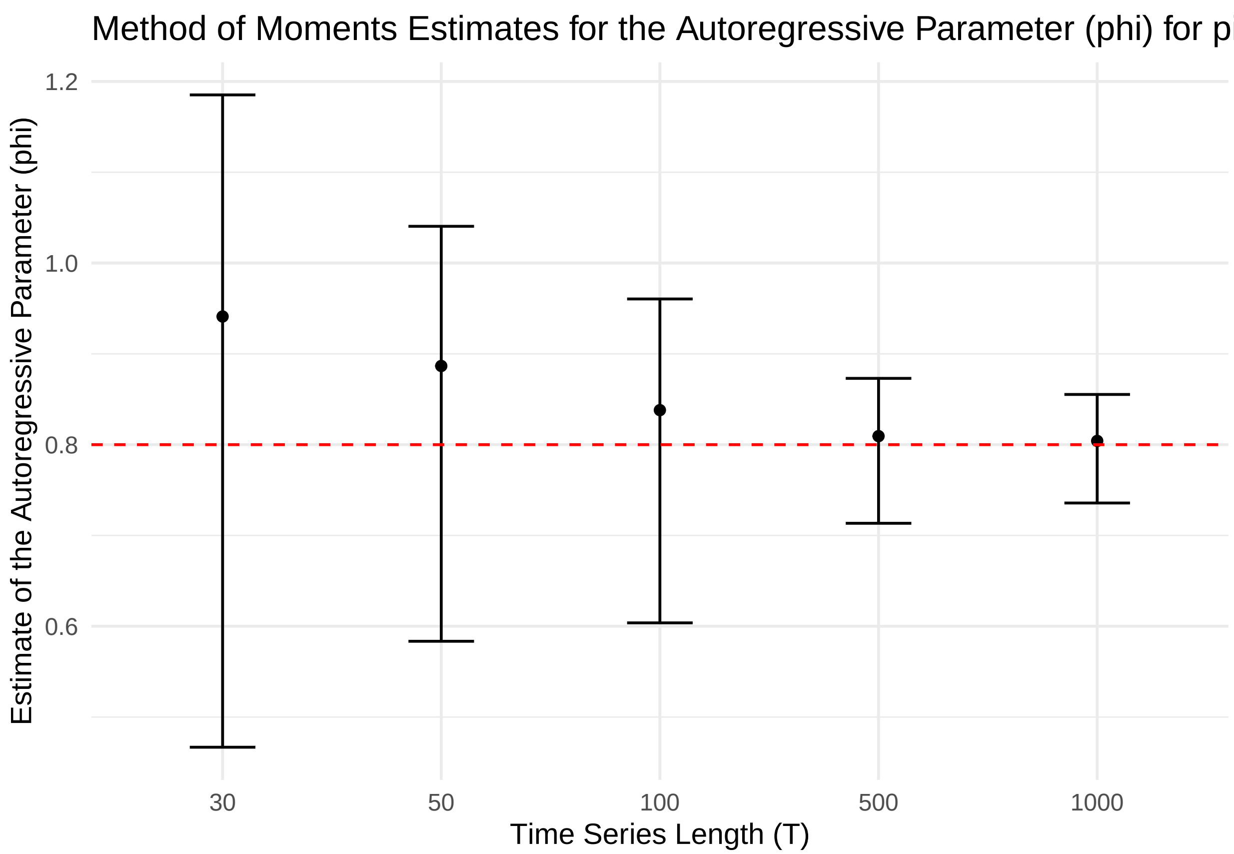
Method of Moments Estimates for the Autoregressive Parameter (phi) for pi =
3
2
1
0
Estimate of the Autoregressive Parameter (phi)
30
50
100
500
1000
Time Series Length (T)

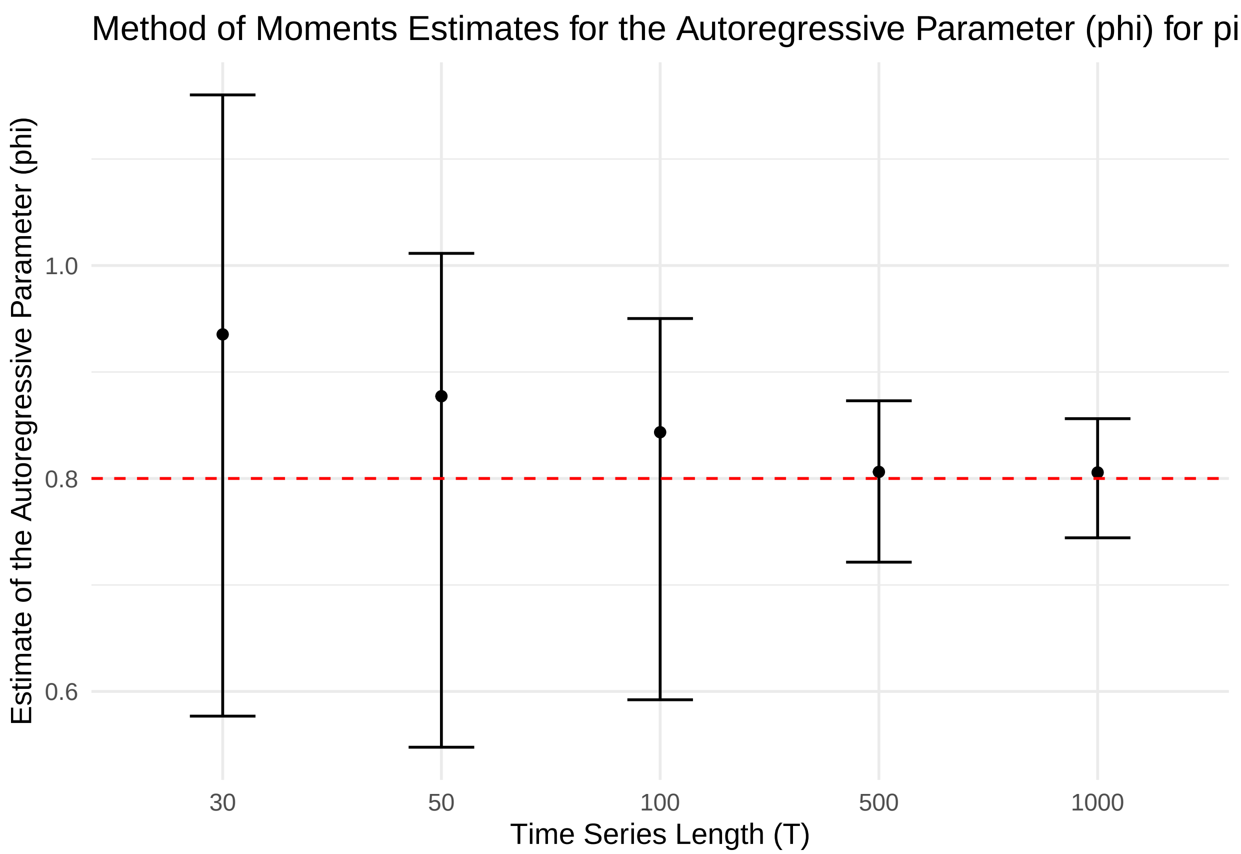

Method of Moments Estimates for the Autoregressive Parameter (phi) for pi
Estimate of the Autoregressive Parameter (phi)
2.0
1.5
1.0
0.5
30
50
100
500
1000
Time Series Length (T)


Method of Moments Estimates for the Autoregressive Parameter (phi) for pi
Estimate of the Autoregressive Parameter (phi)
1.0
0.5
30
50
100
500
1000
Time Series Length (T)

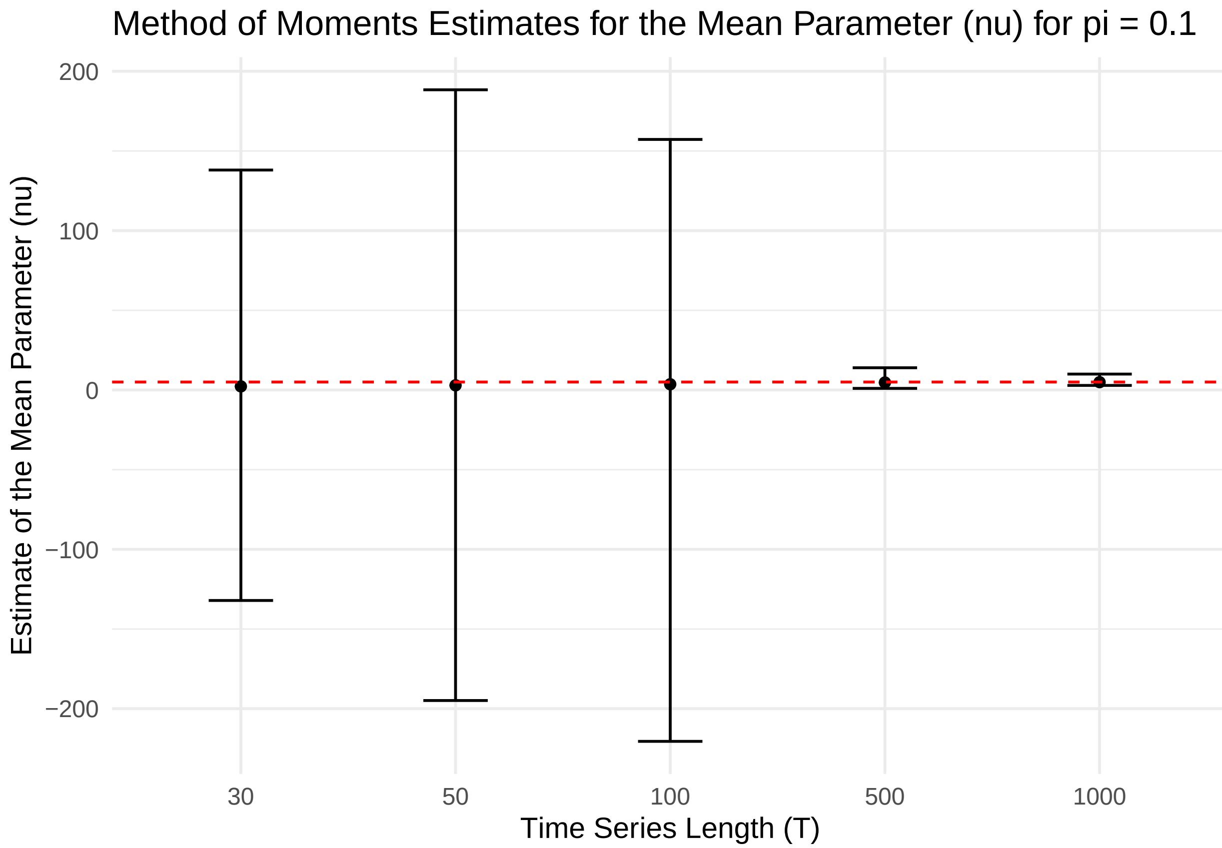

Method of Moments Estimates for the Autoregressive Parameter (phi) for pi
Estimate of the Autoregressive Parameter (phi)
1.4
1.2
1.0
0.8
0.6
0.4
30
50
100
500
1000
Time Series Length (T)

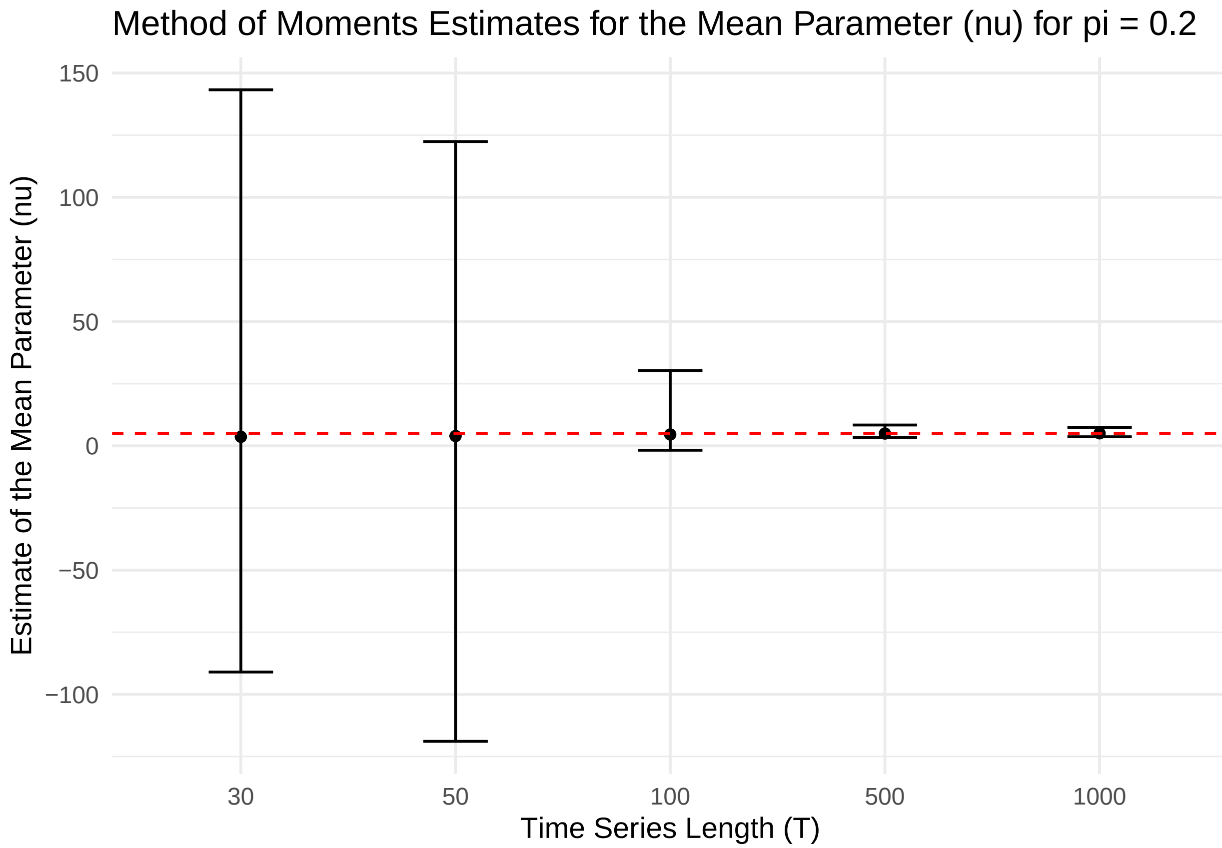

Method of Moments Estimates for the Autoregressive Parameter (phi) for pi
Estimate of the Autoregressive Parameter (phi)
1.2
1.0
0.8
0.6
0.4
30
50
100
500
1000
Time Series Length (T)

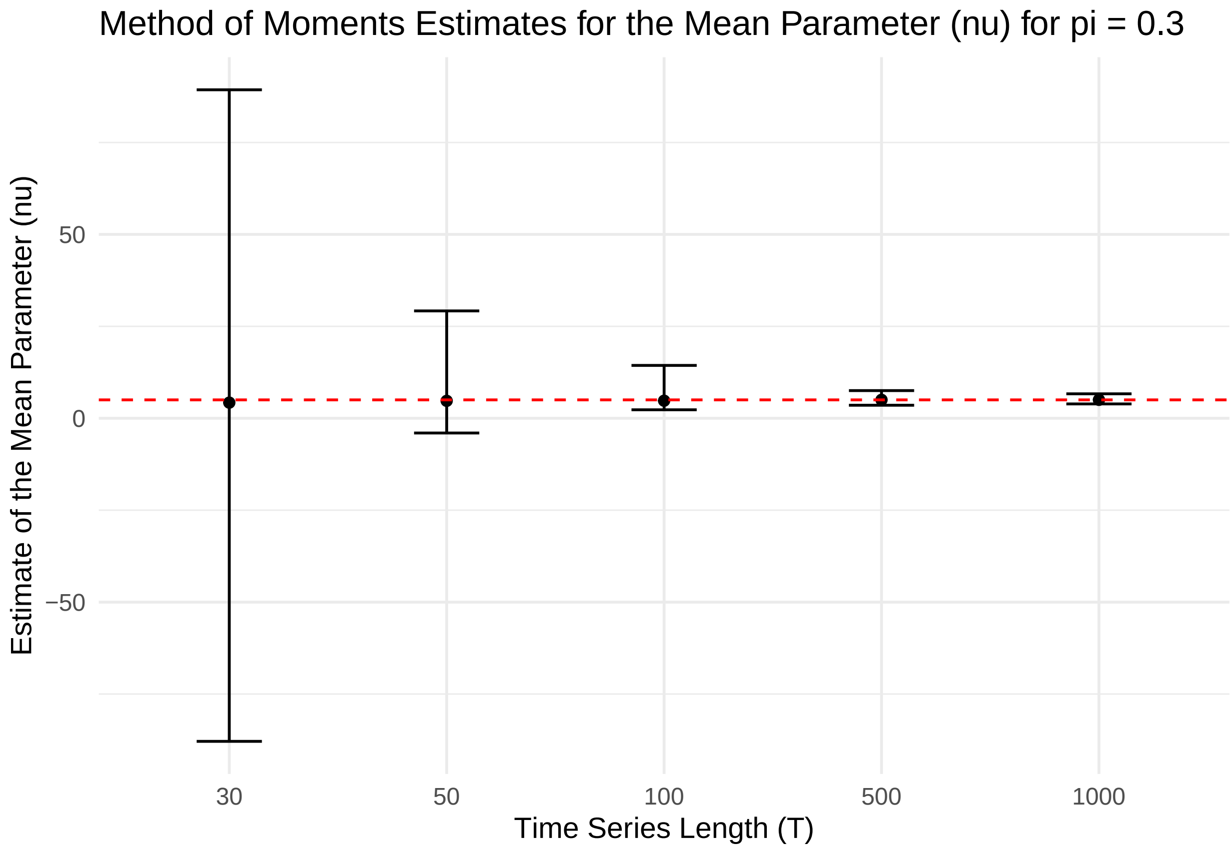

Method of Moments Estimates for the Autoregressive Parameter (phi) for pi
Estimate of the Autoregressive Parameter (phi)
1.2
1.0
0.8
0.6
30
50
100
500
1000
Time Series Length (T)

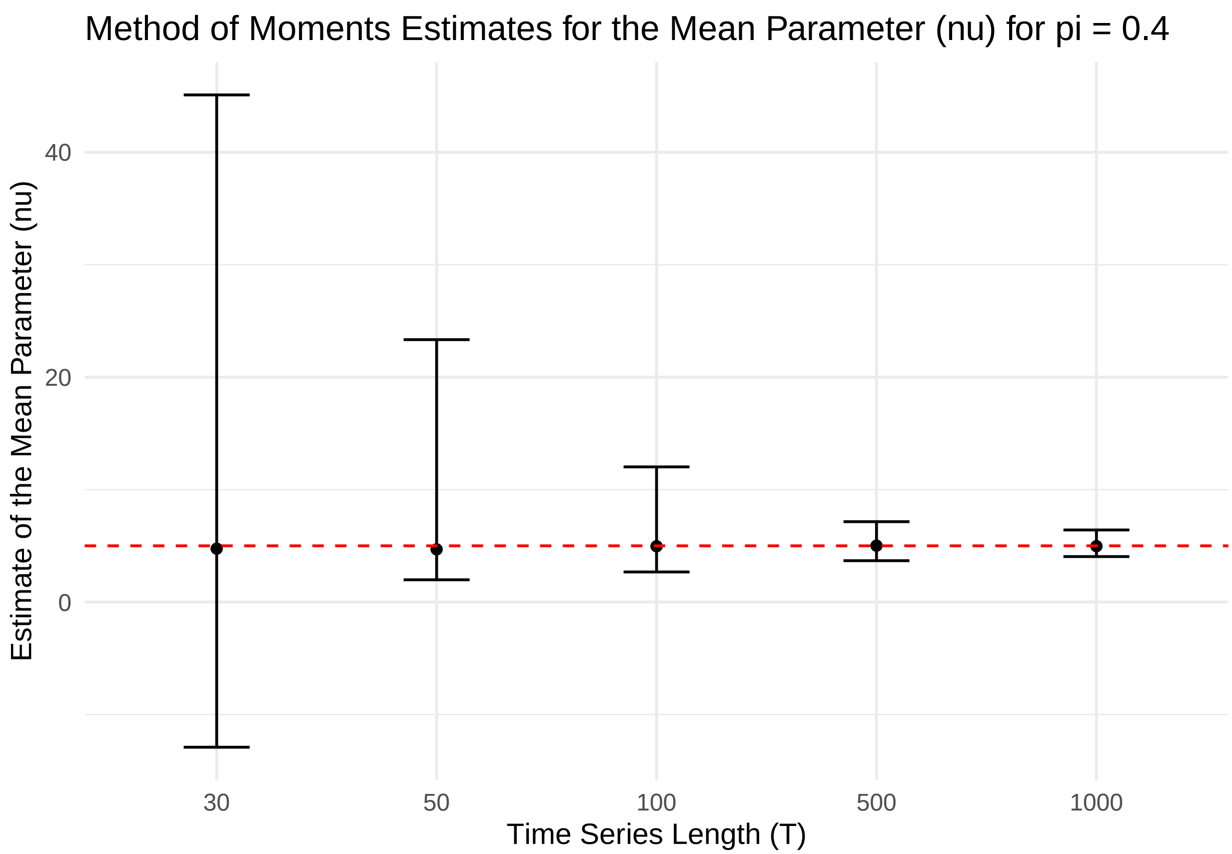


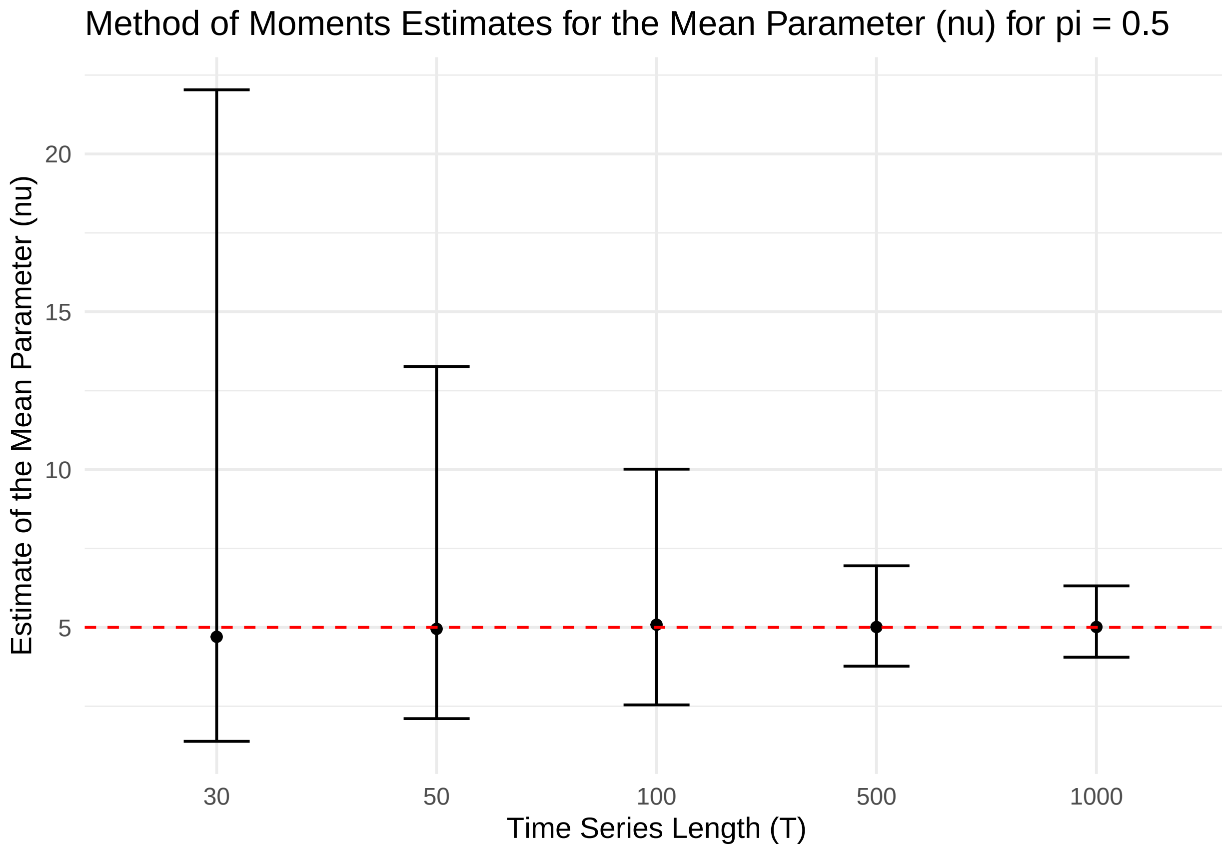

```
for (i in 1:length(nu_plots)) {
  print(nu_plots[[i]])
}
```

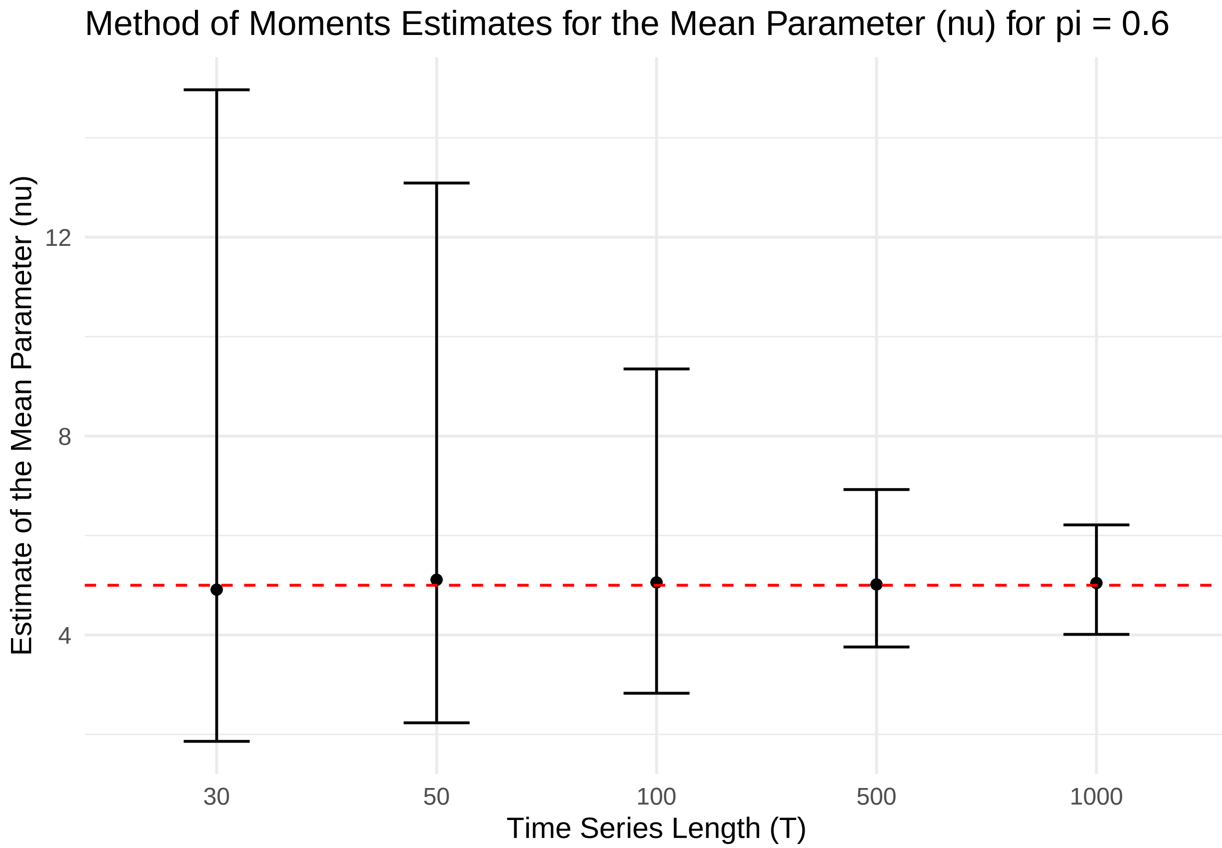

Method of Moments Estimates for the Mean Parameter (nu) for pi = 0.1
Estimate of the Mean Parameter (nu)
200
100
0
−100
−200
30
50
100
500
1000
Time Series Length (T)

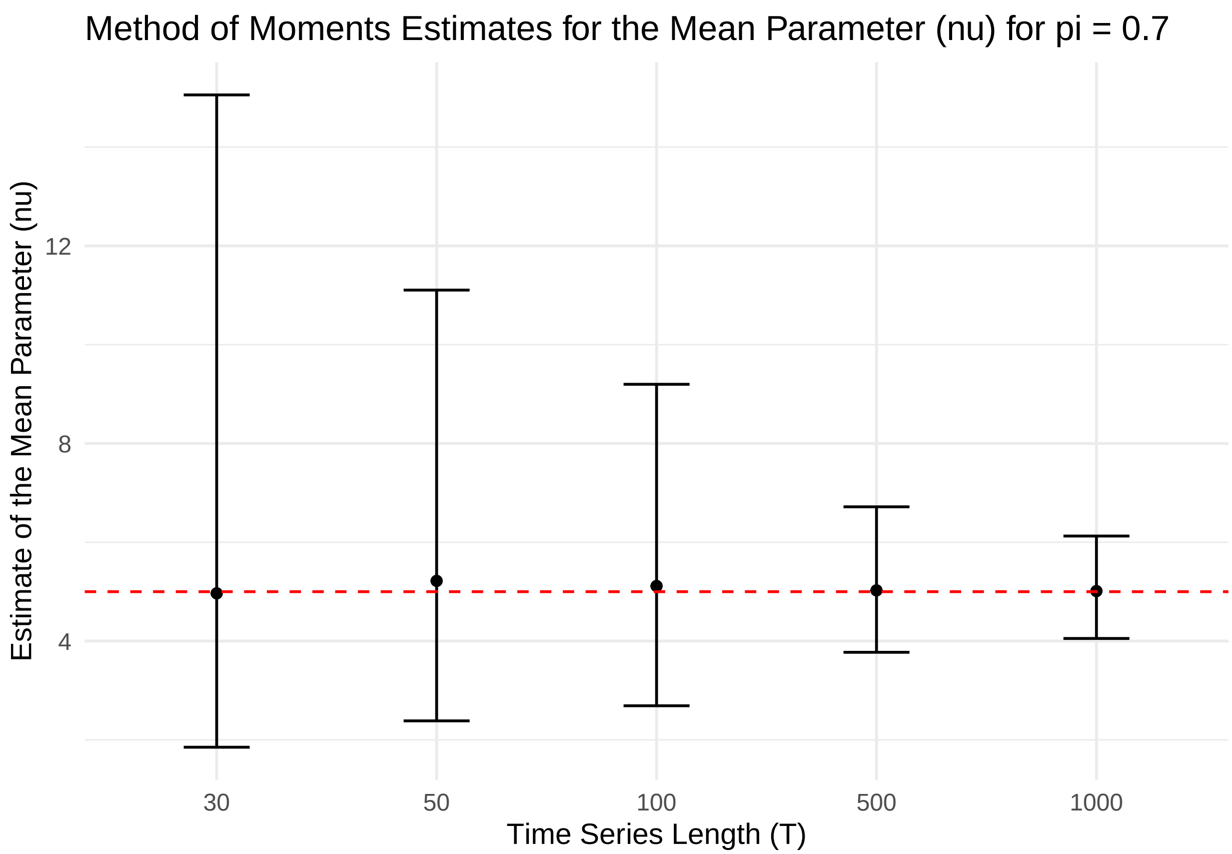

Method of Moments Estimates for the Mean Parameter (nu) for pi = 0.2
Estimate of the Mean Parameter (nu)
150
100
50
0
−50
−100
30
50
100
500
1000
Time Series Length (T)

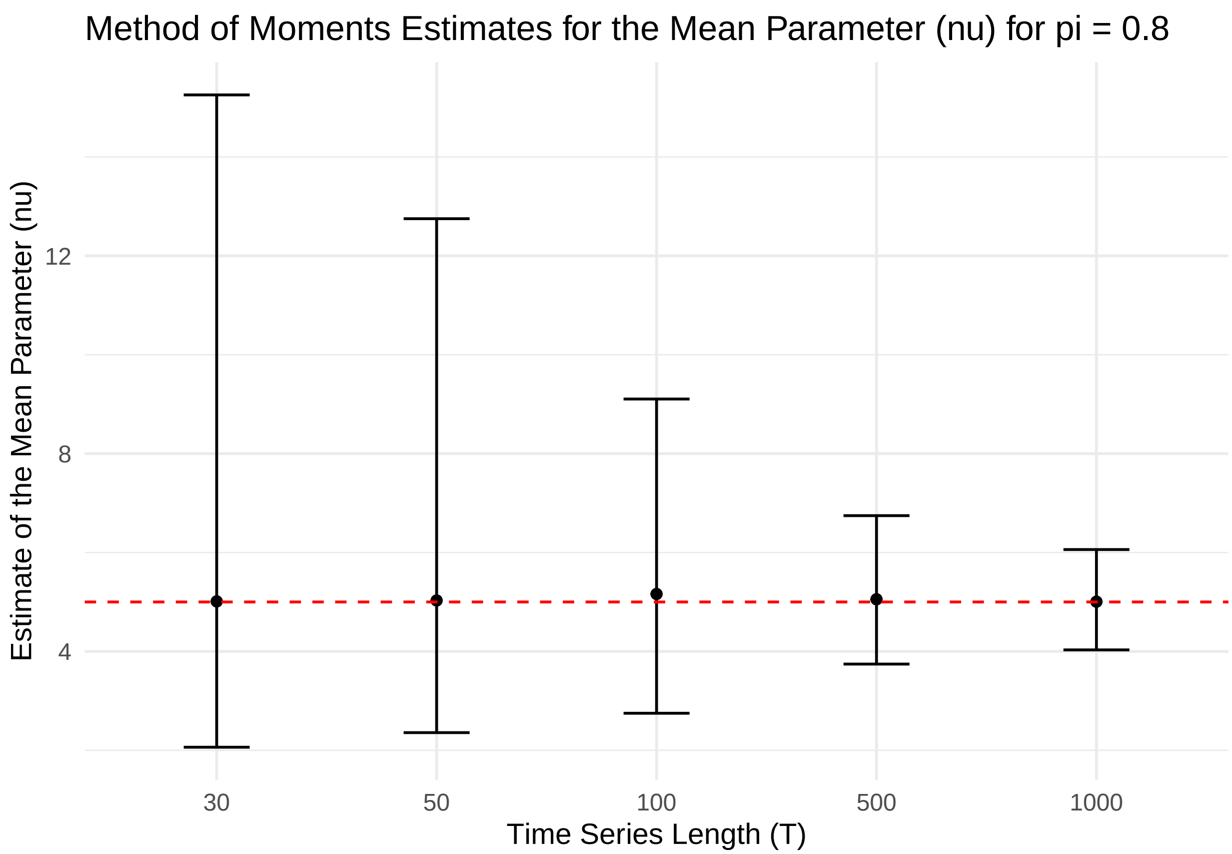

Method of Moments Estimates for the Mean Parameter (nu) for pi = 0.3
Estimate of the Mean Parameter (nu)
50
0
−50
30
50
100
500
1000
Time Series Length (T)

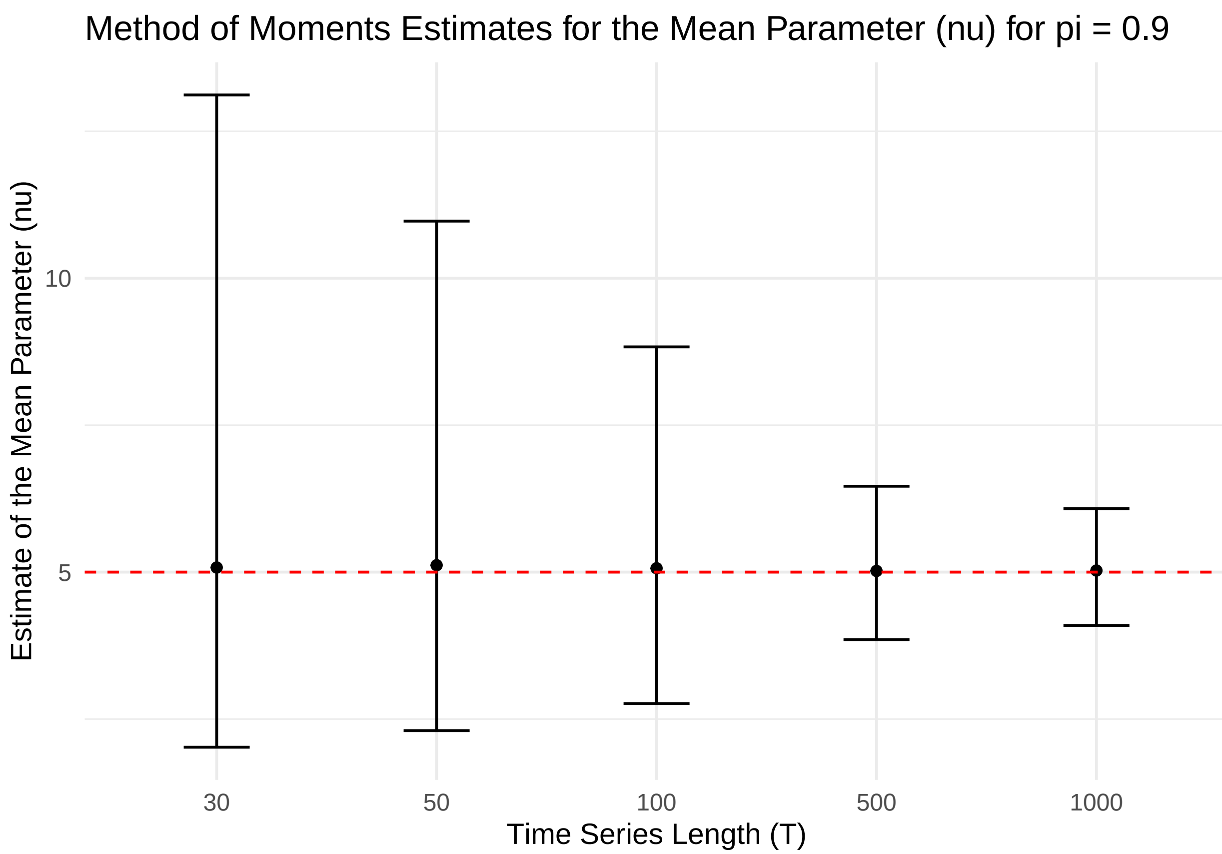

Method of Moments Estimates for the Mean Parameter (nu) for pi = 0.4
40
20
0
Estimate of the Mean Parameter (nu)
30
50
100
500
1000
Time Series Length (T)


Method of Moments Estimates for the Mean Parameter (nu) for pi = 0.5
Estimate of the Mean Parameter (nu)
20
15
10
5
30
50
100
500
1000
Time Series Length (T)

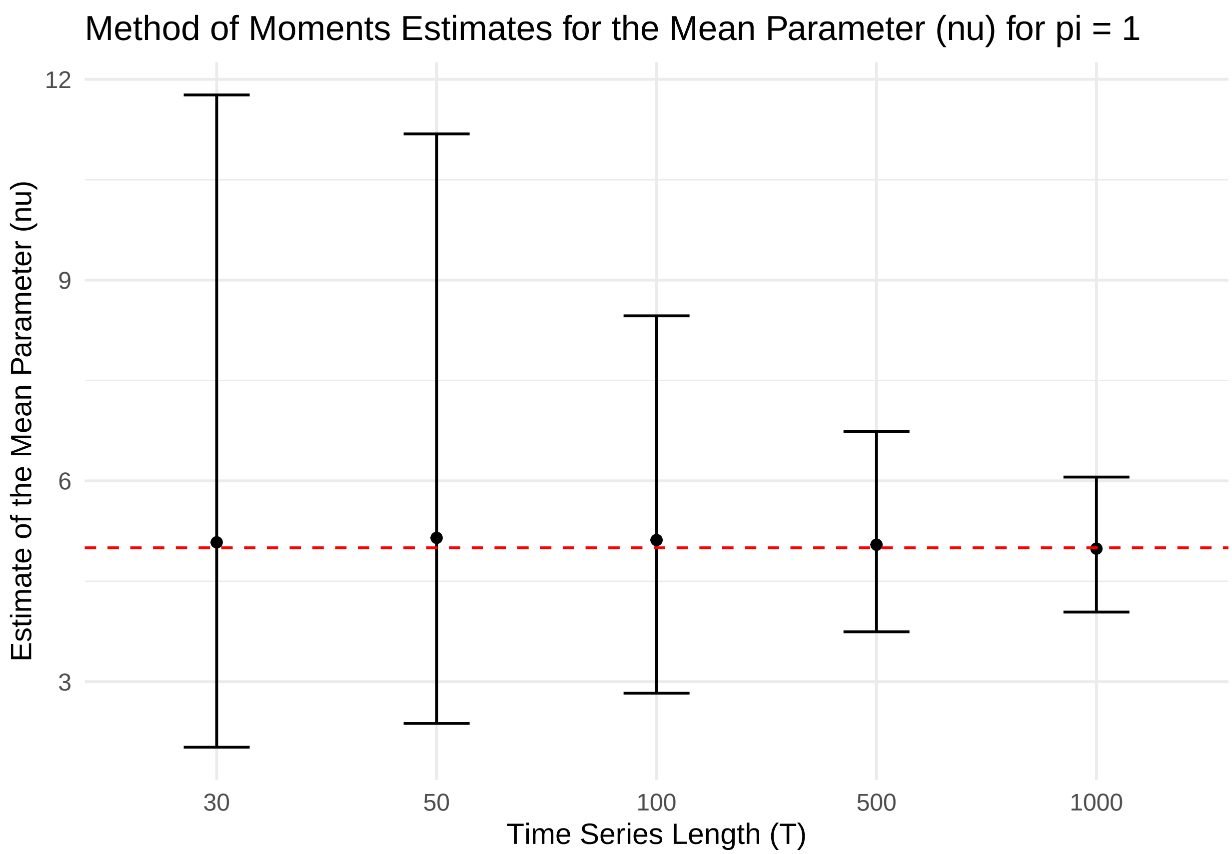

Method of Moments Estimates for the Mean Parameter (nu) for pi = 0.6
Estimate of the Mean Parameter (nu)
12
8
4
30
50
100
500
1000
Time Series Length (T)


Method of Moments Estimates for the Mean Parameter (nu) for pi = 0.7
Estimate of the Mean Parameter (nu)
12
8
4
30
50
100
500
1000
Time Series Length (T)


Method of Moments Estimates for the Mean Parameter (nu) for pi = 0.8
Estimate of the Mean Parameter (nu)
12
8
4
30
50
100
500
1000
Time Series Length (T)



Method of Moments Estimates for the Mean Parameter (nu) for pi = 0.9
Estimate of the Mean Parameter (nu)
10
5
30
50
100
500
1000
Time Series Length (T)




```r
# Load the "surveillance" package
library(surveillance)
```

```
## Loading required package: sp
```

```
## Loading required package: xtable
```

```
## This is surveillance 1.24.1; see 'package?surveillance' or
## https://surveillance.R-Forge.R-project.org/ for an overview.
```

```r
# Set parameter values
phi <- 0.8   # Autoregressive parameter
nu <- 5   # Mean parameter
pi <- 0.1
T <- 1000

# Function to simulate and estimate
simulate_and_estimate <- function(pi, phi, nu, T) {
  z <- rep(0, T)   # Initialize time series
  z[1] <- 5   # Initialize Z_1 at 5

  # Simulate time series
  for (i in 2:T) {
    z[i] <- rpois(1, lambda = nu + phi * z[i-1])
```

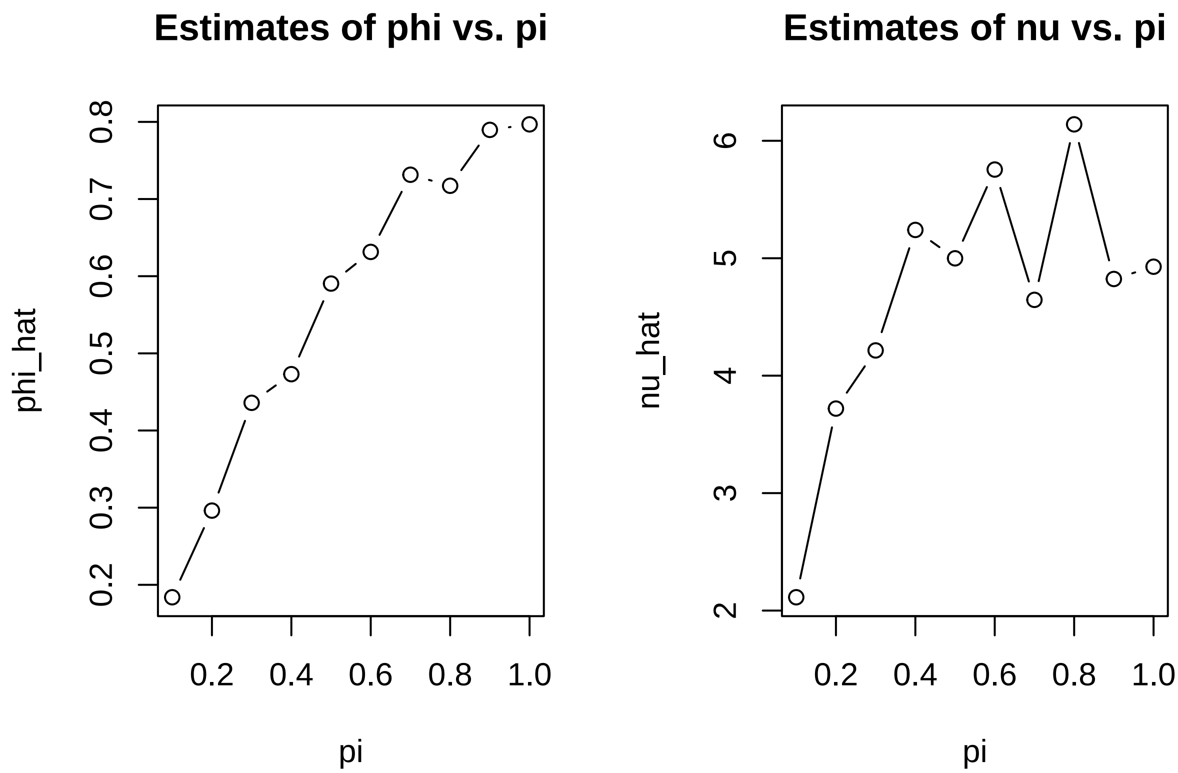
```r
  }

  # Thin the time series using binomial thinning
  y <- rbinom(T, z, pi)

  # Check for missing or NaN values
  if (any(is.na(y)) || any(is.nan(y))) {
    return(c(NA, NA))
  }

  # Convert y to "sts" object
  sts_object <- sts(y)

  # Specify the "endemic" and "epidemic" components using control parameter
  hhh_result <- hhh4(sts_object, control = list(end = list(f = ~ 1), ar = list(f = ~1),
                     family = "Poisson"))

  # Get parameter estimates and standard errors using summary()
  hhh_summary <- summary(hhh_result)
  phi_hat <- exp(hhh_summary$fixef["ar.1", "Estimate"])
  nu_hat <- exp(hhh_summary$fixef["end.1", "Estimate"])

  # Get standard errors
  phi_se <- hhh_summary$fixef["ar.1", "Std. Error"]
  nu_se <- hhh_summary$fixef["end.1", "Std. Error"]

  return(list(phi_hat = phi_hat, nu_hat = nu_hat, phi_se = phi_se, nu_se = nu_se))
}

# We will run the above function for different values of pi, and plot the estimates of nu vs pi, and ph
# Knit the Markdown file

estimates <- simulate_and_estimate(pi = pi, phi = phi, nu = nu, T = 1000)


phi_estimates = c()
nu_estimates = c()

phi_estimates[1] = estimates$phi_hat
nu_estimates[1] = estimates$nu_hat

#exp(phi_hat)
#exp(nu_hat)

# Print the estimates
cat("Estimates for pi =", pi, "=> phi_hat =", estimates$phi_hat, " nu_hat =", estimates$nu_hat, "\n")
```

```
## Estimates for pi = 0.1 => phi_hat = 0.1839054  nu_hat = 2.113534
```

```r
cat("Standard Errors => phi_se =", estimates$phi_se, " nu_se =", estimates$nu_se, "\n")
```

```
## Standard Errors => phi_se = 0.1625545  nu_se = 0.04132217
```

```
## Estimates for pi = 0.2 => phi_hat = 0.2961915  nu_hat = 3.719902
## Standard Errors => phi_se = 0.1028398  nu_se = 0.04498947
## Estimates for pi = 0.3 => phi_hat = 0.4358681  nu_hat = 4.215075
## Standard Errors => phi_se = 0.05919022  nu_se = 0.04628825
## Estimates for pi = 0.4 => phi_hat = 0.4730341  nu_hat = 5.240797
## Standard Errors => phi_se = 0.05390644  nu_se = 0.04852893
## Estimates for pi = 0.5 => phi_hat = 0.5904351  nu_hat = 4.999223
## Standard Errors => phi_se = 0.03838206  nu_se = 0.05443087
## Estimates for pi = 0.6 => phi_hat = 0.6314585  nu_hat = 5.755158
## Standard Errors => phi_se = 0.03383671  nu_se = 0.05680721
## Estimates for pi = 0.7 => phi_hat = 0.7314786  nu_hat = 4.645519
## Standard Errors => phi_se = 0.02790858  nu_se = 0.07342935
## Estimates for pi = 0.8 => phi_hat = 0.7171818  nu_hat = 6.139962
## Standard Errors => phi_se = 0.03020383  nu_se = 0.07486584
## Estimates for pi = 0.9 => phi_hat = 0.7896081  nu_hat = 4.823443
## Standard Errors => phi_se = 0.02489187  nu_se = 0.08992725
## Estimates for pi = 1 => phi_hat = 0.7967919  nu_hat = 4.927901
## Standard Errors => phi_se = 0.02381891  nu_se = 0.08975039
```



Estimates of phi vs. pi
phi_hat
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.2
0.4
0.6
0.8
1.0
pi
Estimates of nu vs. pi
nu_hat
2
3
4
5
6
0.2
0.4
0.6
0.8
1.0
pi