

Supplementary material I

Efficient Approximation of Interval Specific Estimates for Biological Growth Curve Models using Localized Likelihood Maximization and Its Applications to Real Data Analysis

Md Aktar Ul Karim¹, Amiya Ranjan Bhowmick^{1,*}

1. Null distribution

```
Initial data set up
K = 100
x0 = 10
r = 0.4
t = seq(from = 0, to = 19, by = 1)
q = length(t)
n= 100 # Trajectories

h = numeric(length = length(t)-1)
for(i in 1:(length(t)-1)){
  h[i] = t[i+1] - t[i]
}
time_int=length(t)-2

mu= K*(1 + ((x0/K)^(1/3) - 1)*exp(-r*t))^3 # Von Bertalanffy mean function
```

Change the mean function mu for different growth model.

```
plot(mu)
rep= 1000
# The ISRP estimators derived by Bhowmick et al.,(2014)'s method.
r_hat_vals = matrix(data = NA, nrow = rep, ncol = time_int)

K_hat_vals = matrix(data = NA, nrow = rep, ncol = time_int)

modified_RGR = matrix(data = NA, nrow = rep, ncol = time_int)

sigma2 = 0.1
rho = 0.1
library(mvtnorm) # To generate multivariate normal distribution data
x0_mean_vals = numeric(length = rep)
cov_mat = matrix(data = NA, nrow = length(t), ncol = length(t))

for (i in 1:length(t)) {
```

*Corresponding author

Email addresses: mdaktarulkarim829@gmail.com (Md Aktar Ul Karim), amiyaiitb@gmail.com (Amiya Ranjan Bhowmick)

¹Department of Mathematics, Institute of Chemical Technology, Mumbai

```

39   for (j in 1:length(t)) {
40     cov_mat[i,j] = sigma2*rho^(abs(i-j))
41   }
42 }
43
44 for (i in 1:rep) {
45   library(mvtnorm)
46   data = mvtnorm::rmvnorm(n, mean = mu, sigma = cov_mat)
47   data_mean = colMeans(data)
48   x0_mean_vals[i] = mean(data_mean[1])
49
50
51   for(j in 1:time_int){
52     d1= (data_mean[j+1])^(1/3) - (data_mean[j])^(1/3)
53     d2= (data_mean[j+2])^(1/3) - (data_mean[j+1])^(1/3)
54     d12=d1/d2
55     r_hat_vals[i,j]=(3/h[j])*log(d12)
56     K_hat_vals[i,j]= ((x0_mean_vals[i])^(1/3)-(d1/(exp(-r_hat_vals[i,j]*t[j]/3)*
57                       (exp(-r_hat_vals[i,j]*h[j]/3)-1))))^3
58     modified_RGR[i,j]= (r_hat_vals[i,j]*exp(-r_hat_vals[i,j]*(t[j]/3))*
59                       ((K_hat_vals[i,j])^(1/3) - (x0_mean_vals[i])^(1/3)))/
60     ((K_hat_vals[i,j])^(1/3) + exp(-r_hat_vals[i,j]*(t[j]/3))*
61     ((x0_mean_vals[i])^(1/3)-(K_hat_vals[i,j])^(1/3)))
62   }
63 }
64
65 V = matrix(data = NA, nrow = rep, ncol = time_int)
66
67 test_stat = matrix(data = NA, nrow = rep, ncol = time_int-2)
68
69 for (i in 1:rep) {
70   for (j in 1:time_int) {
71     V[i,j ] = log((1/modified_RGR[i,j]) + (1/ r_hat_vals[i,j]))
72   }
73
74   for (j in 1:(time_int-2)) {
75     test_stat[i,j] = V[i,j+2] - 2*V[i,j+1] + V[i,j]
76   }
77 }
78
79 par(mfrow = c(1,1))
80
81 #View(test_stat)
82
83 test_stat_final=na.omit(test_stat) # to delete the replications containing at-least one NaN entries.
84 View(test_stat_final)
85
86 # Histogram for the test statistics
87 # We have to change the time point to get different test statistics for different time points.
88 hist( test_stat_final[,11]/sd(test_stat_final[,11]), probability = TRUE, main ='',
89       col = "lightgreen", border = "black", xlab = expression(T[t]), breaks = 10, ylim=c(0,0.4))
90 curve(dnorm(x), add = TRUE, col = "red", lwd = 2, lty = 1)
91
92

```

```

93 # The ISRP estimators of the parameter derived by localized MLE method
94 n = 100      # number of independent trajectories
95 data = mvtnorm::rmvnorm(n, mean = mu, sigma = cov_mat)
96 matplot(t, t(data), type = "l")
97
98 # Likelihood function for computing overall estimates that is it will
99 # give a single estimate of the parameters using the data at all time
100 # points 1 to q.
101 fun_likelihoood = function(param){
102     r = param[1]
103     K = param[2]
104     sigma2 = param[3]
105     rho = param[4]
106     mu = K*((1 + ((x0/K)^(1/3) - 1)*exp(-r*t/3))^3)
107
108     cov_mat = matrix(data = NA, nrow = length(t), ncol = length(t))
109     for (i in 1:q) {
110         for (j in 1:q) {
111             cov_mat[i,j] = sigma2*rho^(abs(i-j))
112         }
113     }
114     exponent = 0
115     for(i in 1:n){
116         exponent = exponent + t(data[i,]-mu)%*(solve(cov_mat))%*(data[i,]-mu)
117     }
118
119
120     log_lik = - (n*q/2)*log(2*pi) - (n/2)*log(det(cov_mat)) - (1/2)*exponent
121     return(-log_lik)
122 }
123 param_0 = c(0.4, 100, 3, 0.1)
124 out = optim(param_0, fun_likelihoood, method = "L-BFGS-B",
125             hessian = TRUE, lower = c(0.001, 50, 0.0001, -0.001), upper = c(3, 150, 10, 0.999))
126 out$par
127 solve(out$hessian)

```

128 The Main programme starts here for ISRP estimates by maximizing localized likelihood. The following
 129 program will estimate the ISRP of r , K , σ^2 and ρ by maximizing localized likelihood function that will use
 130 only the data for three consecutive time points. Therefore, for $q = 20$ time points, we will obtain 18 estimates
 131 of r and K . Note that we did not provide the ISRP estimate of X_0 . We estimated X_0 by taking the average of
 132 all the observations at time point $t = 1$.

```

133 est_local = matrix(data = NA, nrow = q-2, ncol = 4)
134
135 fun_local_likelihoood = function(param){
136     r = param[1]
137     K = param[2]
138     sigma2 = param[3]
139     rho = param[4]
140     mu = K*((1 + ((x0/K)^(1/3) - 1)*exp(-r*t/3))^3)
141
142     cov_mat = matrix(data = NA, nrow = length(t), ncol = length(t))

```

```

143   for (i in 1:q) {
144     for (j in 1:q) {
145       cov_mat[i,j] = sigma2*rho^(abs(i-j))
146     }
147   }
148
149   # localized estimates
150   mu_local = mu[ind:(ind + 2)]
151   data_local = data[,ind:(ind + 2)]
152   cov_mat_local = cov_mat[ind:(ind+2), ind:(ind+2)]
153
154   exponent = 0
155   for(i in 1:n){
156     exponent = exponent + t(data_local[i,] - mu_local)%%(solve(cov_mat_local))%%(data_local[i,] - mu_local)
157   }
158
159   log_lik = - (n*length(ind:(ind+2))/2)*log(2*pi) - (n/2)*log(det(cov_mat_local)) - (1/2)*exponent
160   return(-log_lik)
161 }
162
163 # The following is for verification purpose for whether we are able to get the
164 # ISRP estimates by maximizing the localized likelihood function
165 for(j in 1:(q-2)){
166   ind = j
167   param_0 = c(0.4, 100, 3, 0.1)
168   out = optim(param_0, fun_local_likelihood, method = "L-BFGS-B",
169             hessian = TRUE, lower = c(0.001, 50, 0.0001, -0.001),
170             upper = c(3, 150, 10, 0.999))
171   est_local[j, ] = out$par
172 }
173 colnames(est_local) = c("r", "K", "sigma2", "rho")
174
175 par(mfrow = c(2,2))
176 plot(1:(q-2), est_local[,1], type = "b", col = "red", lwd = 2,
177      main = expression(widehat(r(Delta~t))[MLE]),
178      ylab = expression(hat(r)))
179 plot(1:(q-2), est_local[,2], type = "b", col = "red", lwd = 2,
180      main = expression(widehat(K(Delta~t))[MLE]),
181      ylab = expression(hat(K)))
182 plot(1:(q-2), est_local[,3], type = "b", col = "red", lwd = 2,
183      main = expression(widehat(sigma^2)),
184      ylab = expression(hat(sigma^2)))
185 plot(1:(q-2), est_local[,4], type = "b", col = "red", lwd = 2,
186      main = expression(widehat(rho)),
187      ylab = expression(hat(rho)))

```

188 In this section we validate the asymptotic distribution of the test statistic $(T_1, T_2, \dots, T_{q-4})$ when the null
189 hypothesis is true. The following experiment will be carried for different n values replicate this process for
190 $n = 100$ times and simulation of the distribution of the test statistics under the null hypothesis.

```

191 n=100
192 rep = 500
193 est_local_list = list()

```

```

194 x0_mean_vals = numeric(length = rep)
195 for(l in 1:rep){
196   set.seed(l)
197   print(l)
198   data = rmvnorm::rmvnorm(n, mean = mu, sigma = cov_mat)
199   x0_mean_vals[l] = mean(data[,1])
200   est_local = matrix(data = NA, nrow = q-2, ncol = 4)
201
202   for(j in 1:(q-2)){
203     ind = j
204     param_0 = c(0.4, 100, 3, 0.1)
205     out = optim(param_0, fun_local_likelihood, method = "L-BFGS-B",
206               hessian = TRUE, lower = c(0.001, 50, 0.0001, -0.001),
207               upper = c(3, 150, 10, 0.999))
208     est_local[j, ] = out$par
209   }
210   colnames(est_local) = c("r", "K", "sigma2", "rho")
211   est_local_list[[l]] = est_local
212 }
213
214 # Plot the estimates based on replication
215 # Distribution of ISRP estimates of r using localized maximum likelihood
216 mat_est_r_MLE = matrix(data = NA, nrow = q-2, ncol = rep)
217 for (j in 1:rep) {
218   mat_est_r_MLE[,j] = est_local_list[[j]][,1]
219 }
220 # Distribution of ISRP estimates of K using localized maximum likelihood
221 mat_est_K_MLE = matrix(data = NA, nrow = q-2, ncol = rep)
222 for (j in 1:rep) {
223   mat_est_K_MLE[,j] = est_local_list[[j]][,2]
224 }
225
226 # Estimation of modified RGR and its distribution
227 modified_RGR = matrix(data = NA, nrow = q-2, ncol = rep)
228 for (i in 1:(q-2)) {
229   numerator = mat_est_r_MLE[i,]*exp(-mat_est_r_MLE[i,]*t[i]/3)*((mat_est_K_MLE[i,])^(1/3) - (x0_mean_vals)^(1/3))
230   denominator = (mat_est_K_MLE[i,])^(1/3) + exp(-mat_est_r_MLE[i,]*t[i]/3)*((x0_mean_vals)^(1/3) - (mat_est_K_MLE[i,])^(1/3))
231   modified_RGR[i,] = numerator/denominator
232 }
233
234 # Computation of the Test Statistics and its distribution
235 V = matrix(data = NA, nrow = q-2, ncol = rep)
236 for (i in 1:(q-2)) {
237   V[i, ] = log(1/modified_RGR[i,] + 1/ mat_est_r_MLE[i,])
238 }
239 print(head(V))
240 sum(is.na(V))
241
242 test_stat_mle = matrix(data = NA, nrow = q-4, ncol = rep)
243 for (i in 1:(q-4)) {
244   test_stat_mle[i,] = V[i+2,] - 2*V[i+1,] + V[i,]
245 }
246 # Path setting
247 setwd("D:/My Paper/3rd Paper MLE vs ISRP/MLE ISRP/Von_bertalanffy_0-19/sigma = 0.1 _0.4_3")

```

```

248 write.table(test_stat_mle, file = "VB_Test_STAT_500.txt", row.names = FALSE,
249             col.names = FALSE)
250 # Histogram of the test statistics under the null distribution (required)
251 par(mfrow = c(1,1))
252 for(i in 1:(q-4)){
253
254     hist( test_mle_final[i,]/sd(test_mle_final[i,]), probability = TRUE,
255         main = '' , xlab = expression(widehat(T[t])), breaks = 10)
256     curve(dnorm(x), add = TRUE, col = "red", lwd = 2, lty = 2)
257 }

```

258 2. Power curve plotting

```

259 # The following parameters will be considered as the fixed set up for simulation.
260 K = 100
261 x0 = 10
262 r = 0.4
263 t = seq(from = 0, to = 19, by = 1)
264 q = length(t)
265
266 h = numeric(length = length(t)-1)
267 for(i in 1:(length(t)-1)){
268     h[i] = t[i+1] - t[i]
269 }
270
271
272 mu = K*((1 + ((x0/K)^(1/3) - 1)*exp(-r*(t/3)))^3) # Von Bertalanffy mean function
273 par(mfrow = c(1,1))
274 plot(t, mu, col = "red", lwd=2, type = "b", pch = 19,
275      ylab = "size, X(t)", xlab = "time (t)")
276
277 sigma2 = 0.1 # variance of X_t
278 rho = 0.1 # corr(X_t, X_{t+1})
279 cov_mat = matrix(data = NA, nrow = length(t), ncol = length(t))
280 for (i in 1:length(t)) {
281     for (j in 1:length(t)) {
282         cov_mat[i,j] = sigma2*rho^(abs(i-j))
283     }
284 }
285 n = 100 # number of independent trajectories
286 data = mvtnorm::rmvnorm(n, mean = mu, sigma = cov_mat)
287 matplot(t, t(data), type = "l")
288
289 abline(v = c(3, 6, 9, 12, 15, 18), lwd = 2, col = "red", lty = 2)
290 text(4.5, 70, expression(widehat(r(Delta~ t))))
291 # Likelihood function for computing overall estimates that is it will give a single estimate of the
292 # parameters using the data at all time points 1 to q.
293 fun_likelihood = function(param){
294     r = param[1]
295     K = param[2]
296     sigma2 = param[3]
297     rho = param[4]
298     mu = K*((1 + ((x0/K)^(1/3) - 1)*exp(-r*(t/3)))^3)
299

```

```

300   cov_mat = matrix(data = NA, nrow = length(t), ncol = length(t))
301   for (i in 1:q) {
302     for (j in 1:q) {
303       cov_mat[i,j] = sigma2*rho^(abs(i-j))
304     }
305   }
306   exponent = 0
307   for(i in 1:n){
308     exponent = exponent + t(data[i,]-mu)%*(solve(cov_mat))%*(data[i,]-mu)
309   }
310
311
312   log_lik = - (n*q/2)*log(2*pi) - (n/2)*log(det(cov_mat)) - (1/2)*exponent
313   return(-log_lik)
314 }
315 param_0 = c(0.3, 100, 3, 0.3)
316 out = optim(param_0, fun_likelihood, method = "L-BFGS-B",
317           hessian = TRUE, lower = c(0.001, 50, 0.0001, -0.001), upper = c(3, 150, 10, 0.999))
318 out$par
319 solve(out$hessian)

```

320 The Main programme starts here ISRP estimates by maximizing localized likelihood. The following program
 321 will estimate the ISRP of r, K, σ^2 and ρ by maximizing localized likelihood function that will use only the data
 322 for three consecutive time points. Therefore, for $q = 20$ time points, we will obtain 18 estimates of r and K .
 323 Note that we did not provide the ISRP estimate of X_0 . We estimated X_0 by taking the average of all the
 324 observations at first time point. Here, we consider Von Bertalanffy model as the test bed model. For other
 325 model, we need to change the mean (μ) according to the different growth model. Also for different σ^2 and ρ
 326 values, we need to set different σ^2 and ρ values.

```

327 est_local = matrix(data = NA, nrow = q-2, ncol = 4)
328
329 fun_local_likelihood = function(param){
330   r = param[1]
331   K = param[2]
332   sigma2 = param[3]
333   rho = param[4]
334   mu = K*((1 + ((x0/K)^(1/3) - 1)*exp(-r*(t/3)))^3)
335
336   cov_mat = matrix(data = NA, nrow = length(t), ncol = length(t))
337   for (i in 1:q) {
338     for (j in 1:q) {
339       cov_mat[i,j] = sigma2*rho^(abs(i-j))
340     }
341   }
342
343   # localized estimates
344   mu_local = mu[ind:(ind + 2)]
345   data_local = data[,ind:(ind + 2)]
346   cov_mat_local = cov_mat[ind:(ind+2), ind:(ind+2)]
347
348   exponent = 0

```

```

349   for(i in 1:n){
350       exponent = exponent + t(data_local[i,] - mu_local)%*(solve(cov_mat_local))%*(data_local[i,] - mu_local)
351   }
352
353   log_lik = - (n*length(ind:(ind+2))/2)*log(2*pi) - (n/2)*log(det(cov_mat_local)) - (1/2)*exponent
354   return(-log_lik)
355 }
356
357 # The following is for verification purpose for whether we are able to get the
358 # ISRP estimates by maximizing the localized likelihood function
359 for(j in 1:(q-2)){
360     ind = j
361     param_0 = c(0.3, 100, 3, 0.3)
362     out = optim(param_0, fun_local_likelihood, method = "L-BFGS-B",
363               hessian = TRUE, lower = c(0.001, 50, 0.0001, -0.001),
364               upper = c(3, 150, 10, 0.999))
365     est_local[j, ] = out$par
366 }
367 colnames(est_local) = c("r", "K", "sigma2", "rho")
368
369 par(mfrow = c(2,2))
370 plot(1:(q-2), est_local[,1], type = "b", col = "red", lwd = 2,
371      main = expression(widehat(r(Delta~t))[MLE]),
372      ylab = expression(hat(r)))
373 plot(1:(q-2), est_local[,2], type = "b", col = "red", lwd = 2,
374      main = expression(widehat(K(Delta~t))[MLE]),
375      ylab = expression(hat(K)))
376 plot(1:(q-2), est_local[,3], type = "b", col = "red", lwd = 2,
377      main = expression(widehat(sigma^2)),
378      ylab = expression(hat(sigma^2)))
379 plot(1:(q-2), est_local[,4], type = "b", col = "red", lwd = 2,
380      main = expression(widehat(rho)),
381      ylab = expression(hat(rho)))
382
383
384 # In this section we validate the asymptotic distribution of the test statistic (T_1, T_2, ..., T_{q-4})
385 # when the null hypothesis is true. The following experiment carried for n = 100 but we run the program
386 # for different values of n (n = 10, n = 20, n = 50, n = 70, n = 100). Replicate this process
387 # for 1000 times and simulation of the distribution of the test statistics under the null hypothesis.
388
389 # We create a path to save the data
390 setwd("C:/Users/student/Desktop/Aktar/VON BERTALANFFY/n = 100")
391 n = 100      # As folder changes, change the value of n
392 rep = 1000
393 est_local_list = list()
394 x0_mean_vals = numeric(length = rep)
395 for(l in 1:rep){
396     set.seed(1)
397     print(l)
398     data = rmvnorm::rmvnorm(n, mean = mu, sigma = cov_mat)
399     x0_mean_vals[l] = mean(data[,1])
400     est_local = matrix(data = NA, nrow = q-2, ncol = 4)
401
402     for(j in 1:(q-2)){

```

```

403     ind = j
404     param_0 = c(0.3, 100, 3, 0.3)
405     out = optim(param_0, fun_local_likelihoood, method = "L-BFGS-B",
406               hessian = TRUE, lower = c(0.001, 50, 0.0001, -0.001),
407               upper = c(3, 150, 10, 0.999))
408     est_local[j, ] = out$par
409   }
410   colnames(est_local) = c("r", "K", "sigma2", "rho")
411   est_local_list[[l]] = est_local
412 }
413
414 # Plot the estimates based on replication
415 # Distribution of ISRP estimates of r using localized maximum likelihood
416 mat_est_r_MLE = matrix(data = NA, nrow = q-2, ncol = rep)
417 for (j in 1:rep) {
418   mat_est_r_MLE[,j] = est_local_list[[j]][,1]
419 }
420 write.table(mat_est_r_MLE, file = "r_MLE_ISRP.txt", row.names = FALSE,
421            col.names = FALSE)
422 matplot(1:(q-2), mat_est_r_MLE, type = "l",
423        main = expression(widehat(r(Delta~t))[MLE]),
424        ylab = expression(hat(r)))
425 legend("topright", legend = paste("n = ", n), bty = "n", cex = 1.5)
426
427 # Distribution of ISRP estimates of K using localized maximum likelihood
428 mat_est_K_MLE = matrix(data = NA, nrow = q-2, ncol = rep)
429 for (j in 1:rep) {
430   mat_est_K_MLE[,j] = est_local_list[[j]][,2]
431 }
432 write.table(mat_est_K_MLE, file = "K_MLE_ISRP.txt", row.names = FALSE,
433            col.names = FALSE)
434 matplot(1:(q-2), mat_est_K_MLE, type = "l",
435        main = expression(widehat(K(Delta~t))[MLE]),
436        ylab = expression(hat(K)))
437 legend("topright", legend = paste("n = ", n), bty = "n", cex = 1.5)
438
439 # Estimation of modified RGR and its distribution
440 modified_RGR = matrix(data = NA, nrow = q-2, ncol = rep)
441 for (i in 1:(q-2)) {
442   numerator = mat_est_r_MLE[i,]*exp(-mat_est_r_MLE[i,]*(t[i]/3))*
443             ((mat_est_K_MLE[i,])^(1/3) - (x0_mean_vals)^(1/3))
444   denominator = (mat_est_K_MLE[i,])^(1/3) + exp(-mat_est_r_MLE[i,]*(t[i]/3))*
445             ((x0_mean_vals)^(1/3) - (mat_est_K_MLE[i,])^(1/3))
446   modified_RGR[i,] = numerator/denominator
447 }
448 write.table(modified_RGR, file = "modified_RGR.txt", row.names = FALSE,
449            col.names = FALSE)
450
451 par(mfrow = c(2,3))
452 for(i in 1:(q-2)){
453   hist(modified_RGR[i,], probability = TRUE, main = paste("t = ", i),
454        xlab = expression(hat(R[t])))
455 }
456

```

```

457 # Computation of the Test Statistics and its distribution
458 V = matrix(data = NA, nrow = q-2, ncol = rep)
459 for (i in 1:(q-2)) {
460   V[i, ] = log(1/modified_RGR[i,] + 1/ mat_est_r_MLE[i,])
461 }
462 print(head(V))
463 sum(is.na(V))
464
465 test_stat = matrix(data = NA, nrow = q-4, ncol = rep)
466 for (i in 1:(q-4)) {
467   test_stat[i,] = V[i+2,] - 2*V[i+1,] + V[i,]
468 }
469 write.table(test_stat, file = "test_stat.txt", row.names = FALSE,
470             col.names = FALSE)
471
472 # Histogram of the test statistics under the null distribution (required)
473 par(mfrow = c(3,3))
474 for(i in 1:(q-4)){
475
476   hist( test_stat[i,]/sd(test_stat[i,]), probability = TRUE, main = paste("t = ", i),
477        xlab = expression(widehat(T[t])), breaks = 10)
478   curve(dnorm(x), add = TRUE, col = "red", lwd = 2, lty = 2)
479 }
480
481 # Following we simulate the observations of size n = 100 from the Generalized Von Bertalanffy model
482 # f or different choices of theta. We simulate the power curves for each theta values based on 500 replicatio
483 # The null hypothesis is  $\theta = 1/3$ , that is the Von Bertalanffy model. The other parameters' choices are
484 #  $K = 100$ ,  $x_0 = 10$ ,  $r = 0.3$   $q = 20$  (number of time points),  $\sigma^2 = 2$ ,  $\rho = 0.5$ 
485 # There will be  $q-4 = 16$  power curves and the first power curves based on the statistics  $T_1$  will test
486 #whether the data followed Von Bertalanffy model ( $\theta = 1/3$ ) in the time interval  $\{1,2,3,4,5\}$ ,
487 # similarly  $T_2$  will check in the interval  $\{2,3,4,5,5\}$  and so on till  $T_{\{q-4\}}$ .
488 # All the power curves are generated based on 1000 replications.
489 # Simulation of Power function for different values of theta.
490
491 setwd("C:/Users/student/Desktop/Aktar/VON BERTALANFFY/n = 100/Power Curves")
492
493 theta_vals = seq(from = 0.23, to = 0.80, length.out = 30)
494 power_vals = matrix(data = NA, nrow = length(theta_vals),
495                     ncol = q-4)
496
497 for (k in 1:length(theta_vals)) {
498   theta = theta_vals[k]
499   mu_theta_GVB =  $K*((1 + ((x_0/K)^(theta) - 1)*exp(-r*t*theta))^(1/theta))$ 
500   # Generalized Von Bertalanffy mean function
501
502   rep = 1000
503   est_local_list = list()
504   x0_mean_vals = numeric(length = rep)
505   for(l in 1:rep){
506     set.seed(l)
507     print(l)
508     data = rmvnorm::rmvnorm(n, mean = mu_theta_GVB, sigma = cov_mat)
509     x0_mean_vals[l] = mean(data[,1])
510     est_local = matrix(data = NA, nrow = q-2, ncol = 4)

```

```

511
512   for(j in 1:(q-2)){
513     ind = j
514     param_0 = c(0.3, 100, 3, 0.3)
515     out = optim(param_0, fun_local_likelihoood, method = "L-BFGS-B",
516               hessian = TRUE, lower = c(0.001, 50, 0.0001, -0.001),
517               upper = c(3, 150, 10, 0.999))
518     est_local[j, ] = out$par
519   }
520   colnames(est_local) = c("r", "K", "sigma2", "rho")
521   est_local_list[[1]] = est_local
522 }
523 mat_est_r_MLE = matrix(data = NA, nrow = q-2, ncol = rep)
524 for (j in 1:rep) {
525   mat_est_r_MLE[,j] = est_local_list[[j]][,1]
526 }
527
528 mat_est_K_MLE = matrix(data = NA, nrow = q-2, ncol = rep)
529 for (j in 1:rep) {
530   mat_est_K_MLE[,j] = est_local_list[[j]][,2]
531 }
532
533 # Estimation of modified RGR
534 modified_RGR = matrix(data = NA, nrow = q-2, ncol = rep)
535 for (i in 1:(q-2)) {
536   numerator = mat_est_r_MLE[i,]*exp(-mat_est_r_MLE[i,]*(t[i]/3))*
537     ((mat_est_K_MLE[i,])^(1/3) - (x0_mean_vals)^(1/3))
538   denominator = (mat_est_K_MLE[i,])^(1/3) + exp(-mat_est_r_MLE[i,]*(t[i]/3))*
539     ((x0_mean_vals)^(1/3) - (mat_est_K_MLE[i,])^(1/3))
540   modified_RGR[i,] = numerator/denominator
541 }
542
543 V = matrix(data = NA, nrow = q-2, ncol = rep)
544 for (i in 1:(q-2)) {
545   V[i, ] = log(1/modified_RGR[i,] + 1/ mat_est_r_MLE[i,])
546 }
547
548 test_stat = matrix(data = NA, nrow = q-4, ncol = rep)
549 for (i in 1:(q-4)) {
550   test_stat[i,] = V[i+2,] - 2*V[i+1,] + V[i,]
551 }
552
553 alpha = 0.05
554 prop_reject = numeric(length = q-4)
555 for (i in 1:(q-4)) {
556   prop_reject[i] = sum(abs(test_stat[i,]/sd(test_stat[i,]))>qnorm(1-alpha/2))/rep
557 }
558 power_vals[k, ] = prop_reject
559 }
560
561 write.table(power_vals, file = "power.txt", row.names = FALSE,
562           col.names = FALSE)
563
564

```

```

565 par(mfrow = c(2,3))
566 for (i in 1:(q-4)) {
567     plot(theta_vals, power_vals[,i], type = "l", lwd = 2, col = "red",
568          ylab = expression(beta(theta)), xlab = expression(theta),
569          main = paste("t = ", i))
570 }
571
572

```

573 3. stability

574 Stability of the estimators of the parameter r and K derived by Bhowmick et al.,(2014)'s method.

```

575 # The following parameters will be considered as the fixed set up for simulation.
576 K = 100 # carrying capacity
577 x0 = 10 # initial population size
578 r = 0.4 # initial intrinsic growth rate
579 t = seq(from = 0, to = 19, by = 1) # time # change for different time series
580 time_int=length(t)-2
581
582 h = numeric(length = length(t)-1) # difference between two consecutive time points
583 for(i in 1:(length(t)-1)){
584     h[i] = t[i+1] - t[i]
585 }
586
587
588 mu= K*((1 + ((x0/K)^(1/3) - 1)*exp(-r*t/3))^3) # Generalized VB mean # change for different growth models
589 RGR= (r*exp(-r*t/3)*(K^(1/3)-x0^(1/3)))/(K^(1/3)+exp(-r*t/3)*(x0^(1/3)-K^(1/3)))
590 par(mfrow = c(1,1))
591 par(mar = c(5.1, 5.1, 4.1, 2.1))
592
593 plot(t, mu, col = "red", lwd=2, type = "b", pch = 19, ylab = "size, X(t)", xlab = "time (t)")
594
595 rep = 1000 # number of simulations
596 n = 50 # number of trajectories
597 sigma2 = 1 # variance of X(t) # Change for different values
598 rho = 0.1 # Correlation between X(t) and X(t+1)
599 x0_mean_vals = numeric(length = rep)
600 cov_mat = matrix(data = NA, nrow = length(t), ncol = length(t)) # covariance matrix
601 for (i in 1:length(t)) {
602     for (j in 1:length(t)) {
603         cov_mat[i,j] = sigma2*rho^(abs(i-j))
604     }
605 }
606
607 r_hat_vals = matrix(data = NA, nrow = rep, ncol = time_int)
608
609 K_hat_vals = matrix(data = NA, nrow = rep, ncol = time_int)
610
611 modified_RGR = matrix(data = NA, nrow = rep, ncol = time_int)
612
613 # simulation of artificial data sets
614 for (i in 1:rep) {

```

```

615 library(mvtnorm)
616 data = mvtnorm::rmvnorm(n, mean = mu, sigma = cov_mat)
617 x0_mean_vals[i] = mean(data[,1])
618 data_mean = colMeans(data)
619
620 # Interval estimates of r
621 for(j in 1:(length(t)-2)){
622   d1= (data_mean[j+1])^(1/3) - (data_mean[j])^(1/3)
623   d2= (data_mean[j+2])^(1/3) - (data_mean[j+1])^(1/3)
624   d12=d1/d2
625   r_hat_vals[i,j]=(3/h[j])*log(d12)
626   K_hat_vals[i,j]= ((x0_mean_vals[i])^(1/3)-(d1/(exp(-r_hat_vals[i,j]*(t[j]/3))*
627                                     (exp(-r_hat_vals[i,j]*(h[j]/3))-1))))^(3)
628   modified_RGR[i,j]= (r_hat_vals[i,j]*exp(-r_hat_vals[i,j]*(t[j]/3))*
629                       ((K_hat_vals[i,j])^(1/3) - (x0_mean_vals[i])^(1/3)))/
630   ((K_hat_vals[i,j])^(1/3) + exp(-r_hat_vals[i,j]*(t[j]/3))*
631     ((x0_mean_vals[i])^(1/3)-(K_hat_vals[i,j])^(1/3)))
632 }
633 }
634
635 par(mfrow = c(1,1))
636 par(mar = c(5.1, 5.1, 4.1, 2.1))
637
638 for (j in 1:(length(t)-2)) {
639   hist(r_hat_vals[,j], probability = TRUE, col = "lightgreen",
640        xlab = expression(widehat(r[t])), main = '', breaks = 10, cex.axis=1.9, cex.lab=1.9)
641   points(r, 0, col = "red", pch = 19, cex = 2.8)
642 }
643
644 for (j in 1:(length(t)-2)) {
645   hist(K_hat_vals[,j], probability = TRUE, col = "lightgreen",
646        xlab = expression(widehat(K[t])), main = '', breaks = 10, cex.axis=1.9, cex.lab=1.9)
647   points(K, 0, col = "red", pch = 19, cex = 2.8)
648 }
649
650 }
651

```

652 Stability of the estimators of the parameter r K derived by localized MLE method.

```

653 # The following parameters will be considered as the fixed set up for simulation.
654 K = 100 # carrying capacity
655 x0 = 10 # initial population size
656 r = 0.4 # initial intrinsic growth rate
657 t = seq(from = 0, to = 19, by = 1) # time # change for different time series
658 q = length(t)
659
660 h = numeric(length = length(t)-1) # difference between two consecutive time points
661 for(i in 1:(length(t)-1)){
662   h[i] = t[i+1] - t[i]
663 }
664
665
666 # For other growth model change mu according to that growth model

```

```

667 mu= K*((1 + ((x0/K)^(1/3) - 1)*exp(-r*t/3))^3) # Generalized VB mean
668 RGR= (r*exp(-r*t/3)*(K^(1/3)-x0^(1/3)))/(K^(1/3)+exp(-r*t/3)*(x0^(1/3)-K^(1/3)))
669 par(mfrow = c(1,1))
670 plot(t, mu, col = "red", lwd=2, type = "l", pch = 19,
671      ylab = "Size X(t)", xlab = "Time (t)")
672
673 # For different sigma2 and rho values, we have to change the values and set new values here.
674
675 sigma2 = 1 # variance of X(t) # Change for different values
676 rho = 0.1 # corr(X_t, X_{t+1})
677 cov_mat = matrix(data = NA, nrow = length(t), ncol = length(t))
678 for (i in 1:length(t)) {
679   for (j in 1:length(t)) {
680     cov_mat[i,j] = sigma2*rho^(abs(i-j))
681   }
682 }
683
684 n = 50 # number of independent trajectories
685 data = mvtnorm::rmvnorm(n, mean = mu, sigma = cov_mat)
686 matplot(t, t(data), type = "l")
687
688 # Likelihood function
689 fun_likelihoood = function(param){
690   r = param[1]
691   K = param[2]
692   sigma2 = param[3]
693   rho = param[4]
694   mu = K*((1 + ((x0/K)^(1/3) - 1)*exp(-r*t/3))^3)
695
696   cov_mat = matrix(data = NA, nrow = length(t), ncol = length(t))
697   for (i in 1:q) {
698     for (j in 1:q) {
699       cov_mat[i,j] = sigma2*rho^(abs(i-j))
700     }
701   }
702   exponent = 0
703   for(i in 1:n){
704     exponent = exponent + t(data[i,]-mu)%*(solve(cov_mat))%*(data[i,]-mu)
705   }
706
707   log_lik = - (n*q/2)*log(2*pi) - (n/2)*log(det(cov_mat)) - (1/2)*exponent
708   return(-log_lik)
709 }
710
711 param_0 = c(0.4, 100, 3, 0.1)
712 out = optim(param_0, fun_likelihoood, method = "L-BFGS-B",
713            hessian = TRUE, lower = c(0.001, 50, 0.0001, -0.001), upper = c(3, 150, 10, 0.999))
714 out$par
715 solve(out$hessian)
716
717 # Localized likelihood function
718
719 est_local = matrix(data = NA, nrow = q-2, ncol = 4)
720

```

```

721 fun_local_likelihood = function(param){
722   r = param[1]
723   K = param[2]
724   sigma2 = param[3]
725   rho = param[4]
726   mu = K*((1 + ((x0/K)^(1/3) - 1)*exp(-r*t/3))^3)
727
728   cov_mat = matrix(data = NA, nrow = length(t), ncol = length(t))
729   for (i in 1:q) {
730     for (j in 1:q) {
731       cov_mat[i,j] = sigma2*rho^(abs(i-j))
732     }
733   }
734
735   # localized estimates
736   mu_local = mu[ind:(ind + 2)]
737   data_local = data[,ind:(ind + 2)]
738   cov_mat_local = cov_mat[ind:(ind+2), ind:(ind+2)]
739
740   exponent = 0
741   for(i in 1:n){
742     exponent = exponent + t(data_local[i,] - mu_local)%%(solve(cov_mat_local))%%(data_local[i,] - mu_local)
743   }
744
745   log_lik = - (n*length(ind:(ind+2))/2)*log(2*pi) - (n/2)*log(det(cov_mat_local)) - (1/2)*exponent
746   return(-log_lik)
747 }
748
749 # Verification of localized likelihood
750
751 for(j in 1:(q-2)){
752   ind = j
753   param_0 = c(0.4, 100, 3, 0.1)
754   out = optim(param_0, fun_local_likelihood, method = "L-BFGS-B",
755             hessian = TRUE, lower = c(0.001, 50, 0.0001, -0.001),
756             upper = c(3, 150, 10, 0.999))
757   est_local[j, ] = out$par
758 }
759 colnames(est_local) = c("r", "K", "sigma2", "rho")
760
761 par(mfrow = c(2,2))
762 plot(1:(q-2), est_local[,1], type = "b", col = "red", lwd = 2,
763      main = expression(widehat(r(Delta~t))[MLE]),
764      ylab = expression(hat(r)))
765 plot(1:(q-2), est_local[,2], type = "b", col = "red", lwd = 2,
766      main = expression(widehat(K(Delta~t))[MLE]),
767      ylab = expression(hat(K)))
768 plot(1:(q-2), est_local[,3], type = "b", col = "red", lwd = 2,
769      main = expression(widehat(sigma^2)),
770      ylab = expression(hat(sigma^2)))
771 plot(1:(q-2), est_local[,4], type = "b", col = "red", lwd = 2,
772      main = expression(widehat(rho)),
773      ylab = expression(hat(rho)))
774

```

```

775
776 # Localized MLE of the parameters
777
778 # Path to save the values
779 setwd("D:/My Paper/3rd Paper MLE vs ISRP/Final_Plot_file/Von/1-20_1")
780 n=50 # Number of trajectories
781 rep = 1000 # Replication no.
782 est_local_list = list()
783 x0_mean_vals = numeric(length = rep)
784 for(l in 1:rep){
785   set.seed(l)
786   print(l)
787   data = rmvnorm::rmvnorm(n, mean = mu, sigma = cov_mat)
788   x0_mean_vals[l] = mean(data[,1])
789   est_local = matrix(data = NA, nrow = q-2, ncol = 4)
790
791   for(j in 1:(q-2)){
792     ind = j
793     param_0 = c(0.4, 100, 3, 0.1)
794     out = optim(param_0, fun_local_likelihoood, method = "L-BFGS-B",
795               hessian = TRUE, lower = c(0.001, 50, 0.0001, -0.001),
796               upper = c(3, 150, 10, 0.999))
797     est_local[j, ] = out$par
798   }
799   colnames(est_local) = c("r", "K", "sigma2", "rho")
800   est_local_list[[l]] = est_local
801 }
802
803 # Plot the estimates based on replication
804 # Distribution of ISRP estimates of r using localized maximum likelihood
805 mat_est_r_MLE = matrix(data = NA, nrow = q-2, ncol = rep)
806 for (j in 1:rep) {
807   mat_est_r_MLE[,j] = est_local_list[[j]][,1]
808 }
809 write.table(mat_est_r_MLE, file = "r_MLE_ISRP.txt", row.names = FALSE,
810           col.names = FALSE)
811 matplot(1:(q-2), mat_est_r_MLE, type = "l",
812       main = expression(widehat(r(Delta~t))[MLE]),
813       ylab = expression(hat(r)))
814 legend("topright", legend = paste("n = ", n), bty = "n", cex = 1.5)
815
816 # Distribution of ISRP estimates of K using localized maximum likelihood
817 mat_est_K_MLE = matrix(data = NA, nrow = q-2, ncol = rep)
818 for (j in 1:rep) {
819   mat_est_K_MLE[,j] = est_local_list[[j]][,2]
820 }
821 write.table(mat_est_K_MLE, file = "K_MLE_ISRP.txt", row.names = FALSE,
822           col.names = FALSE)
823 matplot(1:(q-2), mat_est_K_MLE, type = "l",
824       main = expression(widehat(K(Delta~t))[MLE]),
825       ylab = expression(hat(K)))
826 legend("topright", legend = paste("n = ", n), bty = "n", cex = 1.5)
827
828 # Estimation of modified RGR and its distribution

```

```

829 modified_RGR = matrix(data = NA, nrow = q-2, ncol = rep)
830 for (i in 1:(q-2)) {
831     numerator = mat_est_r_MLE[i,]*exp(-mat_est_r_MLE[i,]*t[i]/3)*((mat_est_K_MLE[i,])^(1/3) - (x0_mean_vals)^(1/3))
832     denominator = (mat_est_K_MLE[i,])^(1/3) + exp(-mat_est_r_MLE[i,]*t[i]/3)*((x0_mean_vals)^(1/3) - (mat_est_K_MLE[i,])^(1/3))
833     modified_RGR[i,] = numerator/denominator
834 }
835 write.table(modified_RGR, file = "modified_RGR.txt", row.names = FALSE,
836             col.names = FALSE)
837
838 par(mfrow = c(1,1))
839 par(mar = c(5.1, 5.1, 4.1, 2.1))
840
841 for(i in 1:(q-2)){
842     hist(mat_est_r_MLE[i,], probability = TRUE, main = '', breaks = 10,
843          xlab = expression(widehat(r[t])), cex.axis=1.9, cex.lab=1.9)
844     points(r, 0, col = "red", pch = 19, cex = 2.8)
845 }
846
847 for(i in 1:(q-2)){
848     hist(mat_est_K_MLE[i,], probability = TRUE, main = '', breaks = 10,
849          xlab = expression(widehat(K[t])), cex.axis=1.9, cex.lab=1.9)
850     points(K, 0, col = "red", pch = 19, cex = 2.8)
851 }
852
853 for(i in 1:(q-2)){
854     hist(modified_RGR[i,], probability = TRUE, main = '', breaks = 10,
855          xlab = expression(widehat(R[t])), cex.axis=1.9, cex.lab=1.9)
856     points(RGR[i], 0, col = "red", pch = 19, cex = 2.8)
857 }
858

```

859 4. Efficiency

```

860 # Testing of efficiency by using Multivariate Delta Method
861 # m = 5 ## Number of sigma values
862 p = 1000 ### repetition
863 index.sum1 = numeric(p); index.sum1
864
865 for(a in 1:p)
866 {
867     K = 100 # carrying capacity
868     x0 = 10 # initial population size
869     r0 = 0.4 # initial intrinsic growth rate
870     t = seq(from = 1, to = 20, by = 1) # time
871     q=length(t)
872
873     h = numeric(length = length(t)-1) # difference between two consecutive time points
874     for(i in 1:(length(t)-1)){
875         h[i] = t[i+1] - t[i]
876     }
877
878     mu = K*(1 + ((x0/K)^(1/3) - 1)*exp((-r0*t)/3))^3 # Von Bertalanffy mean function
879     par(mfrow = c(1,1))
880     plot(t, mu, col = "red", lwd=2, type = "b", pch = 19,

```

```

881     ylab = "size, X(t)", xlab = "time (t)")
882
883
884     sigma1 = 0.1 # variance of X_t
885     rho = 0.1    # corr(X_t, X_{t+1})
886
887
888     cov_mat1 = matrix(data = NA, nrow = length(t), ncol = length(t))
889     for (i in 1:length(t)) {
890         for (j in 1:length(t)) {
891             cov_mat1[i,j] = sigma1*rho^(abs(i-j))
892         }
893     }
894
895     n = 100 # number of trajectories
896     library(mvtnorm)
897     data = mvtnorm::rmvnorm(n, mean = mu, sigma = cov_mat1)
898     matplot(t(data), type = "l", ylab = "Size", xlab = "Time")
899     legend("topleft", legend = paste("n = ", n), bty = "n")
900
901     d1 = matrix(data = NA, nrow = n, ncol = length(t)-2)
902     d2 = matrix(data = NA, nrow = n, ncol = length(t)-2)
903
904     for(j in 1:(length(t)-2)){
905         d1[,j] = ((data[,j+1])^(1/3)) - ((data[,j])^(1/3))
906         d2[,j] = ((data[,j+2])^(1/3)) - ((data[,j+1])^(1/3))
907     }
908
909     # Computation for r_hat
910     r_hat_vals = matrix(data = NA, nrow = n, ncol = length(t)-2)
911     for (j in 1:(length(t)-2)) {
912         r_hat_vals[,j] = (3/h[j])*log(d1[,j]/d2[,j])
913     }
914
915     selected_index = rep(FALSE, n)
916     for (i in 1:n) {
917         if(sum(is.na(r_hat_vals[i,])) ==0)
918             selected_index[i] = TRUE
919     }
920
921     selected_index # List of selected rows for further calculations
922     index.sum1[a] = sum(selected_index)
923 }
924 index.sum1
925
926 # For second sigma value #
927 index.sum2 = numeric(p); index.sum2
928 for(a in 1:p)
929 {
930     K = 100 # carrying capacity
931     x0 = 10 # initial population size
932     r0 = 0.4 # initial intrinsic growth rate
933     t = seq(from = 1, to = 20, by = 1) # time
934     q=length(t)

```

```

935
936 h = numeric(length = length(t)-1) # difference between two consecutive time points
937 for(i in 1:(length(t)-1)){
938   h[i] = t[i+1] - t[i]
939 }
940
941 mu = K*(1 + ((x0/K)^(1/3) - 1)*exp((-r0*t)/3))^3 # Von Bertalanffy mean function
942 par(mfrow = c(1,1))
943 plot(t, mu, col = "red", lwd=2, type = "b", pch = 19,
944      ylab = "size, X(t)", xlab = "time (t)")
945
946
947 sigma2 = 0.15 # variance of X_t
948 rho = 0.1 # corr(X_t, X_{t+1})
949
950 cov_mat2 = matrix(data = NA, nrow = length(t), ncol = length(t))
951 for (i in 1:length(t)) {
952   for (j in 1:length(t)) {
953     cov_mat2[i,j] = sigma2*rho^(abs(i-j))
954   }
955 }
956
957 n = 100 # number of trajectories
958 library(mvtnorm)
959 data = mvtnorm::rmvnorm(n, mean = mu, sigma = cov_mat2)
960 matplot(t(data), type = "l", ylab = "Size", xlab = "Time")
961 legend("topleft", legend = paste("n = ", n), bty = "n")
962
963 d1 = matrix(data = NA, nrow = n, ncol = length(t)-2)
964 d2 = matrix(data = NA, nrow = n, ncol = length(t)-2)
965
966 for(j in 1:(length(t)-2)){
967   d1[,j] = ((data[,j+1])^(1/3)) - ((data[,j])^(1/3))
968   d2[,j] = ((data[,j+2])^(1/3)) - ((data[,j+1])^(1/3))
969 }
970
971 # Computation for r_hat
972 r_hat_vals = matrix(data = NA, nrow = n, ncol = length(t)-2)
973 for (j in 1:(length(t)-2)) {
974   r_hat_vals[,j] = (3/h[j])*log(d1[,j]/d2[,j])
975 }
976
977
978 selected_index = rep(FALSE, n)
979 for (i in 1:n) {
980   if(sum(is.na(r_hat_vals[i,])) == 0)
981     selected_index[i] = TRUE
982 }
983
984 selected_index # List of selected rows for further calculations
985 index.sum2[a] = sum(selected_index)
986 }
987 index.sum2
988

```

```

989 # For third sigma value #
990 index.sum3 = numeric(p); index.sum3
991 for(a in 1:p)
992 {
993   K = 100 # carrying capacity
994   x0 = 10 # initial population size
995   r0 = 0.4 # initial intrinsic growth rate
996   t = seq(from = 1, to = 20, by = 1) # time
997   q=length(t)
998
999   h = numeric(length = length(t)-1) # difference between two consecutive time points
1000   for(i in 1:(length(t)-1)){
1001     h[i] = t[i+1] - t[i]
1002   }
1003
1004   mu = K*(1 + ((x0/K)^(1/3) - 1)*exp((-r0*t)/3))^3 # Von Bertalanffy mean function
1005   par(mfrow = c(1,1))
1006   plot(t, mu, col = "red", lwd=2, type = "b", pch = 19,
1007        ylab = "size, X(t)", xlab = "time (t)")
1008
1009
1010   sigma3 = 0.20 # variance of X_t
1011   rho = 0.1 # corr(X_t, X_{t+1})
1012
1013   cov_mat3 = matrix(data = NA, nrow = length(t), ncol = length(t))
1014   for (i in 1:length(t)) {
1015     for (j in 1:length(t)) {
1016       cov_mat3[i,j] = sigma3*rho^(abs(i-j))
1017     }
1018   }
1019
1020   n = 100 # number of trajectories
1021   library(mvtnorm)
1022   data = mvtnorm::rmvnorm(n, mean = mu, sigma = cov_mat3)
1023   matplot(t(data), type = "l", ylab = "Size", xlab = "Time")
1024   legend("topleft", legend = paste("n = ", n), bty = "n")
1025
1026   d1 = matrix(data = NA, nrow = n, ncol = length(t)-2)
1027   d2 = matrix(data = NA, nrow = n, ncol = length(t)-2)
1028
1029   for(j in 1:(length(t)-2)){
1030     d1[,j] = ((data[,j+1])^(1/3)) - ((data[,j])^(1/3))
1031     d2[,j] = ((data[,j+2])^(1/3)) - ((data[,j+1])^(1/3))
1032   }
1033
1034   # Computation for r_hat
1035   r_hat_vals = matrix(data = NA, nrow = n, ncol = length(t)-2)
1036   for (j in 1:(length(t)-2)) {
1037     r_hat_vals[,j] = (3/h[j])*log(d1[,j]/d2[,j])
1038   }
1039
1040   selected_index = rep(FALSE, n)
1041   for (i in 1:n) {
1042     if(sum(is.na(r_hat_vals[i,])) ==0)

```

```

1043     selected_index[i] = TRUE
1044
1045 }
1046 selected_index      # List of selected rows for further calculations
1047 index.sum3[a] = sum(selected_index)
1048 }
1049 index.sum3
1050
1051 # For fourth sigma value #
1052 index.sum4 = numeric(p); index.sum4
1053 for(a in 1:p)
1054 {
1055     K = 100 # carrying capacity
1056     x0 = 10 # initial population size
1057     r0 = 0.4 # initial intrinsic growth rate
1058     t = seq(from = 1, to = 20, by = 1) # time
1059     q=length(t)
1060
1061     h = numeric(length = length(t)-1) # difference between two consecutive time points
1062     for(i in 1:(length(t)-1)){
1063         h[i] = t[i+1] - t[i]
1064     }
1065
1066     mu = K*(1 + ((x0/K)^(1/3) - 1)*exp((-r0*t)/3))^3 # Von Bertalanffy mean function
1067     par(mfrow = c(1,1))
1068     plot(t, mu, col = "red", lwd=2, type = "b", pch = 19,
1069          ylab = "size, X(t)", xlab = "time (t)")
1070
1071
1072     sigma4 = 0.25 # variance of X_t
1073     rho = 0.1      # corr(X_t, X_{t+1})
1074
1075     cov_mat4 = matrix(data = NA, nrow = length(t), ncol = length(t))
1076     for (i in 1:length(t)) {
1077         for (j in 1:length(t)) {
1078             cov_mat4[i,j] = sigma4*rho^(abs(i-j))
1079         }
1080     }
1081
1082     n = 100 # number of trajectories
1083     library(mvtnorm)
1084     data = mvtnorm::rmvnorm(n, mean = mu, sigma = cov_mat4)
1085     matplot(t(data), type = "l", ylab = "Size", xlab = "Time")
1086     legend("topleft", legend = paste("n = ", n), bty = "n")
1087
1088     d1 = matrix(data = NA, nrow = n, ncol = length(t)-2)
1089     d2 = matrix(data = NA, nrow = n, ncol = length(t)-2)
1090
1091     for(j in 1:(length(t)-2)){
1092         d1[,j] = ((data[,j+1])^(1/3)) - ((data[,j])^(1/3))
1093         d2[,j] = ((data[,j+2])^(1/3)) - ((data[,j+1])^(1/3))
1094     }
1095
1096     # Computation for r_hat

```

```

1097   r_hat_vals = matrix(data = NA, nrow = n, ncol = length(t)-2)
1098   for (j in 1:(length(t)-2)) {
1099     r_hat_vals[,j] = (3/h[j])*log(d1[,j]/d2[,j])
1100   }
1101
1102   selected_index = rep(FALSE, n)
1103   for (i in 1:n) {
1104     if(sum(is.na(r_hat_vals[i,])) ==0)
1105       selected_index[i] = TRUE
1106   }
1107   selected_index      # List of selected rows for further calculations
1108   index.sum4[a] = sum(selected_index)
1109 }
1110 index.sum4
1111
1112 # For fifth sigma value #
1113 index.sum5 = numeric(p); index.sum5
1114 for(a in 1:p)
1115 {
1116   K = 100 # carrying capacity
1117   x0 = 10 # initial population size
1118   r0 = 0.4 # initial intrinsic growth rate
1119   t = seq(from = 1, to = 20, by = 1) # time
1120   q=length(t)
1121
1122   h = numeric(length = length(t)-1) # difference between two consecutive time points
1123   for(i in 1:(length(t)-1)){
1124     h[i] = t[i+1] - t[i]
1125   }
1126
1127   mu = K*(1 + ((x0/K)^(1/3) - 1)*exp((-r0*t)/3))^3 # Von Bertalanffy mean function
1128   par(mfrow = c(1,1))
1129   plot(t, mu, col = "red", lwd=2, type = "b", pch = 19,
1130        ylab = "size, X(t)", xlab = "time (t)")
1131
1132
1133   sigma5 = 0.30 # variance of X_t
1134   rho = 0.1      # corr(X_t, X_{t+1})
1135
1136   cov_mat5 = matrix(data = NA, nrow = length(t), ncol = length(t))
1137   for (i in 1:length(t)) {
1138     for (j in 1:length(t)) {
1139       cov_mat5[i,j] = sigma5*rho^(abs(i-j))
1140     }
1141   }
1142 }
1143
1144 n = 100 # number of trajectories
1145 library(mvtnorm)
1146 data = mvtnorm::rmvnorm(n, mean = mu, sigma = cov_mat5)
1147 matplot(t(data), type = "l", ylab = "Size", xlab = "Time")
1148 legend("topleft", legend = paste("n = ", n), bty = "n")
1149
1150 d1 = matrix(data = NA, nrow = n, ncol = length(t)-2)

```

```

1151 d2 = matrix(data = NA, nrow = n, ncol = length(t)-2)
1152
1153 for(j in 1:(length(t)-2)){
1154   d1[,j] = ((data[,j+1])^(1/3)) - ((data[,j])^(1/3))
1155   d2[,j] = ((data[,j+2])^(1/3)) - ((data[,j+1])^(1/3))
1156 }
1157
1158 # Computation for r_hat
1159 r_hat_vals = matrix(data = NA, nrow = n, ncol = length(t)-2)
1160 for (j in 1:(length(t)-2)) {
1161   r_hat_vals[,j] = (3/h[j])*log(d1[,j]/d2[,j])
1162 }
1163
1164 selected_index = rep(FALSE, n)
1165 for (i in 1:n) {
1166   if(sum(is.na(r_hat_vals[i,])) ==0)
1167     selected_index[i] = TRUE
1168 }
1169
1170 selected_index      # List of selected rows for further calculations
1171 index.sum5[a] = sum(selected_index)
1172 }
1173 index.sum5
1174
1175 index.sum = rbind(index.sum1/100, index.sum2/100, index.sum3/100, index.sum4/100,
1176                   index.sum5/100)
1177 colnames(index.sum) = seq(1:p)
1178 rownames(index.sum) = c(0.10, 0.15, 0.20, 0.25, 0.30)
1179 index.sum
1180 boxplot(index.sum~rownames(index.sum), col="lightgreen", ylab = "Proportion of successful simulation",
1181         xlab = expression(paste(sigma^2)))

```

1182 Similarly we can plot the efficiency boxplot for other growth models. We just have to change the mu value as
 1183 per the growth model.

1184 5. Sensitivity analysis

1185 Parameter sensitivity analysis using the ISRP of the parameters derived by localized MLE method

```

1186 fun_local_likelihood = function(r,K){
1187   sigma2 = 1
1188   rho = 0.5
1189   mu = K/(1 + (K/x0 - 1)*exp(-r*t)) #change for other growth eqn
1190
1191   cov_mat = matrix(data = NA, nrow = length(t), ncol = length(t))
1192   for (i in 1:q) {
1193     for (j in 1:q) {
1194       cov_mat[i,j] = sigma2*rho^(abs(i-j))
1195     }
1196   }
1197
1198   # localized estimates
1199   mu_local = mu[ind:(ind + 2)]

```

```

1200 data_local = data[,ind:(ind + 2)]
1201 cov_mat_local = cov_mat[ind:(ind+2), ind:(ind+2)]
1202
1203 exponent = 0
1204 for(i in 1:n){
1205     exponent = exponent + t(data_local[i,] - mu_local)%*(solve(cov_mat_local))%*(data_local[i,] - mu_local)
1206 }
1207
1208 log_lik = - (n*length(ind:(ind+2))/2)*log(2*pi) - (n/2)*log(det(cov_mat_local)) - (1/2)*exponent
1209 return(-log_lik)
1210 }
1211
1212 r = seq(0.01, 3, length = 10)
1213 K = seq(50, 150, length = 10)
1214
1215 par(mfrow = c(2,3))
1216
1217 par(mfrow = c(1,1))
1218 for (k in 1:(q-2)) {
1219     ind = k
1220     val = matrix(data = NA, nrow = length(r), ncol = length(K))
1221     for(i in 1:length(r)){
1222         for (j in 1:length(K)) {
1223             val[i,j] = fun_local_likelihood(r[i], K[j])
1224         }
1225     }
1226     if(k%%3==0){
1227         persp(r,K,val, col = "lightgrey", main = "",
1228             theta = 30, phi = 30)
1229     }
1230
1231
1232 }
1233
1234 library(rgl)
1235 open3d()
1236 mfrow3d(2, 3, sharedMouse = TRUE)
1237
1238 ind = 1
1239 val = matrix(data = NA, nrow = length(r), ncol = length(K))
1240 for(i in 1:length(r)){
1241     for (j in 1:length(K)) {
1242         val[i,j] = fun_local_likelihood(r[i], K[j])
1243     }
1244 }
1245 persp3d(r,K,val, col = "yellow", main = paste0("t=",ind),
1246     theta = 50, phi = 30)
1247
1248 ind = 4
1249 val = matrix(data = NA, nrow = length(r), ncol = length(K))
1250 for(i in 1:length(r)){
1251     for (j in 1:length(K)) {
1252         val[i,j] = fun_local_likelihood(r[i], K[j])
1253     }

```

```

1254 }
1255 persp3d(r,K,val, col = "yellow", main = paste0("t=",ind),
1256         theta = 50, phi = 30)
1257
1258 ind = 8
1259 val = matrix(data = NA, nrow = length(r), ncol = length(K))
1260 for(i in 1:length(r)){
1261     for (j in 1:length(K)) {
1262         val[i,j] = fun_local_likelihood(r[i], K[j])
1263     }
1264 }
1265 persp3d(r,K,val, col = "yellow", main = paste0("t=",ind),
1266         theta = 50, phi = 30)
1267
1268
1269 ind = 11
1270 val = matrix(data = NA, nrow = length(r), ncol = length(K))
1271 for(i in 1:length(r)){
1272     for (j in 1:length(K)) {
1273         val[i,j] = fun_local_likelihood(r[i], K[j])
1274     }
1275 }
1276 persp3d(r,K,val, col = "yellow", main = paste0("t=",ind),
1277         theta = 50, phi = 30)
1278
1279 ind = 14
1280 val = matrix(data = NA, nrow = length(r), ncol = length(K))
1281 for(i in 1:length(r)){
1282     for (j in 1:length(K)) {
1283         val[i,j] = fun_local_likelihood(r[i], K[j])
1284     }
1285 }
1286 persp3d(r,K,val, col = "yellow", main = paste0("t=",ind),
1287         theta = 50, phi = 30)
1288
1289
1290 ind = 17
1291 val = matrix(data = NA, nrow = length(r), ncol = length(K))
1292 for(i in 1:length(r)){
1293     for (j in 1:length(K)) {
1294         val[i,j] = fun_local_likelihood(r[i], K[j])
1295     }
1296 }
1297 persp3d(r,K,val, col = "yellow", main = paste0("t=",ind),
1298         theta = 50, phi = 30)

```

1299 6. Real data analysis

```

1300 REAL Data Set for this analysis:
1301 id G 0 14 28 42 56 70 84 98 112 126 133
1302 1 B 210 215 230 244 259 266 277 292 292 290 264
1303 2 B 230 240 258 277 277 293 300 323 327 340 343
1304 3 B 226 233 248 277 297 313 322 340 354 365 362
1305 4 B 233 239 253 277 292 310 318 333 336 353 338

```

1306	5	B	238	241	262	282	300	314	319	331	338	348	338
1307	6	B	225	228	237	261	271	288	300	316	319	333	330
1308	7	B	224	225	239	257	268	290	304	313	310	318	318
1309	8	B	237	241	255	276	293	307	312	336	336	344	328
1310	9	B	237	224	234	239	256	266	276	300	302	293	269
1311	10	B	233	239	259	283	294	313	320	347	348	362	352
1312	11	B	217	222	235	256	267	285	295	317	315	308	301
1313	12	B	228	223	246	266	277	287	300	312	308	328	333
1314	13	B	241	247	268	290	309	323	336	348	359	372	370
1315	14	B	221	221	240	253	273	282	292	307	306	317	318
1316	15	B	217	220	235	259	262	276	284	305	303	315	317
1317	16	B	214	221	237	256	271	283	287	314	316	320	298
1318	17	B	224	231	241	256	265	283	295	314	313	328	334
1319	18	B	200	203	221	236	248	262	276	294	291	311	310
1320	19	B	238	232	252	268	285	298	303	320	324	320	327
1321	20	B	230	222	243	253	268	284	290	316	314	330	330
1322	21	B	217	224	242	265	284	302	309	324	328	338	334
1323	22	B	209	209	221	238	256	267	281	295	301	309	289
1324	23	B	224	227	245	267	279	294	312	328	329	297	297
1325	24	B	230	231	244	261	272	283	294	318	320	333	338
1326	25	B	216	218	223	243	259	270	270	290	301	314	297
1327	26	B	231	239	254	276	294	304	317	335	333	319	307
1328	27	B	207	216	228	255	275	285	296	314	319	330	330
1329	28	B	227	236	251	264	276	287	297	315	309	313	294
1330	29	B	221	232	251	274	284	295	300	323	319	333	322
1331	30	B	233	238	254	266	282	294	295	310	320	327	326
1332	31	A	233	224	245	258	271	287	287	287	290	293	297
1333	32	A	231	238	260	273	290	300	311	313	317	321	326
1334	33	A	232	237	245	265	285	298	304	319	317	334	329
1335	34	A	239	246	268	288	308	309	327	324	327	336	341
1336	35	A	215	216	239	264	282	299	307	321	328	332	337
1337	36	A	236	226	242	255	263	277	290	299	300	308	310
1338	37	A	219	229	246	265	279	292	299	299	298	300	290
1339	38	A	231	245	270	292	302	321	322	334	323	337	337
1340	39	A	230	228	243	255	272	276	277	289	289	300	303
1341	40	A	232	240	247	263	275	286	294	302	308	319	326
1342	41	A	234	237	259	289	311	324	342	347	355	368	368
1343	42	A	237	235	258	263	282	304	318	327	336	349	353
1344	43	A	229	234	254	276	294	315	323	341	346	352	357
1345	44	A	220	227	248	273	290	308	322	326	330	342	343
1346	45	A	232	241	255	276	293	309	310	330	326	329	330
1347	46	A	210	225	242	260	272	277	273	295	292	305	306
1348	47	A	229	241	252	265	274	285	303	308	315	328	328
1349	48	A	204	198	217	233	251	258	272	283	279	295	298
1350	49	A	220	221	236	260	274	295	300	301	310	318	316
1351	50	A	233	234	250	268	280	298	308	319	318	336	333
1352	51	A	234	234	254	274	294	306	318	334	343	349	350
1353	52	A	200	207	217	238	252	267	284	282	282	284	288
1354	53	A	220	213	229	252	254	273	293	289	294	292	298
1355	54	A	225	239	254	269	289	308	313	324	327	347	344
1356	55	A	236	245	257	271	294	307	317	327	328	328	325
1357	56	A	231	231	237	261	274	285	291	301	307	315	320
1358	57	A	208	211	238	254	267	287	306	312	320	337	338
1359	58	A	232	248	261	285	292	307	312	323	318	328	329

```

1360 59 A 233 241 252 273 301 316 332 336 339 348 345
1361 60 A 221 219 231 251 270 272 287 294 292 292 299
1362
1363 setwd("D:/My Paper/3rd Paper MLE vs ISRP/RDA")
1364
1365 cattle_data_1=read.delim("cattle_data.txt",sep=" ")
1366 colnames(cattle_data_1)[1:13]<-c("id","G","0","14","28","42","56","70","84","98","112","126","133")
1367 cattle_data=cattle_data_1[,1:13]
1368
1369 data_B=cattle_data[1:30,2:12]#for Group B
1370 data_A=cattle_data[31:60,2:13]#for Group A
1371 t=c(0,14,28,42,56,70,84,98,112,126,133)
1372
1373 write.table(data_A[,-1], "data_A.txt", row.names = FALSE, col.names = FALSE)
1374
1375 t = seq(from = 1, to = 11, by = 1)
1376
1377 q=length(t)
1378 data=as.matrix(read.table("data_A.txt", header = FALSE))
1379
1380 n= 30
1381 View(data)
1382
1383 # The following parameters will be considered as the fixed set up for simulation.
1384 r= 0.05
1385 x0 = 220
1386 sigma2= 1
1387 rho= 0.1
1388
1389 mu = x0*exp(r*t)
1390 plot(t,mu)
1391
1392 fun_likelihood = function(param){
1393   r = param[1]
1394   sigma2 = param[2]
1395   rho = param[3]
1396   mu = x0*exp(r*t)
1397
1398   cov_mat = matrix(data = NA, nrow = length(t), ncol = length(t))
1399   for (i in 1:q) {
1400     for (j in 1:q) {
1401       cov_mat[i,j] = sigma2*rho^(abs(i-j))
1402     }
1403   }
1404   exponent = 0
1405   for(i in 1:n){
1406     exponent = exponent + t(data[i,]-mu)%*(solve(cov_mat))%*(data[i,]-mu)
1407   }
1408
1409   log_lik = - (n*q/2)*log(2*pi) - (n/2)*log(det(cov_mat)) - (1/2)*exponent
1410   return(-log_lik)
1411 }
1412
1413 param_0 = c(0.1, 1, 0.1)

```

```

1414 out = optim(param_0, fun_likelihoood, method = "L-BFGS-B",
1415             hessian = TRUE, lower = c(0.001, 0.1, -0.001), upper = c(1, 10, 0.999))
1416 out$par
1417 solve(out$hessian)
1418
1419 # Local MLE
1420 est_local = matrix(data = NA, nrow = q-1, ncol = 3)
1421
1422 fun_local_likelihoood = function(param){
1423   r = param[1]
1424   sigma2 = param[2]
1425   rho = param[3]
1426   mu = x0*exp(r*t)
1427
1428   cov_mat = matrix(data = NA, nrow = length(t), ncol = length(t))
1429   for (i in 1:q) {
1430     for (j in 1:q) {
1431       cov_mat[i,j] = sigma2*rho^(abs(i-j))
1432     }
1433   }
1434
1435   # localized estimates
1436   mu_local = mu[ind:(ind + 1)]
1437   data_local = data[,ind:(ind + 1)]
1438   cov_mat_local = cov_mat[ind:(ind+1), ind:(ind+1)]
1439
1440   exponent = 0
1441   for(i in 1:n){
1442     exponent = exponent + t(data_local[i,] - mu_local)%*(solve(cov_mat_local))%*(data_local[i,] - mu_local)
1443   }
1444
1445   log_lik = - (n*length(ind:(ind+1))/2)*log(2*pi) - (n/2)*log(det(cov_mat_local)) - (1/2)*exponent
1446   return(-log_lik)
1447 }
1448
1449
1450 # likelihood function
1451 for(j in 1:(q-1)){
1452   ind = j
1453   param_0 = c(0.1, 1, 0.1)
1454   out = optim(param_0, fun_local_likelihoood, method = "L-BFGS-B",
1455             hessian = TRUE, lower = c(0.001, 0.1, -0.001), upper = c(1, 10, 0.999))
1456   est_local[j, ] = out$par
1457 }
1458 colnames(est_local) = c("r", "sigma2", "rho")
1459
1460 par(mar = c(5.1, 5.1, 4.1, 2.1))
1461 plot(1:(q-1), est_local[,1], type = "b", col = "red", lwd = 2,
1462      ylab = expression(widehat(r(Delta~t))[MLE]),
1463      xlab = "Time")
1464 r_hat=est_local[,1]
1465 x = 1:10
1466 fit = nls(r_hat ~ b*exp(-c*x)*x, start = list(b=1,c=3))
1467

```

```

1468 points(x,r_hat, col = "red", lwd = 2, cex= 1.5, pch = 19)
1469
1470 curve(coef(fit)[1]*exp(-coef(fit)[2]*x)*x, lwd = 3, col = "blue", add = TRUE)
1471
1472 legend("bottomright", legend = bquote(r(t) == b*e^(-c*t)*t),
1473       bty = "n", lwd = 3, col = "blue")
1474
1475 b_hat = coef(fit)[1]
1476 c_hat = coef(fit)[2]
1477 AIC(fit)
1478

```