https://johoe.kmu.ac.ir
doi:10.34172/.johoe.2509.1793
JOHOE. 2026;15:2509.1793-S2

Table S1: Average Bias of Individual Prevalence, GEE Estimator, Overall Prevalence, and REM Estimator for Untreated Dental Caries across Varying Sample Sizes and ICC Values

Prevalence	ICC	Sample size	Average Individual Prevalence	GEE estimator	Overall Prevalence	REM estimator
0.05	**0.05**	100	0.001	0.001	0.001	-0.015
		200	0.000	0.000	0.000	-0.016
		300	0.000	0.000	0.000	-0.016
	0.1	100	0.002	0.001	0.001	-0.026
		200	0.001	0.001	0.001	-0.027
		300	0.001	0.001	0.001	-0.027
	0.2	100	-0.001	-0.001	-0.001	-0.041
		200	-0.001	-0.001	-0.001	-0.042
		300	-0.001	-0.001	-0.001	-0.042
0.1	**0.05**	100	-0.001	-0.001	-0.001	-0.018
		200	-0.001	-0.001	-0.001	-0.018
		300	-0.001	-0.001	-0.001	-0.018
	0.1	100	0.000	0.000	0.001	-0.032
		200	0.000	0.000	0.001	-0.032
		300	0.000	0.000	0.001	-0.032
	0.2	100	0.000	0.000	0.000	-0.058
		200	0.000	0.000	0.000	-0.058
		300	0.000	0.000	0.000	-0.058
0.2	**0.05**	100	-0.001	-0.001	-0.001	-0.015
		200	-0.001	-0.001	-0.001	-0.016
		300	-0.001	0.000	0.000	-0.015
	0.1	100	-0.001	-0.001	-0.001	-0.029
		200	-0.002	-0.002	-0.001	-0.030
		300	-0.002	-0.002	-0.001	-0.030
	0.2	100	0.000	0.000	0.000	-0.059
		200	0.000	0.000	0.000	-0.060
		300	0.000	0.000	-0.001	-0.060
5%+1%×number of missing teeth for each individual	**0.05**	100	-0.009	-0.005	0.000	-0.019
		200	-0.009	-0.005	0.000	-0.019
		300	-0.009	-0.005	0.000	-0.019
	0.1	100	-0.010	-0.007	-0.001	-0.032
		200	-0.010	-0.007	-0.001	-0.033
		300	-0.009	-0.006	0.000	-0.032

	0.2	100	-0.007	-0.005	0.002	-0.059
		200	-0.007	-0.005	0.002	-0.060
		300	-0.007	-0.006	0.001	-0.060

Table S2: Average Confidence Interval length for Individual Prevalence, GEE Estimator, Overall Prevalence, and REM Estimator of Untreated Dental Caries Across Different Sample Sizes and ICC Values.

Prevalence	ICC	Sample size	Average Individual Prevalence	GEE estimator	Overall Prevalence	REM estimator
0.05	**0.05**	100	0.026	0.026	0.019	0.024
		200	0.019	0.019	0.013	0.017
		300	0.015	0.015	0.011	0.014
	0.1	100	0.033	0.033	0.019	0.024
		200	0.023	0.023	0.013	0.016
		300	0.019	0.019	0.011	0.013
	0.2	100	0.040	0.041	0.018	0.017
		200	0.029	0.029	0.013	0.011
		300	0.024	0.024	0.011	0.008
0.1	**0.05**	100	0.036	0.036	0.025	0.028
		200	0.025	0.025	0.018	0.022
		300	0.021	0.021	0.015	0.018
	0.1	100	0.044	0.044	0.026	0.038
		200	0.031	0.031	0.018	0.027
		300	0.025	0.025	0.015	0.022
	0.2	100	0.056	0.057	0.025	0.038
		200	0.040	0.040	0.018	0.027
		300	0.033	0.033	0.015	0.022
0.2	**0.05**	100	0.048	0.048	0.034	0.049
		200	0.034	0.034	0.024	0.034
		300	0.028	0.028	0.020	0.028
	0.1	100	0.058	0.057	0.034	0.058
		200	0.041	0.041	0.024	0.041
		300	0.034	0.033	0.020	0.034
	0.2	100	0.075	0.075	0.034	0.063
		200	0.053	0.053	0.024	0.050
		300	0.044	0.043	0.020	0.041
5%+1%×number of missing teeth for each individual	**0.05**	100	0.056	0.056	0.038	0.058
		200	0.039	0.039	0.027	0.041
		300	0.032	0.032	0.022	0.033
	0.1	100	0.065	0.065	0.038	0.069
		200	0.047	0.046	0.027	0.049

	300	0.038	0.038	0.022	0.040
0.2	100	0.084	0.084	0.038	0.090
	200	0.060	0.060	0.027	0.064
	300	0.049	0.049	0.022	0.052

Simulation syntax for Untreated dental caries Independent of missing teeth scenarios

```
library(VGAM)

library(geepack)

library(lme4)

library(boot)

library(doParallel)

library(foreach)



}sim<-function(n,p, rho)


}al<-function(p)

exp(p)/(1+exp(p))

{


Fit2#############################################################################


Function to fit the model and capture all warnings #

} ()capture_warnings2 <- function

Initialize a vector to store warning messages #

()warning_messages <- character


Custom warning handler to append warnings to the vector #

} warning_handler <- function(warn)

warning_messages <<- c(warning_messages, warn$message)
```

```
invokeRestart("muffleWarning")  # Prevent the warning from being printed
{

Try fitting the model and capture warnings #
})fit <- tryCatch
})withCallingHandlers
fit2 <- geeglm(D ~ 1, id = ID, data = datalong, family = binomial(link = "logit"), corstr = "exchangeable")
(warning = warning_handler ,{
fit
} (error = function(e ,{
Handle errors if necessary #
NULL
({

Return the vector of warning messages #
return(list(fit = fit2, warnings = warning_messages))
{




Fit3##############################################################################

Function to fit the model and capture all warnings #
} ()capture_warnings3 <- function
Initialize a vector to store warning messages #
()warning_messages <- character

Custom warning handler to append warnings to the vector #
```

```
} warning_handler <- function(warn)
warning_messages <<- c(warning_messages, warn$message)
invokeRestart("muffleWarning")  # Prevent the warning from being printed
{

Try fitting the model and capture warnings #
})fit <- tryCatch
})withCallingHandlers
fit3 <- glmer(D ~ 1 + (1 | ID), data = datalong, family = binomial(link = "logit"))
(warning = warning_handler ,{
fit
} (error = function(e ,{
Handle errors if necessary #
NULL
({

Return the vector of warning messages #
return(list(fit = fit3, warnings = warning_messages))
{



} statistic_function <- function(data, indices)
Subset the data #
sample_data <- data[indices, ]

Calculate D/NT for each row #
ratios <- sample_data$DT / sample_data$NT

Return the mean of the ratios #
```

```
return(mean(ratios))
{

sample_size=n

(1235)set.seed
ID=1:5000
NT=sample(14:28,5000,replace=TRUE)
DT=rbetabinom(5000,NT,p,rho)
pop=data.frame(ID,NT,DT)
sum(DT)/sum(NT)
()results=data.frame
}for(j in 1:1000)

random_sample <- pop[sample(nrow(pop), sample_size), ]
datashort=random_sample

()datalong=data.frame
Loop through each row of the random_sample data frame #
} for (i in 1:nrow(random_sample))
Get the current ID, NT, and DT values #
current_id <- random_sample$ID[i]
current_nt <- random_sample$NT[i]
current_dt <- random_sample$DT[i]

Create a vector of zeros and ones #
zeros <- rep(0, current_nt - current_dt)
ones <- rep(1, current_dt)
```

```
Combine zeros and ones #
D <- c(zeros, ones)

Create a temporary data frame for the current ID #
temp_df <- data.frame(ID = rep(current_id, current_nt), D = D)

Append the temporary data frame to the long_format data frame #
datalong <- rbind(datalong , temp_df)
{

()result2 <- capture_warnings2
Call the function and get the warning messages #
()warnings_vector2=character
fit2 <- result2$fit
warnings_vector2 <- result2$warnings

}if(length(warnings_vector2)>0)
convergence2=max( grepl("convergence not obtained",warnings_vector2))
{else{convergence2=0 {

()result3 <- capture_warnings3
Call the function and get the warning messages #
()warnings_vector3=character
fit3 <- result3$fit
warnings_vector3 <- result3$warnings
}if(length(warnings_vector3)>0)
convergence3=max( grepl("Model failed to converge",warnings_vector3))
{else{convergence3=0 {
```

```
bootstrap_results <- boot(data = datashort, statistic = statistic_function, R = 1000)
bootstrap_ci<- boot.ci(bootstrap_results, type = "perc")

pfit1=sum(datashort$DT)/sum(datashort$NT)
clU1=pfit1+1.96*sqrt(pfit1*(1-pfit1)/sum(datashort$NT))
clL1=pfit1-1.96*sqrt(pfit1*(1-pfit1)/sum(datashort$NT))

pfit2<- al(unname(coef(fit2)[1]))
clU2<- al(unname(coef(fit2)[1])+1.96*unname(sqrt(diag(vcov(fit2)))[1]))
clL2<-al(unname(coef(fit2)[1])-1.96*unname(sqrt(diag(vcov(fit2)))[1]))

pfit3<- al(unname(fixef(fit3)[1]))
clU3<- al(unname(fixef(fit3)[1])+1.96*unname(sqrt(diag(vcov(fit3)))[1]))
clL3<- al(unname(fixef(fit3)[1])-1.96*unname(sqrt(diag(vcov(fit3)))[1]))

pfit4=mean(datashort$DT/datashort$NT)
a=unlist(bootstrap_ci)
clU4=a$percent5
clL4=a$percent4


)results <- rbind(results,data.frame
,Names = c("Overal Prevalence","GEE estimate","REM Estimate","Average Individual Prevalence" )
,Estimate = c(pfit1, pfit2, pfit3, pfit4)
,Lower_CI = c(clL1, clL2, clL3, clL4)
((Upper_CI = c(clU1, clU2, clU3, clU4)

{
```

```
return(results)
{


library(parallel)
library(foreach)
library(doParallel)

Detect the number of cores and create a cluster #
numCores <- detectCores() - 1  # Use one less than the total number of cores
cl <- makeCluster(numCores)
registerDoParallel(cl)

Load necessary packages #
packages = c("VGAM", "geepack", "lme4", "boot", "dplyr", "tidyverse")
lapply(packages, require, character.only = TRUE)

Define parameters #
(0.2 ,0.05,0.1)prevalence = c
(0.2 ,0.1 ,0.05)ICC = c
(300)sample_sizes = c

Outer loop: prevalence #
%:% results <- foreach(n = sample_sizes, .combine = 'rbind', .packages = packages)
%:%foreach(p = prevalence , .combine = 'rbind')
} %foreach(rho = ICC, .combine = 'rbind') %dopar
Run the simulation #
result <- sim(n,p, rho)
```

```
Create a filename for each result #
filename <- paste0("popv_",p,"_", n, "_", rho, ".csv")
write.csv(result, filename, row.names = FALSE)  # Write result to CSV

return(result)  # Return the result if needed
{

Stop the cluster #
stopCluster(cl)
```

Simulation syntax for Untreated dental caries Independent of missing teeth scenarios

```
library(VGAM)
library(geepack)
library(lme4)
library(boot)
library(doParallel)
library(foreach)



}sim<-function(p, n, rho)

}al<-function(p)
exp(p)/(1+exp(p))
{

Fit2#############################################################################

Function to fit the model and capture all warnings #
} ()capture_warnings2 <- function
```

```
Initialize a vector to store warning messages #
()warning_messages <- character

Custom warning handler to append warnings to the vector #
} warning_handler <- function(warn)
warning_messages <<- c(warning_messages, warn$message)
invokeRestart("muffleWarning")  # Prevent the warning from being printed
{

Try fitting the model and capture warnings #
})fit <- tryCatch
})withCallingHandlers
fit2 <- geeglm(D ~ 1, id = ID, data = datalong, family = binomial(link = "logit"), corstr = "exchangeable")
(warning = warning_handler ,{
fit
} (error = function(e ,{
Handle errors if necessary #
NULL
({

Return the vector of warning messages #
return(list(fit = fit2, warnings = warning_messages))
{




Fit3##############################################################################
```

```
Function to fit the model and capture all warnings #
} ()capture_warnings3 <- function
Initialize a vector to store warning messages #
()warning_messages <- character

Custom warning handler to append warnings to the vector #
} warning_handler <- function(warn)
warning_messages <<- c(warning_messages, warn$message)
invokeRestart("muffleWarning")  # Prevent the warning from being printed
{

Try fitting the model and capture warnings #
})fit <- tryCatch
})withCallingHandlers
fit3 <- glmer(D ~ 1 + (1 | ID), data = datalong, family = binomial(link = "logit"))
(warning = warning_handler ,{
fit
} (error = function(e ,{
Handle errors if necessary #
NULL
({

Return the vector of warning messages #
return(list(fit = fit3, warnings = warning_messages))
{



} statistic_function <- function(data, indices)
Subset the data #
```

```
sample_data <- data[indices, ]

Calculate D/NT for each row #
ratios <- sample_data$DT / sample_data$NT

Return the mean of the ratios #
return(mean(ratios))
{

(1235)set.seed
sample_size=n
ID=1:5000
NT=sample(14:28,5000,replace=TRUE)
p=0.05+0.01*NT
DT=rbetabinom(5000,NT,p,rho)
pop=data.frame(ID,NT,DT)
sum(DT)/sum(NT)
()results=data.frame
}for(i in 1:1000)

random_sample <- pop[sample(nrow(pop), sample_size), ]
datashort=random_sample

()datalong=data.frame
Loop through each row of the random_sample data frame #
} for (i in 1:nrow(random_sample))
Get the current ID, NT, and DT values #
current_id <- random_sample$ID[i]
current_nt <- random_sample$NT[i]
```

```
current_dt <- random_sample$DT[i]

Create a vector of zeros and ones #
zeros <- rep(0, current_nt - current_dt)
ones <- rep(1, current_dt)

Combine zeros and ones #
D <- c(zeros, ones)

Create a temporary data frame for the current ID #
temp_df <- data.frame(ID = rep(current_id, current_nt), D = D)

Append the temporary data frame to the long_format data frame #
datalong <- rbind(datalong , temp_df)
{

()result2 <- capture_warnings2
Call the function and get the warning messages #
()warnings_vector2=character
fit2 <- result2$fit
warnings_vector2 <- result2$warnings

}if(length(warnings_vector2)>0)
convergence2=max( grepl("convergence not obtained",warnings_vector2))
{else{convergence2=0 {

()result3 <- capture_warnings3
Call the function and get the warning messages #
()warnings_vector3=character
```

```
fit3 <- result3$fit
warnings_vector3 <- result3$warnings
}if(length(warnings_vector3)>0)
convergence3=max( grepl("Model failed to converge",warnings_vector3))
{else{convergence3=0 {

bootstrap_results <- boot(data = datashort, statistic = statistic_function, R = 1000)
bootstrap_ci<- boot.ci(bootstrap_results, type = "perc")

pfit1=sum(datashort$DT)/sum(datashort$NT)
clU1=pfit1+1.96*sqrt(pfit1*(1-pfit1)/sum(datashort$NT))
clL1=pfit1-1.96*sqrt(pfit1*(1-pfit1)/sum(datashort$NT))

pfit2<- al(unname(coef(fit2)[1]))
clU2<- al(unname(coef(fit2)[1])+1.96*unname(sqrt(diag(vcov(fit2)))[1]))
clL2<-al(unname(coef(fit2)[1])-1.96*unname(sqrt(diag(vcov(fit2)))[1]))

pfit3<- al(unname(fixef(fit3)[1]))
clU3<- al(unname(fixef(fit3)[1])+1.96*unname(sqrt(diag(vcov(fit3)))[1]))
clL3<- al(unname(fixef(fit3)[1])-1.96*unname(sqrt(diag(vcov(fit3)))[1]))

pfit4=mean(datashort$DT/datashort$NT)
a=unlist(bootstrap_ci)
clU4=a$percent5
clL4=a$percent4


)results <- rbind(results,data.frame
,Names = c("Overal Prevalence","GEE estimate","REM Estimate","Average Individual Prevalence" )
```

```
,Estimate = c(pfit1, pfit2, pfit3, pfit4)
,Lower_CI = c(clL1, clL2, clL3, clL4)
((Upper_CI = c(clU1, clU2, clU3, clU4)

{

return(results)
{



library(parallel)
library(foreach)
library(doParallel)

Detect the number of cores and create a cluster #
numCores <- detectCores() - 1  # Use one less than the total number of cores
cl <- makeCluster(numCores)
registerDoParallel(cl)

Load necessary packages #
packages = c("VGAM", "geepack", "lme4", "boot", "dplyr", "tidyverse")
lapply(packages, require, character.only = TRUE)

Define parameters #
(0.2 ,0.1 ,0.05)ICC = c
(300 ,200 ,100)sample_sizes = c

Outer loop: prevalence #
%:% results <- foreach(p = prevalence, .combine = 'rbind', .packages = packages)
```

```
%:% foreach(n = sample_sizes, .combine = 'rbind')
} %foreach(rho = ICC, .combine = 'rbind') %dopar
Run the simulation #
result <- sim(p, n, rho)

Create a filename for each result #
filename <- paste0("pop_", p, "_", n, "_", rho, ".csv")
write.csv(result, filename, row.names = FALSE)  # Write result to CSV

return(result)  # Return the result if needed
{

Stop the cluster #
stopCluster(cl)
```