library(tidyverse)  
library(kableExtra)  
library(geepack)  
# States info that were collected in 2017-18   
state2017 <- c("Colorado", "Connecticut", "Minnesota", "Montana",   
 "New Jersey", "New York", "North Dakota",   
 "Pennsylvania", "South Dakota", "Utah", "Washington")  
  
# State abbreviations mapping  
state\_abbreviations <- c(  
 "Alabama"="AL", "Alaska"="AK", "Arizona"="AZ", "Arkansas"="AR",   
 "California"="CA", "Colorado"="CO", "Connecticut"="CT", "Delaware"="DE",   
 "Florida"="FL", "Georgia"="GA", "Hawaii"="HI", "Idaho"="ID", "Illinois"="IL", "Indiana"="IN", "Iowa"="IA", "Kansas"="KS",   
 "Kentucky"="KY", "Louisiana"="LA", "Maine"="ME", "Maryland"="MD",   
 "Massachusetts"="MA", "Michigan"="MI", "Minnesota"="MN",   
 "Mississippi"="MS", "Missouri"="MO", "Montana"="MT",   
 "Nebraska"="NE", "Nevada"="NV", "New Hampshire"="NH", "New Jersey"="NJ",  
 "New Mexico"="NM", "New York"="NY", "North Carolina"="NC",   
 "North Dakota"="ND", "Ohio"="OH", "Oklahoma"="OK", "Oregon"="OR",  
 "Pennsylvania"="PA", "Rhode Island"="RI", "South Carolina"="SC",  
 "South Dakota"="SD", "Tennessee"="TN", "Texas"="TX", "Utah"="UT",   
 "Vermont"="VT","Virginia"="VA", "Washington"="WA", "West Virginia"="WV",  
 "Wisconsin"="WI", "Wyoming"="WY"  
)  
  
vaccine <- read.csv("vaccine.csv") %>%   
 mutate(type =  
 case\_when(type == ""~"Unknown",   
 type %in% c("BOCES", "Kindergarten", "Nonpublic") ~ "Others",  
 TRUE ~ type),   
 mmr = ifelse(mmr == -1, NA, mmr),   
 overall = ifelse(overall == -1, NA, overall),   
 per\_capita = statespending2016/schagepop2016,   
 year = case\_when(state %in% state2017 ~ "2017-18",   
 TRUE ~ "2018-19"),  
 county = ifelse(county == "", NA, county),   
 city = ifelse(city == "", NA, city),  
 state\_abbr = state\_abbreviations[state])

obsID <- 1:nrow(vaccine)  
vac <- vaccine %>% mutate(oID = obsID) %>% relocate(oID, .before =state)  
duplicate\_df <- vac %>%   
 group\_by(state, county, city, name) %>%   
 reframe(oID = oID, type = type, year = year, n= n(),   
 enroll = enroll, mmr = mmr, overall = overall) %>%   
 filter(n > 1) %>%   
 group\_by(state, county, city, name) %>%   
 mutate(record = row\_number()) %>% ungroup()  
  
# Calculate average of duplicated data   
avg\_duplicates <- duplicate\_df %>%  
 group\_by(state, county, city, name) %>%   
 mutate(mean\_enroll = mean(enroll), mean\_mmr = mean(mmr),   
 mean\_overall = mean(overall)) %>%   
 select(-c(mmr, overall, enroll, n)) %>%   
 rename(mmr = mean\_mmr, overall=mean\_overall, enroll = mean\_enroll)  
  
remove\_sid <- avg\_duplicates %>% filter(record != 1) %>% pull(oID)  
  
data <- vac %>% filter(!(oID %in% remove\_sid)) %>%   
 # Add the total number of schools in each state   
 left\_join(vac %>% group\_by(state) %>%   
 summarise(n\_school = n\_distinct(name)), by = "state")  
  
# Separate Data  
MMR <- data %>% filter(!is.na(mmr))   
nStates <- n\_distinct(MMR$state)  
# Create state ID   
MMR <- MMR %>% left\_join(data.frame(state = unique(MMR$state),   
 stateID = 1:nStates))  
  
overall <- data %>% filter(!is.na(overall))   
nStates <- n\_distinct(overall$state)  
overall <- overall %>% left\_join(data.frame(state = unique(MMR$state),   
 stateID = 1:nStates))

# =============== Stratified Sampling ==================== #   
# MMR  
# Step 1: Calculate weights for each state based on the number of schools  
state\_weights <- mmr\_dat %>%  
 count(state) %>%  
 mutate(weight = n / sum(n))  
  
# Step 2: Sample schools within each state, proportional to state weights  
# Set a total sample size of N schools  
N <- 3000 # Total desired sample size  
  
# Calculate the number of schools to sample from each state, based on weights  
state\_weights <- state\_weights %>%  
 mutate(sample\_size = round(weight \* N))  
  
set.seed(123)  
  
mmr\_samples <- NULL  
for (i in 1:nrow(state\_weights)) {  
 state <- state\_weights[i, ]$state  
 size <- state\_weights$sample\_size[i]  
 sample <- mmr\_dat %>% filter(state == state) %>%  
 sample\_n(size, replace = FALSE)  
 mmr\_samples <- rbind(mmr\_samples, sample)  
}  
  
mmr\_samples <- mmr\_samples %>% group\_by(state) %>%  
 mutate(schoolID = row\_number()) %>%  
 relocate(schoolID, .after = stateID) %>%  
 ungroup() %>%  
 arrange(stateID)  
  
# Overall Vaccination Rate  
# Step 1: Calculate weights for each state based on the number of schools  
state\_weights <- overall\_dat %>%  
 count(state) %>%  
 mutate(weight = n / sum(n))  
  
# Step 2: Sample schools within each state, proportional to state weights  
# Set a total sample size of N schools  
N <- 3000 # Total desired sample size  
  
# Calculate the number of schools to sample from each state, based on weights  
state\_weights <- state\_weights %>%  
 mutate(sample\_size = round(weight \* N))  
  
set.seed(10049)  
overall\_samples <- NULL  
for (i in 1:nrow(state\_weights)) {  
 state <- state\_weights[i, ]$state  
 size <- state\_weights$sample\_size[i]  
 sample <- overall\_dat %>% filter(state == state) %>%   
 sample\_n(size, replace = FALSE)  
 overall\_samples <- rbind(overall\_samples, sample)  
}  
  
overall\_samples <- overall\_samples %>% group\_by(state) %>%   
 mutate(schoolID = row\_number()) %>%   
 relocate(schoolID, .after = stateID) %>%   
 ungroup() %>%   
 arrange(stateID)  
  
# ===================== GEE ========================== #   
# 1. MMR with Unadjusted   
# GEE with Exchangeable Structure  
mmr\_ex <- geeglm(mmr ~ per\_capita, family = gaussian,   
 data = mmr\_samples, id = stateID,   
 corstr = "exchangeable")  
  
# GEE with AR1 Structure  
mmr\_ar1 <- geeglm(mmr ~ per\_capita, family = gaussian,   
 data = mmr\_samples, id = stateID,   
 corstr = "ar1")  
  
# GEE with Independent Structure  
mmr\_ind <- geeglm(mmr ~ per\_capita, family = gaussian,   
 data = mmr\_samples, id = stateID,   
 corstr = "independence")  
  
# Model comparison   
anova(mmr\_ex, mmr\_ar1)  
anova(mmr\_ex, mmr\_ind)  
anova(mmr\_ind, mmr\_ar1)  
  
# QIC Measures of Fit   
MuMIn::model.sel(mmr\_ex, mmr\_ar1, mmr\_ind, rank="QIC")  
  
# 2. MMR with adjusted   
# Adjusted GEE with exchangeable   
mmr\_adj\_ex <- geeglm(mmr ~ per\_capita + n\_school + type,   
 family = gaussian,   
 data = mmr\_samples,   
 id = stateID,   
 corstr = "exchangeable")  
  
# Adjusted GEE with AR1   
mmr\_adj\_ar1 <- geeglm(mmr ~ per\_capita + n\_school + type,   
 family = gaussian,   
 data = mmr\_samples,   
 id = stateID,   
 corstr = "ar1")  
  
mmr\_adj\_ind <- geeglm(mmr ~ per\_capita + n\_school + type,   
 family = gaussian,   
 data = mmr\_samples,   
 id = stateID,   
 corstr = "independence")  
  
anova(mmr\_adj\_ex, mmr\_adj\_ar1)  
anova(mmr\_adj\_ex, mmr\_adj\_ind)  
anova(mmr\_adj\_ar1, mmr\_adj\_ind)  
  
# QIC Measures of Fit   
MuMIn::model.sel(mmr\_adj\_ex, mmr\_adj\_ar1, mmr\_adj\_ind, rank="QIC")  
  
# 3. Overall with unadjusted   
# Exchangeable  
unadj\_overall\_ex <- geeglm(overall ~ per\_capita,   
 family = gaussian,   
 data = overall\_samples,   
 id = stateID,   
 corstr = "exchangeable")  
  
# AR1  
unadj\_overall\_ind <- geeglm(overall ~ per\_capita,   
 family = gaussian,   
 data = overall\_samples,   
 id = stateID,   
 corstr = "independence")  
  
# Fit the GEE model  
unadj\_overall\_ar1 <- geeglm(overall ~ per\_capita,   
 family = gaussian,   
 data = overall\_samples,   
 id = stateID,   
 corstr = "ar1")  
  
anova(unadj\_overall\_ex, unadj\_overall\_ar1)  
anova(unadj\_overall\_ex, unadj\_overall\_ind)  
anova(unadj\_overall\_ar1, unadj\_overall\_ind)  
MuMIn::model.sel(unadj\_overall\_ex, unadj\_overall\_ind,   
 unadj\_overall\_ar1, rank="QIC")  
  
# 4. Overall with adjusted   
# Exchangeable  
adj\_overall\_ex <- geeglm(overall ~ per\_capita + n\_school + type,   
 family = gaussian,   
 data = overall\_samples,   
 id = stateID,   
 corstr = "exchangeable")  
  
# Independence  
adj\_overall\_ind <- geeglm(overall ~ per\_capita + n\_school + type,   
 family = gaussian,   
 data = overall\_samples,   
 id = stateID,   
 corstr = "independence")  
  
# AR1  
adj\_overall\_ar1 <- geeglm(overall ~ per\_capita + n\_school + type,   
 family = gaussian,   
 data = overall\_samples,   
 id = stateID,   
 corstr = "ar1")  
  
anova(adj\_overall\_ex, adj\_overall\_ind)  
anova(adj\_overall\_ex, adj\_overall\_ar1)  
anova(adj\_overall\_ar1, adj\_overall\_ind)  
MuMIn::model.sel(adj\_overall\_ex, adj\_overall\_ind, adj\_overall\_ar1, rank="QIC")