### setup. R

# Load required packages

if (!require("pacman", character.only = TRUE)) {

install.packages("pacman")

}

pacman::p\_load(DiagrammeR, greta, dplyr, tidyr, binom, matrixStats, ggplot2, bayesplot,viridis, patchwork,ggridges)

### functions.R

# please supply a good name for this function

'%notin%' <- function(x, table) {

!(x %in% table)

}

# Functions to convert between odds and probability

odds\_to\_probability <- function(odds) {

# Calculate probability from given odds

odds / (1 + odds)

}

probability\_to\_odds <- function(probability) {

# Calculate odds from given probability

probability / (1 - probability)

}

# Function to calculate weighted averages of predictions across months

get\_weight\_month\_pred <- function(fit\_array, pop\_survey\_data) {

# Determine the number of months and predictions

n\_month <- dim(fit\_array)[3]

n\_site <- dim(fit\_array)[2]

n\_preds <- dim(fit\_array)[1]

# Initialize an array for storing weighted averages

weighted\_averages <- array(dim = c(n\_month, n\_preds))

# Create a grid of all possible site-month combinations

full\_grid <- expand.grid(site = 1:n\_site, month = 1:n\_month)

# Merge this grid with population survey data to include sample sizes

# Replace missing values with 0, indicating no samples

sample\_sizes\_df <- full\_grid %>%

left\_join(pop\_survey\_data, by = c("site", "month")) %>%

replace\_na(list(sample\_size = 0)) %>%

arrange(site, month)

# Convert the merged data into a matrix format for easier processing

sample\_sizes\_matrix <- matrix(sample\_sizes\_df$sample\_size, nrow = n\_site, byrow = TRUE)

# Loop through each prediction and month to calculate weighted averages

for (pred in 1:n\_preds) {

for (month in 1:n\_month) {

predictions <- fit\_array[pred, , month] # Extract predictions for all sites

weights <- sample\_sizes\_matrix[, month] # Extract corresponding weights

# Calculate the weighted average and store it

weighted\_averages[month, pred] <- sum(predictions \* weights) / sum(weights)

}

}

return(weighted\_averages)

}

# Function to calculate weighted averages of predictions across sites

get\_weight\_site\_pred <- function(fit\_array, pop\_survey\_data) {

# The dimensions for n\_month should be corrected to n\_site for consistency in variable naming

n\_month <- dim(fit\_array)[3]

n\_site <- dim(fit\_array)[2]

n\_preds <- dim(fit\_array)[1]

# Initialize an array for storing weighted averages

weighted\_averages <- array(dim = c(n\_site, n\_preds))

# Create a grid of all possible site-month combinations

full\_grid <- expand.grid(site = 1:n\_site, month = 1:n\_month)

# Merge this grid with population survey data to include sample sizes

sample\_sizes\_df <- full\_grid %>%

left\_join(pop\_survey\_data, by = c("site", "month")) %>%

replace\_na(list(sample\_size = 0)) %>%

arrange(site, month)

# Convert the dataframe to a matrix of sample sizes

sample\_sizes\_matrix <- matrix(sample\_sizes\_df$sample\_size, nrow = n\_site, byrow = TRUE)

# Loop through each prediction and site to calculate weighted averages

for (pred in 1:n\_preds) {

for (site in 1:n\_site) {

predictions <- fit\_array[pred, site, ] # Extract predictions for all months

weights <- sample\_sizes\_matrix[site, ] # Extract corresponding weights

# Calculate the weighted average and store it

weighted\_averages[site, pred] <- sum(predictions \* weights) / sum(weights)

}

}

return(weighted\_averages)

}

### data\_generation.R

generate\_data\_m1<-function(n\_month=50,

n\_sites=20,

base\_par=0,

month\_sd=2,

site\_sd =1,

ANC\_log\_odds\_ratio=-0.5,

survey\_samples\_site\_month\_min=20,

survey\_samples\_site\_month\_max=100,

ANC\_samples\_site\_month\_min=10,

ANC\_samples\_site\_month\_max=50

){

# generate prevalence and site-specific params

log\_odd\_ratio\_month <- rnorm(n\_month, 0, month\_sd) # Odds ratios for each site

log\_odd\_ratio\_site <- rnorm(n\_sites, 0, site\_sd) # Odds ratios for each site

# Initialize an empty dataframes for survey and ANC

survey\_data <- data.frame(month = integer(), site = integer(), sample\_size = integer(), positive = integer())

ANC\_data<- data.frame(month = integer(), site = integer(), sample\_size = integer(), positive = integer())

# Loop over sites and months

for (site in 1:n\_sites) {

for (month in 1:n\_month) {

survey\_sample\_size <- sample(survey\_samples\_site\_month\_min:survey\_samples\_site\_month\_max, 1)

ANC\_sample\_size <- sample(ANC\_samples\_site\_month\_min:ANC\_samples\_site\_month\_max, 1)

odds\_survey<-exp(base\_par+log\_odd\_ratio\_month[month]+log\_odd\_ratio\_site[site])

survey\_prevalence\_s\_m<-odds\_to\_probability(odds\_survey)

odds\_ANC<-exp(base\_par+log\_odd\_ratio\_month[month]+log\_odd\_ratio\_site[site]+ANC\_log\_odds\_ratio)

ANC\_prev\_s\_m<-odds\_to\_probability(odds\_ANC)

survey\_positive <- rbinom(1, size = survey\_sample\_size, prob = survey\_prevalence\_s\_m)

ANC\_samples<-rbinom(ANC\_sample\_size,1,ANC\_prev\_s\_m)

survey\_data <- rbind(survey\_data, data.frame(month=month, site=site,sample\_size=survey\_sample\_size,positive=survey\_positive))

ANC\_data <- rbind(ANC\_data, data.frame(month=month, site=site, sample\_size=1, positive=ANC\_samples))

}

}

ANC\_data$ANC=1

survey\_data$ANC=0

data\_to\_model<-rbind(ANC\_data, survey\_data)

return(list(

param\_df=data.frame(

base\_par=base\_par,

preg\_par=ANC\_log\_odds\_ratio,

tau\_month=month\_sd,

tau\_site=site\_sd),

simulated\_survey\_data=survey\_data,

data\_to\_model=data\_to\_model

))

}

generate\_data\_m1\_runif\_site<-function(n\_month=50,

n\_sites=20,

month\_sd =2,

ANC\_log\_odds\_ratio=-0.5,

survey\_samples\_site\_month\_min=20,

survey\_samples\_site\_month\_max=100,

ANC\_samples\_site\_month\_min=10,

ANC\_samples\_site\_month\_max=50,

pred\_sites=c(15:20)

){

# generate prevalence and site-specific params

survey\_prev <- runif(n\_sites) # Random uniform probabilities for each site

log\_odd\_ratio\_month <- rnorm(n\_month, 0, month\_sd) # Odds ratios for each site

# Initialize an empty dataframes for survey and ANC

survey\_data <- data.frame(month = integer(), site = integer(), sample\_size = integer(), positive = integer())

ANC\_data<- data.frame(month = integer(), site = integer(), sample\_size = integer(), positive = integer())

# Loop over sites and months

for (site in 1:n\_sites) {

for (month in 1:n\_month) {

survey\_sample\_size <- sample(survey\_samples\_site\_month\_min:survey\_samples\_site\_month\_max, 1)

ANC\_sample\_size <- sample(ANC\_samples\_site\_month\_min:ANC\_samples\_site\_month\_max, 1)

survey\_prevalence\_s\_m<-odds\_to\_probability(probability\_to\_odds(survey\_prev[site])\*exp(log\_odd\_ratio\_month[month]))

ANC\_prev\_s\_m<-odds\_to\_probability(probability\_to\_odds(survey\_prevalence\_s\_m)\*exp(ANC\_log\_odds\_ratio))

survey\_positive <- rbinom(1, size = survey\_sample\_size, prob = survey\_prevalence\_s\_m)

ANC\_samples<-rbinom(ANC\_sample\_size,1,ANC\_prev\_s\_m)

survey\_data <- rbind(survey\_data, data.frame(month=month, site=site,sample\_size=survey\_sample\_size,positive=survey\_positive))

ANC\_data <- rbind(ANC\_data, data.frame(month=month, site=site, sample\_size=1, positive=ANC\_samples))

}

}

ANC\_data$ANC=1

survey\_data$ANC=0

data\_to\_model<-rbind(ANC\_data, survey\_data)%>%

filter(!(site %in% pred\_sites&ANC==0))

return(list(

param\_df=data.frame(preg\_par=ANC\_log\_odds\_ratio,

tau\_month=month\_sd),

pred\_sites=pred\_sites,

simulated\_survey\_data=survey\_data,

data\_to\_model=data\_to\_model

))

}

generate\_data\_m1\_runif\_month<-function(n\_month=50,

n\_sites=20,

site\_sd =2,

ANC\_log\_odds\_ratio=-0.5,

survey\_samples\_site\_month\_min=20,

survey\_samples\_site\_month\_max=100,

ANC\_samples\_site\_month\_min=10,

ANC\_samples\_site\_month\_max=50,

pred\_months=c(4:28,32:50)

){

# generate prevalence and site-specific params

survey\_prev <- runif(n\_month) # Random uniform probabilities for each month

log\_odd\_ratio\_site <- rnorm(n\_sites, 0, site\_sd) # Odds ratios for each site

# Initialize an empty dataframes for survey and ANC

survey\_data <- data.frame(month = integer(), site = integer(), sample\_size = integer(), positive = integer())

ANC\_data<- data.frame(month = integer(), site = integer(), sample\_size = integer(), positive = integer())

# Loop over sites and months

for (site in 1:n\_sites) {

for (month in 1:n\_month) {

survey\_sample\_size <- sample(survey\_samples\_site\_month\_min:survey\_samples\_site\_month\_max, 1)

ANC\_sample\_size <- sample(ANC\_samples\_site\_month\_min:ANC\_samples\_site\_month\_max, 1)

survey\_prevalence\_s\_m<-odds\_to\_probability(probability\_to\_odds(survey\_prev[month])\*exp(log\_odd\_ratio\_site[site]))

ANC\_prev\_s\_m<-odds\_to\_probability(probability\_to\_odds(survey\_prevalence\_s\_m)\*exp(ANC\_log\_odds\_ratio))

survey\_positive <- rbinom(1, size = survey\_sample\_size, prob = survey\_prevalence\_s\_m)

ANC\_samples<-rbinom(ANC\_sample\_size,1,ANC\_prev\_s\_m)

survey\_data <- rbind(survey\_data, data.frame(month=month, site=site,sample\_size=survey\_sample\_size,positive=survey\_positive))

ANC\_data <- rbind(ANC\_data, data.frame(month=month, site=site, sample\_size=1, positive=ANC\_samples))

}

}

ANC\_data$ANC=1

survey\_data$ANC=0

data\_to\_model<-rbind(ANC\_data, survey\_data)%>%

filter(!(month %in% pred\_months&ANC==0))

return(list(

param\_df=data.frame(preg\_par=ANC\_log\_odds\_ratio,

tau\_site=site\_sd),

pred\_months=pred\_months,

simulated\_survey\_data=survey\_data,

data\_to\_model=data\_to\_model

))

}

generate\_data\_m5<-function(n\_month=50,

n\_sites=20,

base\_par=0,

month\_sd=2,

site\_sd =1,

ANC\_log\_odds\_intercept\_cat1=0.1,

ANC\_log\_odds\_intercept\_cat2=-0.2,

ANC\_log\_odds\_intercept\_cat3=-0.5,

ANC\_log\_odds\_gradient\_cat1=-0.15,

ANC\_log\_odds\_gradient\_cat2=-0.3,

ANC\_log\_odds\_gradient\_cat3=-0.6,

prop\_cat1=0.2,

prop\_cat2=0.35,

survey\_samples\_site\_month\_min=20,

survey\_samples\_site\_month\_max=100,

ANC\_samples\_site\_month\_min=10,

ANC\_samples\_site\_month\_max=50,

pred\_months=c(4:28,32:50)

){

# generate prevalence and site-specific params

#survey\_prev <- runif(n\_month) # Random uniform probabilities for each month

log\_odd\_ratio\_month <- rnorm(n\_month, 0, month\_sd) # Odds ratios for each site

log\_odd\_ratio\_site <- rnorm(n\_sites, 0, site\_sd) # Odds ratios for each site

# Initialize an empty dataframes for survey and ANC

survey\_data <- data.frame(month = integer(), site = integer(), sample\_size = integer(), positive = integer())

ANC\_data<- data.frame(month = integer(), site = integer(), sample\_size = integer(), positive = integer())

# Loop over sites and months

for (site in 1:n\_sites) {

for (month in 1:n\_month) {

survey\_sample\_size <- sample(survey\_samples\_site\_month\_min:survey\_samples\_site\_month\_max, 1)

ANC\_sample\_size <- sample(ANC\_samples\_site\_month\_min:ANC\_samples\_site\_month\_max, 1)

ANC\_sample\_by\_grav\_cat<-rmultinom(ANC\_sample\_size,1,prob=c(prop\_cat1,prop\_cat2,1-prop\_cat1-prop\_cat2))

gravcat1<-ANC\_sample\_by\_grav\_cat[1,]

gravcat2<-ANC\_sample\_by\_grav\_cat[2,]

gravcat3<-ANC\_sample\_by\_grav\_cat[3,]

#p<-ilogit(base\_par+primi\_trans\_par\*theta\_site[as.numeric(adjust\_data$site)]\*adjust\_data$primi+primi\_par\*adjust\_data$primi+multi\_trans\_par\*(theta\_site[as.numeric(adjust\_data$site)])\*adjust\_data$multi+multi\_par\*adjust\_data$multi+grand\_trans\_par\*(theta\_site[as.numeric(adjust\_data$site)])\*adjust\_data$grand+grand\_par\*adjust\_data$grand+theta\_month[as.numeric(adjust\_data$month\_code)]+theta\_site[as.numeric(adjust\_data$site)])

survey\_prevalence\_s\_m<-odds\_to\_probability(exp(base\_par+log\_odd\_ratio\_month[month]+log\_odd\_ratio\_site[site]))

odds\_ANC<-exp(

base\_par+log\_odd\_ratio\_month[month]+log\_odd\_ratio\_site[site]+(ANC\_log\_odds\_intercept\_cat1+log\_odd\_ratio\_site[site]\*ANC\_log\_odds\_gradient\_cat1)\*gravcat1+

(ANC\_log\_odds\_intercept\_cat2+log\_odd\_ratio\_site[site]\*ANC\_log\_odds\_gradient\_cat2)\*gravcat2+

(ANC\_log\_odds\_intercept\_cat3+log\_odd\_ratio\_site[site]\*ANC\_log\_odds\_gradient\_cat3)\*gravcat3

)

ANC\_prev\_s\_m<-odds\_to\_probability(odds\_ANC)

survey\_positive <- rbinom(1, size = survey\_sample\_size, prob = survey\_prevalence\_s\_m)

ANC\_samples<-rbinom(n=ANC\_sample\_size,size=1,prob=ANC\_prev\_s\_m)

survey\_data <- rbind(survey\_data, data.frame(month=month, site=site,sample\_size=survey\_sample\_size,positive=survey\_positive,gravcat1=0,gravcat2=0,gravcat3=0))

ANC\_data <- rbind(ANC\_data, data.frame(month=month, site=site, sample\_size=1, positive=ANC\_samples,gravcat1=gravcat1,gravcat2=gravcat2,gravcat3=gravcat3))

}

}

ANC\_data$ANC=1

survey\_data$ANC=0

data\_to\_model<-rbind(ANC\_data, survey\_data)%>%

filter(!(month %in% pred\_months&ANC==0))

return(list(

param\_df=data.frame(

base\_par=base\_par,

tau\_site=site\_sd,

tau\_month=month\_sd,

log\_odd\_ratio\_intercept\_GC1=ANC\_log\_odds\_intercept\_cat1,

log\_odd\_ratio\_intercept\_GC2=ANC\_log\_odds\_intercept\_cat2,

log\_odd\_ratio\_intercept\_GC3=ANC\_log\_odds\_intercept\_cat3,

log\_odd\_ratio\_gradient\_GC1=ANC\_log\_odds\_gradient\_cat1,

log\_odd\_ratio\_gradient\_GC2=ANC\_log\_odds\_gradient\_cat2,

log\_odd\_ratio\_gradient\_GC3=ANC\_log\_odds\_gradient\_cat3

),

pred\_months=pred\_months,

simulated\_survey\_data=survey\_data,

data\_to\_model=data\_to\_model

))

}

deviance <- function(prob, y, n){

return(-2 \*sum(dbinom(y, size = n, prob = prob, log = TRUE)))

}

DIC\_calc<-function(data,probs,REs\_site\_dev,REs\_month\_dev,p\_temp,site\_ave,tau\_site\_ave,month\_ave,tau\_month\_ave){

logit\_dev <- unlist(lapply(X = probs,

FUN = deviance,

y = data$positive,

n = data$sample\_size))

p\_ave <- as.data.frame(do.call(rbind, calculate(p\_temp)))

Re\_site\_hat<- -2\*sum(log(1 / (tau\_site\_ave \* sqrt(2 \* pi)) \* exp(-((site\_ave) / tau\_site\_ave) ^ 2/2)))

Re\_month\_hat<- -2\*sum(log(1 / (tau\_month\_ave \* sqrt(2 \* pi)) \* exp(-((month\_ave) / tau\_month\_ave) ^ 2/2)))

logit\_hat <- deviance(prob=p\_ave$V1,

y = data$positive,

n = data$sample\_size)

mean\_dev<-quantile(logit\_dev+REs\_site\_dev+REs\_month\_dev,p=0.5)

d\_hat<-logit\_hat+Re\_site\_hat+Re\_month\_hat

#p\_d is the effective number of parameters

p\_d <- mean\_dev - d\_hat

DIC <- p\_d + mean\_dev

return(list(

DIC=DIC,

p\_d=p\_d

))

}

fit\_m1<-function(data,warmup,n\_samples,thin){

months=max(data$month)

sites=max(data$site)

tau\_month<-gamma(0.01,0.01)

tau\_site<-gamma(0.01,0.01)

theta\_month<-normal(0,tau\_month,months)

theta\_site<-normal(0,tau\_site,sites)

base\_par<-normal(0,100)

preg\_par<-normal(0,100)

p<-ilogit(base\_par+preg\_par\*data$ANC+theta\_month[data$month]+theta\_site[data$site])

distribution(data$positive)=binomial(data$sample\_size,p)

m<-model(base\_par,preg\_par,theta\_month,tau\_month,theta\_site,tau\_site)

draws <- mcmc(m, n\_samples = n\_samples, warmup = warmup,thin=thin)

fit\_array<-array(0,dim = c(n\_samples/thin\*4,sites,months))

for(i in 1:sites){

cmis\_log\_odds\_month\_site<-calculate(base\_par+theta\_site[i]+theta\_month,values=draws)

fit\_array[,i,]<-odds\_to\_probability(exp(as.matrix(cmis\_log\_odds\_month\_site)))

}

matrix<-odds\_to\_probability(exp(as.matrix(calculate(base\_par+theta\_month,values=draws))))

prev\_trends<-as.data.frame(colQuantiles(matrix,probs=c(0.5,0.025,0.975)))

prev\_trends$month=1:months

names(prev\_trends)<-c("med","low","high","month")

probs <- as.data.frame(t(do.call(rbind, calculate(p, values=draws))))

lik\_site <- (1 / (tau\_site \* sqrt(2 \* pi))) \* exp(-((theta\_site) / tau\_site) ^ 2/2)

lik\_month <- (1 / (tau\_month \* sqrt(2 \* pi))) \* exp(-((theta\_month) / tau\_month) ^ 2/2)

REs\_site\_dev<-unlist(calculate(-2\*sum(log(lik\_site)),values=draws))

REs\_month\_dev<-unlist(calculate(-2\*sum(log(lik\_month)),values=draws))

median\_coefs <- as.list(colQuantiles(do.call(rbind,draws),p=0.5))

base\_ave <- unlist(median\_coefs[grep('base\_par',names(median\_coefs))])

preg\_ave <- unlist(median\_coefs[grep('preg\_par',names(median\_coefs))])

month\_ave <- unlist(median\_coefs[grep('theta\_month',names(median\_coefs))])

site\_ave <- unlist(median\_coefs[grep('theta\_site',names(median\_coefs))])

tau\_site\_ave <- unlist(median\_coefs[grep('tau\_site',names(median\_coefs))])

tau\_month\_ave <- unlist(median\_coefs[grep('tau\_month',names(median\_coefs))])

p\_temp <- ilogit(base\_ave+preg\_ave\*data$ANC+month\_ave[data$month]+site\_ave[data$site])

#Use calculate to turn the greta array into a dataframe

DIC\_calc=DIC\_calc(data,probs,REs\_site\_dev,REs\_month\_dev,p\_temp,site\_ave,tau\_site\_ave,month\_ave,tau\_month\_ave)

return(

list(

data=data,

draws=draws,

prev\_trends=prev\_trends,

fit\_array=fit\_array,

p\_d=DIC\_calc$p\_d,

DIC=DIC\_calc$DIC

)

)

}

fit\_m5<-function(data,warmup,n\_samples,thin){

months=max(data$month)

sites=max(data$site)

tau\_month<-gamma(0.01,0.01)

tau\_site<-gamma(0.01,0.01)

base\_par<-normal(0,100)

theta\_month<-normal(0,tau\_month,months)

theta\_site<-normal(0,tau\_site,sites)

log\_odd\_ratio\_intercept\_GC1<-normal(0,100)

log\_odd\_ratio\_intercept\_GC2<-normal(0,100)

log\_odd\_ratio\_intercept\_GC3<-normal(0,100)

log\_odd\_ratio\_gradient\_GC1<-normal(0,100)

log\_odd\_ratio\_gradient\_GC2<-normal(0,100)

log\_odd\_ratio\_gradient\_GC3<-normal(0,100)

p<-ilogit(base\_par+theta\_site[data$site]+theta\_month[data$month] +

(log\_odd\_ratio\_intercept\_GC1+theta\_site[data$site]\*log\_odd\_ratio\_gradient\_GC1)\*data$gravcat1+

(log\_odd\_ratio\_intercept\_GC2+theta\_site[data$site]\*log\_odd\_ratio\_gradient\_GC2)\*data$gravcat2+

(log\_odd\_ratio\_intercept\_GC3+theta\_site[data$site]\*log\_odd\_ratio\_gradient\_GC3)\*data$gravcat3

)

distribution(data$positive)=binomial(data$sample\_size,p)

m<-model(base\_par,theta\_month,tau\_month,theta\_site,tau\_site,log\_odd\_ratio\_gradient\_GC1,log\_odd\_ratio\_gradient\_GC2,log\_odd\_ratio\_gradient\_GC3,log\_odd\_ratio\_intercept\_GC1,log\_odd\_ratio\_intercept\_GC2,log\_odd\_ratio\_intercept\_GC3)

draws <- mcmc(m, n\_samples = n\_samples, warmup = warmup,thin=thin)

fit\_array<-array(0,dim = c(n\_samples/thin\*4,sites,months))

for(i in 1:sites){

cmis\_log\_odds\_month\_site<-calculate(base\_par+theta\_site[i]+theta\_month,values=draws)

fit\_array[,i,]<-odds\_to\_probability(exp(as.matrix(cmis\_log\_odds\_month\_site)))

}

matrix<-odds\_to\_probability(exp(as.matrix(calculate(base\_par+theta\_month,values=draws))))

prev\_trends<-as.data.frame(colQuantiles(matrix,probs=c(0.5,0.025,0.975)))

prev\_trends$month=1:months

names(prev\_trends)<-c("med","low","high","month")

probs <- as.data.frame(t(do.call(rbind, calculate(p, values=draws))))

lik\_site <- (1 / (tau\_site \* sqrt(2 \* pi))) \* exp(-((theta\_site) / tau\_site) ^ 2/2)

lik\_month <- (1 / (tau\_month \* sqrt(2 \* pi))) \* exp(-((theta\_month) / tau\_month) ^ 2/2)

REs\_site\_dev<-unlist(calculate(-2\*sum(log(lik\_site)),values=draws))

REs\_month\_dev<-unlist(calculate(-2\*sum(log(lik\_month)),values=draws))

median\_coefs <- as.list(colQuantiles(do.call(rbind,draws),p=0.5))

base\_ave <- unlist(median\_coefs[grep('base\_par',names(median\_coefs))])

log\_odd\_ratio\_intercept\_GC1\_ave<-unlist(median\_coefs[grep('log\_odd\_ratio\_intercept\_GC1',names(median\_coefs))])

log\_odd\_ratio\_intercept\_GC2\_ave<-unlist(median\_coefs[grep('log\_odd\_ratio\_intercept\_GC2',names(median\_coefs))])

log\_odd\_ratio\_intercept\_GC3\_ave<-unlist(median\_coefs[grep('log\_odd\_ratio\_intercept\_GC3',names(median\_coefs))])

log\_odd\_ratio\_gradient\_GC1\_ave<-unlist(median\_coefs[grep('log\_odd\_ratio\_gradient\_GC1',names(median\_coefs))])

log\_odd\_ratio\_gradient\_GC2\_ave<-unlist(median\_coefs[grep('log\_odd\_ratio\_gradient\_GC2',names(median\_coefs))])

log\_odd\_ratio\_gradient\_GC3\_ave<-unlist(median\_coefs[grep('log\_odd\_ratio\_gradient\_GC3',names(median\_coefs))])

month\_ave <- unlist(median\_coefs[grep('theta\_month',names(median\_coefs))])

site\_ave <- unlist(median\_coefs[grep('theta\_site',names(median\_coefs))])

tau\_site\_ave <- unlist(median\_coefs[grep('tau\_site',names(median\_coefs))])

tau\_month\_ave <- unlist(median\_coefs[grep('tau\_month',names(median\_coefs))])

p\_temp <- ilogit(base\_ave+site\_ave[data$site]+month\_ave[data$month] +

(log\_odd\_ratio\_intercept\_GC1\_ave+site\_ave[data$site]\*log\_odd\_ratio\_gradient\_GC1\_ave)\*data$gravcat1+

(log\_odd\_ratio\_intercept\_GC2\_ave+site\_ave[data$site]\*log\_odd\_ratio\_gradient\_GC2\_ave)\*data$gravcat2+

(log\_odd\_ratio\_intercept\_GC3\_ave+site\_ave[data$site]\*log\_odd\_ratio\_gradient\_GC3\_ave)\*data$gravcat3

)

#Use calculate to turn the greta array into a dataframe

DIC\_calc=DIC\_calc(data,probs,REs\_site\_dev,REs\_month\_dev,p\_temp,site\_ave,tau\_site\_ave,month\_ave,tau\_month\_ave)

return(

list(

data=data,

draws=draws,

prev\_trends=prev\_trends,

fit\_array=fit\_array,

p\_d=as.numeric(DIC\_calc$p\_d),

DIC=as.numeric(DIC\_calc$DIC)

)

)

}

predict\_months\_m1<-function(data,warmup,n\_samples,thin){

months=max(data$month)

sites=max(data$site)

theta\_month<-normal(0,100,months)

tau\_site<-gamma(0.01,0.01)

theta\_site<-normal(0,tau\_site,sites)

preg\_par<-normal(0,100)

p<-ilogit(preg\_par\*data$ANC+theta\_month[data$month]+theta\_site[data$site])

distribution(data$positive)=binomial(data$sample\_size,p)

m<-model(preg\_par,theta\_month,theta\_site,tau\_site)

draws <- mcmc(m, n\_samples = n\_samples, warmup = warmup,thin=thin)

fit\_array<-array(0,dim = c(n\_samples/thin\*4,sites,months))

for(i in 1:sites){

cmis\_log\_odds\_month\_site<-theta\_site[i]+theta\_month

fit\_array[,i,]<-as.matrix(calculate(odds\_to\_probability(exp(cmis\_log\_odds\_month\_site)),values=draws))

}

matrix<-as.matrix(calculate(odds\_to\_probability(exp(theta\_month)),values=draws))

prev\_trends<-as.data.frame(colQuantiles(matrix,probs=c(0.5,0.025,0.975)))

prev\_trends$month=1:months

names(prev\_trends)<-c("med","low","high","month")

return(

list(

data=data,

draws=draws,

prev\_trends=prev\_trends,

fit\_array=fit\_array

)

)

}

predict\_sites\_m1<-function(data,warmup,n\_samples,thin){

months=max(data$month)

sites=max(data$site)

tau\_month<-gamma(0.01,0.01)

theta\_month<-normal(0,tau\_month,months)

theta\_site<-normal(0,100,sites)

preg\_par<-normal(0,100)

p<-ilogit(preg\_par\*data$ANC+theta\_month[data$month]+theta\_site[data$site])

distribution(data$positive)=binomial(data$sample\_size,p)

m<-model(preg\_par,theta\_month,theta\_site,tau\_month)

draws <- mcmc(m, n\_samples = n\_samples, warmup = warmup,thin=thin)

fit\_array<-array(0,dim = c(n\_samples/thin\*4,sites,months))

for(i in 1:sites){

cmis\_log\_odds\_month\_site<-theta\_site[i]+theta\_month

fit\_array[,i,]<-as.matrix(calculate(odds\_to\_probability(exp(cmis\_log\_odds\_month\_site)),values=draws))

}

matrix<-as.matrix(calculate(odds\_to\_probability(exp(theta\_month)),values=draws))

prev\_trends<-as.data.frame(colQuantiles(matrix,probs=c(0.5,0.025,0.975)))

prev\_trends$month=1:months

names(prev\_trends)<-c("med","low","high","month")

return(

list(

data=data,

draws=draws,

prev\_trends=prev\_trends,

fit\_array=fit\_array

)

)

}

### plot\_functions.R

get\_plots <- function(fitted\_model, survey\_data, param\_df, pred\_months=F, pred\_sites=F) {

cols<-viridis(2)

# Extract predictions and observed statistics

site\_preds <- get\_weight\_site\_pred(fitted\_model$fit\_array, survey\_data)

prev\_by\_site <- survey\_data %>%

group\_by(site) %>%

summarise(observed\_mean = sum(positive) / sum(sample\_size),

observed\_lower = binom.confint(sum(positive), sum(sample\_size), methods = "wilson")$lower,

observed\_upper = binom.confint(sum(positive), sum(sample\_size), methods = "wilson")$upper)%>%

mutate(Status = ifelse(site %in% pred\_sites, "Predicted", "Fitted"))

prev\_by\_site <- cbind(prev\_by\_site, rowQuantiles(site\_preds, p = c(0.025, 0.5, 0.975)))

names(prev\_by\_site) <- c("site", "observed\_mean", "observed\_lower", "observed\_upper","Status","model\_lower", "model\_median", "model\_upper")

# Spatial plot setup

spatial\_plot <- ggplot(prev\_by\_site, aes(x = observed\_mean, y = model\_median,color=Status)) +

geom\_point() +

geom\_errorbar(aes(ymin = model\_lower, ymax = model\_upper), width = 0.01) +

geom\_errorbarh(aes(xmin = observed\_lower, xmax = observed\_upper), height = 0.01) +

labs(x = "Observed Prevalence (Mean)", y = "Modeled Prevalence (Median)")+

theme\_minimal()+geom\_abline()

# Temporal analysis setup

temp\_preds <- get\_weight\_month\_pred(fitted\_model$fit\_array, survey\_data)

prev\_by\_month <- survey\_data %>%

group\_by(month) %>%

summarise(observed\_mean = sum(positive) / sum(sample\_size),

observed\_lower = binom.confint(sum(positive), sum(sample\_size), methods = "wilson")$lower,

observed\_upper = binom.confint(sum(positive), sum(sample\_size), methods = "wilson")$upper) %>%

mutate(Status = ifelse(month %in% pred\_months, "Predicted", "Fitted")) %>%

arrange(month) %>%

mutate(change\_in\_month = c(1, diff(month) > 1),

change\_in\_status = c(1, diff(as.numeric(as.factor(Status))) != 0),

Segment = cumsum(change\_in\_month | change\_in\_status)) %>%

select(-change\_in\_month, -change\_in\_status) %>%

cbind(., rowQuantiles(temp\_preds, p = c(0.025, 0.5, 0.975)))

names(prev\_by\_month) <- c("month", "observed\_mean", "observed\_lower", "observed\_upper","Status","Segment","model\_lower", "model\_median", "model\_upper")

temporal\_plot <- ggplot(prev\_by\_month, aes(x = month)) +

geom\_point(aes(y = observed\_mean), color = "blue") +

geom\_errorbar(aes(ymin = observed\_lower, ymax = observed\_upper), width = 0.2, color = "blue") +

geom\_ribbon(aes(ymin = model\_lower, ymax = model\_upper, fill = Status, group = Segment), alpha = 0.5) +

geom\_line(aes(y = model\_median, color = Status, group = Segment)) +

scale\_fill\_manual(values = c("Predicted" = cols[1], "Fitted" = cols[2])) +

scale\_color\_manual(values = c("Predicted" = cols[1], "Fitted" = cols[2])) +

labs(x = "Month", y = "Prevalence", title = "Observed vs Modelled Prevalence by Month") +

theme\_minimal() +

guides(fill = guide\_legend(title = "Legend"), color = guide\_legend(title = "Legend"))

# Posterior distribution analysis

sample\_matrix <- as.matrix(fitted\_model$draws)

post\_samples <- as.data.frame(sample\_matrix[, !grepl("^theta", colnames(sample\_matrix))])

posterior\_long <- pivot\_longer(post\_samples, cols = everything(), names\_to = "parameter", values\_to = "value")

color\_mapping <- viridis::viridis(length(unique(posterior\_long$parameter)), option = "C")

names(color\_mapping) <- unique(posterior\_long$parameter)

posterior\_long$color <- color\_mapping[as.character(posterior\_long$parameter)]

simulated\_long <- pivot\_longer(param\_df, cols = everything(), names\_to = "parameter", values\_to = "value")

simulated\_long$color <- color\_mapping[as.character(simulated\_long$parameter)]

posterior\_plot <- ggplot(posterior\_long, aes(x = value, y = parameter, fill = color)) +

geom\_density\_ridges(rel\_min\_height = 0.01, alpha = 0.25) +

scale\_fill\_identity() +

geom\_vline(data = simulated\_long, aes(xintercept = value, color = color), linetype = "dashed", linewidth = 1) +

scale\_color\_identity() +

theme\_ridges() +

labs(title = "Posterior Distributions with Simulated Values", x = "Parameter Values", y = "Parameter") +

theme(legend.position = "none")

# Composite plot

overall\_plot <- (posterior\_plot | spatial\_plot) / temporal\_plot +

plot\_layout(heights = c(1, 2))

return(list(

posterior\_plot = posterior\_plot,

spatial\_plot = spatial\_plot,

temporal\_plot = temporal\_plot,

overall\_plot = overall\_plot

))

}

### main.R

source("setup.R")

source("functions.R")

source("data\_generation.R")

source("models.R")

source("plot\_functions.R")

set.seed(99)

## generate synthetic data representing most simple and complex model

## model 1 - simplest:

## prevalence varying by site and month, constant log-odds diff ANC vs community

m1\_data<-generate\_data\_m1()

## model 5 - most complex:

##prevalence varying by site and month, linear relationship on logit-scale with different slopes and intercepts depending upon gravidity category

m5\_data<-generate\_data\_m5()

### fit models

m1\_fitting<-fit\_m1(m1\_data$data\_to\_model,1000,2000,1)

m5\_fitting<-fit\_m5(m5\_data$data\_to\_model,100,200,1)

### check model recaptures parameters

m1\_plots<-get\_plots(m1\_fitting,m1\_data$simulated\_survey\_data,m1\_data$param\_df)

m1\_plots

m5\_plots<-get\_plots(m5\_fitting,m5\_data$simulated\_survey\_data,m5\_data$param\_df)

m5\_plots

## fit the simplest model to data generated by the most complex to highlight fitting metrics

m1\_fit\_to\_m5\_data<-fit\_m1(m5\_data$data\_to\_model,100,200,1)

### can see visually a worse fit (NB parameter fit and simulated values should be different)

m1\_fit\_m5\_data\_plots<-get\_plots(m1\_fit\_to\_m5\_data,m5\_data$simulated\_survey\_data,m5\_data$param\_df)

m1\_fit\_m5\_data\_plots$overall\_plot

### model 5 has approx 6 more effective params..

m5\_fitting$p\_d-m1\_fit\_to\_m5\_data$p\_d

### but still gives better fit adjusting for overfitting via DIC

m5\_fitting$DIC-m1\_fit\_to\_m5\_data$DIC

#### USING MODELS TO RECONSTRUCT SURVEY DATA USING ANC

### Generate data with non-normal underlying prevalence by month

### default censors all data apart from a few months every two years

months\_to\_predict<-generate\_data\_m1\_runif\_month()

## fit model with fixed effects by months

predict\_months<-predict\_months\_m1(months\_to\_predict$data\_to\_model,100,200,1)

## should now see predictions for months as well as those included in the fitting

predict\_months\_plot<-get\_plots(predict\_months,months\_to\_predict$simulated\_survey\_data,months\_to\_predict$param\_df,pred\_months=months\_to\_predict$pred\_months)

predict\_months\_plot

### Generate data with non-normal underlying prevalence by site

### default censors sites 15-20

sites\_to\_predict<-generate\_data\_m1\_runif\_site()

## fit model with fixed effects by months

predict\_sites<-predict\_sites\_m1(sites\_to\_predict$data\_to\_model,100,200,1)

## should now see predictions for months as well as those included in the fitting

predict\_sites\_plot<-get\_plots(predict\_sites,sites\_to\_predict$simulated\_survey\_data,sites\_to\_predict$param\_df,pred\_sites=sites\_to\_predict$pred\_sites)

predict\_sites\_plot

The above is all of the code used in the project. I would like you to document it, provide code to make into rmarkdown, make and suggestions you see fit, help me with a github readme