Developing vignette: Estimate yearly models

This document is modified from the main vignette, to highlight the differences in fitting the model with yearly random effects.

Load Package and Data

```
library(INLA)
library(SUMMER)
library(ggplot2)
if (!isTRUE(requireNamespace("INLA", quietly = TRUE))) {
  install.packages('INLA', repos = 'https://www.math.ntnu.no/inla/R/stable')
}

data(Uganda)
data(UgandaMap)
```

Make Country Summary

```
years <- levels(Uganda[[1]]$time)

data0 <- countrySummary_mult(births = Uganda, years = years, idVar = "id", regionVar = "region",
                             timeVar = "time", clusterVar = "~clustid+id", ageVar = "age",
                             weightsVar = "weights", geo.recode = NULL)
```

Read Maps

```
    geo <- UgandaMap$geo
    mat <- UgandaMap$Amat
```

Make Priors

Using our adjacency matrix, we simulate hyperpriors using `simhyper`. For the new code to estimate yearly model, we default to the scaled version of the latent precision matrix, so we use the same hyperpriors for all random effects.

```
priors <- simhyper(R = 2, nsamp = 1e+05, nsamp.check = 5000, Amat = mat, only.iid = TRUE)
```

Prepare data for meta analysis

First, we aggregate estimators from different surveys.

```
data0$logit.prec <- 1/data0$var.est
time_region <- unique(data0[, c("region", "years")])

data <- data.frame(region = time_region$region, years = time_region$years, u5m = NA, lower=NA, upper=NA
expit<-function(x){
    exp(x)/(1+exp(x))
}
for(i in 1:dim(data)[1]){
  tmp <- intersect(which(data0$region == data$region[i]),
           which(data0$years == data$years[i]))
  # Version adjusting for HIV
  data[i, "logit.prec"] <- sum(data0[tmp, "logit.prec"], na.rm = TRUE)
  if(data[i, "logit.prec"] == 0){
    data[i, "var.est"] <- NA
    data[i, "logit.prec"] <- NA
  }else{
    data[i, "var.est"] <- 1 / data[i, "logit.prec"]
    weights <- data0[tmp, "logit.prec"] / data[i, "logit.prec"]
    data[i, "logit.est"] <- sum(weights * data0[tmp, "logit.est"], na.rm = TRUE)
    data[i, "u5m"] <- expit(data[i, "logit.est"])

    data[i, "lower"] <- expit(data[i, "logit.est"] + qnorm(0.975)*sqrt(data[i, "var.est"]))
    data[i, "upper"] <- expit(data[i, "logit.est"] + qnorm(0.025)*sqrt(data[i, "var.est"]))
  }
  data[i, "region_num"] <- data0[tmp, "region_num"][1]
 }
```

Fit INLA Model for national estimates

Now we are ready to fit the models. The codes to perform the new model fitting is attached at the end of this documentation.

First, we ignore the subnational estimates, and fit a model with temporal random effects only. In this part, we use the subset of data region variable being "All".

Period model

In fitting this model, we first define the list of time periods we wish to project the estimates on.

```
years.all <- c(years, "15-19")
fit1 <- fitINLA_yearly(data = data, geo = NULL, Amat = NULL, year_names = years.all, year_range = c(198
```

Yearly model

Similarly as before

```
fit2 <- fitINLA_yearly(data = data, geo = NULL, Amat = NULL, year_names = years.all, year_range = c(198
```

```
## Warning in inla.model.properties.generic(inla.trim.family(model), (mm[names(mm) == : Model 'rgeneric
##   Use this model with extra care!!! Further warnings are disabled.
```

Obtain smoothed estimates

The marginal posteriors are already stored in the fitted object. We use the following function to extract and re-arrange them.

```
projINLA_yearly <- function(fit, is.yearly=TRUE, year_range = c(1985, 2019), year_label = c("85-89", "9

  expit<-function(x){
      exp(x)/(1+exp(x))
  }

  if(is.null(Amat)){
    region_names <- "All"
    region_nums <- 0
  }else{
    region_names <- colnames(Amat)
    region_nums <- 1:length(region_names)
  }
  if(is.yearly){
    timelabel.yearly <- c(year_range[1] : year_range[2], year_label)
  }else{
    timelabel.yearly <- year_label
  }
  results <- expand.grid(District = region_nums, Year = timelabel.yearly)
  results$med <- results$q025 <- results$q975 <- results$logit.med <- results$logit.q025 <- results$log
  mod <- fit$fit
  lincombs.info <- fit$lincombs.info

  for(i in 1:length(timelabel.yearly)){
    for(j in 1:length(region_names)){
        index <- lincombs.info$Index[lincombs.info$District == region_nums[j] & lincombs.info$Year == i
        tmp.logit <- inla.rmarginal(nsim, mod$marginals.lincomb.derived[[index]])
        marg <- inla.tmarginal(expit, mod$marginals.lincomb.derived[[index]])
        tmp <- inla.rmarginal(nsim, marg)

        results$med[results$District == region_nums[j] & results$Year == timelabel.yearly[i]] <- median
        results$q975[results$District == region_nums[j] & results$Year == timelabel.yearly[i]] <- quant
        results$q025[results$District == region_nums[j] & results$Year == timelabel.yearly[i]] <- quant
        results$logit.med[results$District == region_nums[j] & results$Year == timelabel.yearly[i]] <- 
        results$logit.q975[results$District == region_nums[j] & results$Year == timelabel.yearly[i]] <-
        results$logit.q025[results$District == region_nums[j] & results$Year == timelabel.yearly[i]] <-

    }
  }
  results$is.yearly <- !(results$Year %in% year_label)
  results$Year.num <- suppressWarnings(as.numeric(as.character(results$Year)))
  if(region_names[1] != "All"){
    results$District <- region_names[results$District]
  }

  return(results)
}
```

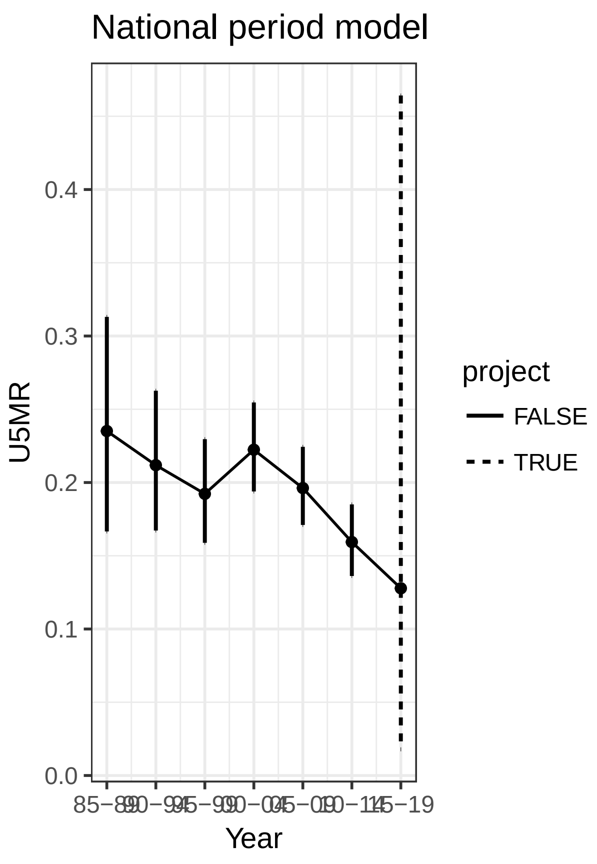
Now we can get the smoothed estimates for both models

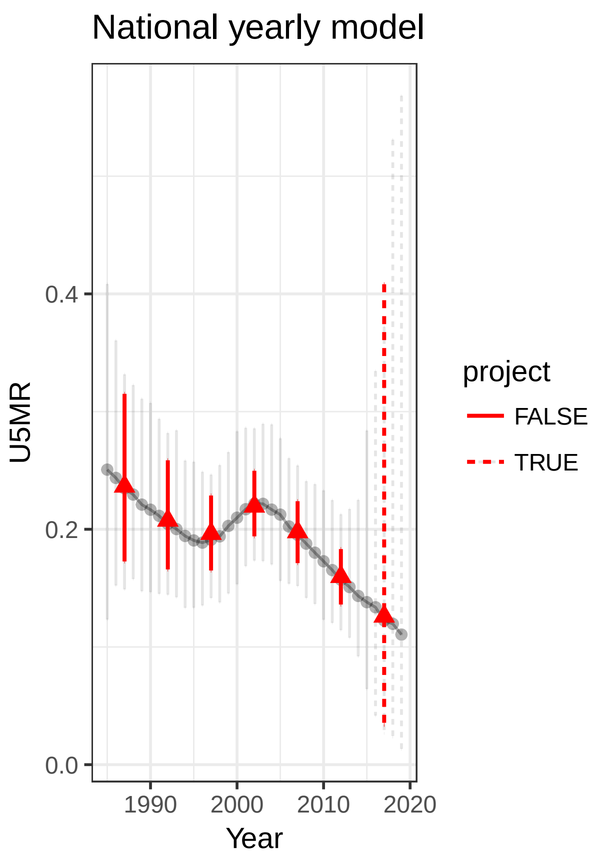
```
out1 <- projINLA_yearly(fit1, is.yearly = FALSE)
out2 <- projINLA_yearly(fit2, is.yearly = TRUE)
```

We can compare the results visually using the function below.

```r
plotINLA <- function(out, years_label = c("85-89", "90-94", "95-99", "00-04", "05-09", "10-14", "15-19"

is.periods <- out$Year %in% years_label
out$Year.num[is.periods] <- years_med[match(out$Year[is.periods], years_label)]
out$project <- FALSE
out$project[out$Year.num > proj_year] <- TRUE


if(is.subnational){
  g <- ggplot(aes(x = Year.num, y = med, ymin = q025, ymax = q975, color = District), data = out)
  my.dodge <- position_dodge(width = 1)
}else{
  g <- ggplot(aes(x = Year.num, y = med, ymin = q025, ymax = q975), data = out)
  my.dodge <- position_dodge(width = 0.2)
}

if(!is.yearly){
  g <- g + geom_point(position = my.dodge)
  g <- g + geom_line(position = my.dodge)
  g <- g + geom_errorbar(aes(linetype=project), size = .7, width = .05, position = my.dodge)
  g <- g + theme_bw() + xlab("Year") + ylab("U5MR")
  g <- g + scale_x_continuous(breaks=years_med, labels=years_label)
}else if(!is.subnational){
  g <- g + geom_point(position = my.dodge, data=subset(out, is.periods==FALSE), alpha = 0.3, color = 1)
  g <- g + geom_line(position = my.dodge, data=subset(out, is.periods==FALSE), alpha = 0.3, color = 1)
  g <- g + geom_errorbar(aes(linetype=project), size = .5, width = .05, position = my.dodge, data=subse
  g <- g + geom_point(shape = 17, size = 2.5, position = my.dodge, data=subset(out, is.periods==TRUE),
  g <- g + geom_errorbar(aes(linetype=project), size = .7, width = .05, position = my.dodge, data=subse
  g <- g + theme_bw() + xlab("Year") + ylab("U5MR")
}else if(is.subnational){
  g <- g + geom_point(position = my.dodge, data=subset(out, is.periods==FALSE), alpha = 0.3)
  g <- g + geom_line(position = my.dodge, data=subset(out, is.periods==FALSE), alpha = 0.3)
  g <- g + geom_point(shape = 17, size = 2.5, position = my.dodge, data=subset(out, is.periods==TRUE))
  g <- g + geom_errorbar(aes(linetype=project), size = .7, width = .05, position = my.dodge, data=subse
  g <- g + theme_bw() + xlab("Year") + ylab("U5MR")
}

return(g)
}
```

```r
library(gridExtra)
g <- NULL
g[[1]] <- plotINLA(out1, is.yearly=FALSE) + ggtitle("National period model")
g[[2]] <- plotINLA(out2, is.yearly=TRUE) + ggtitle("National yearly model")
grid.arrange(grobs=g, ncol = 2)
```





Fit INLA model for subnational estimates

Period model

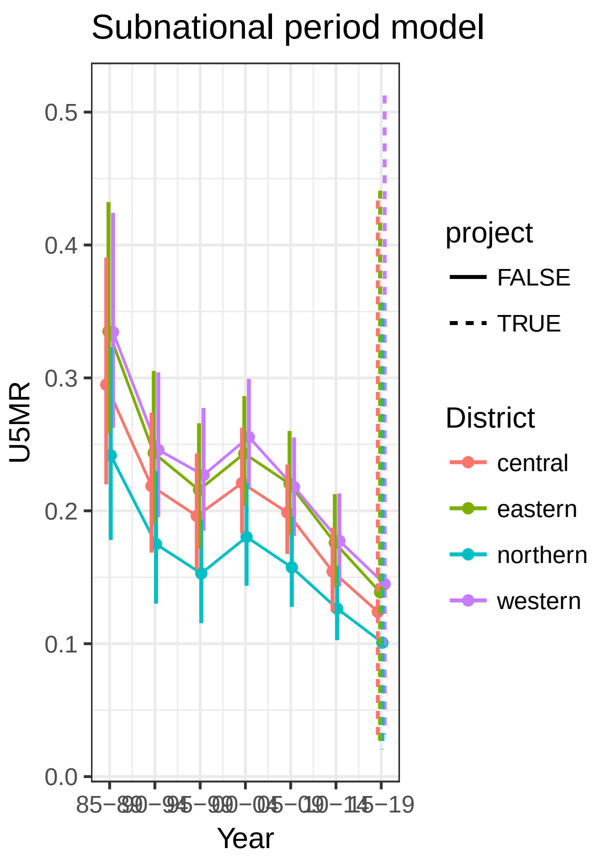
```
fit1 <- fitINLA_yearly(data = data, geo = geo, Amat = mat, year_names = years.all, year_range = c(1985,
out1 <- projINLA_yearly(fit1, Amat = mat, is.yearly = FALSE)
```

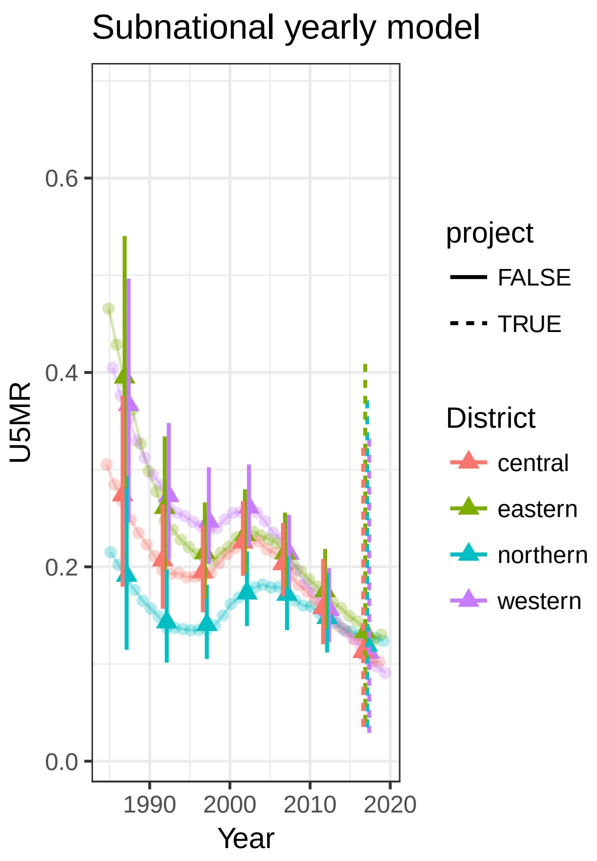
Yearly model with type IV interaction

```
fit2 <- fitINLA_yearly(data = data, geo = geo, Amat = mat, year_names = years.all, year_range = c(1985,
out2 <- projINLA_yearly(fit2, Amat = mat, is.yearly = TRUE)
```

Compare plots

```
g2 <- NULL
g2[[1]] <- plotINLA(out1, is.yearly=FALSE, is.subnational=TRUE) + ggtitle("Subnational period model")
g2[[2]] <- plotINLA(out2, is.yearly=TRUE, is.subnational=TRUE) + ggtitle("Subnational yearly model")
grid.arrange(grobs=g2, ncol = 2)
```





Appendix

```
#'
#' Function to fit INLA to the combined dataset
#' # Changes:
#' 1. default to without survey random effects, for package, either reset to Mercer paper setup, or add
#' 2. Add rw argument to control RW1 or RW2.
#' 3.
#'
#' # Unsure:
#' 1. What needs to be done for priors?
#'
#' # Additional parameters:
#' @param rw Take values 1 or 2, indicating the order of random walk.
#' @param is.yearly Logical indicator for fitting yearly or period model.
#' @param year_range Entire range of the years (inclusive) defined in year_names.
#' @param m Number of years in each period.
#'
#'
#'
  fitINLA_yearly <- function(data, Amat, geo, formula = NULL, rw = 2, is.yearly = TRUE, year_names, yea

      ## -----------------------------------------------------------
      ## New definition of the yearly + multi-year Q structure
      ## -----------------------------------------------------------
```

```
rw.new = function(cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"), t
## assume 'tau', 'order', 'n' and 'm' 'n' is the dim of RW and 'm' is the aggregated length,
## averaging over n/m variables, non-overlapping

## the environment of this function which holds the variables and we can store 'my.cache'
## there.
envir = environment(sys.call()[[1]])

if (!exists("my.cache", envir = envir, mode = "list")) {
  nn = n %/% m
  stopifnot (nn == as.integer(n/m))
  R = INLA:::inla.rw(n, order = order,  scale.model=TRUE, sparse=TRUE)
  A = matrix(0, nn, n)
  j = 1
  for(i in 1:nn) {
    A[i, j:(j+m-1)] = 1/m
    j = j + m
  }
  A = inla.as.sparse(A)
  D = Diagonal(nn, x=1)
  assign("my.cache", list(R=R, A=A, D=D, nn=nn), envir = envir)
}

interpret.theta = function() {
  return(list(kappa = exp(theta[1L])))
}

graph = function() {
  return (Q())
}

Q = function() {
  QQ = rBind(cBind(p$kappa * my.cache$R + tau * t(my.cache$A) %*% my.cache$A,
                   -tau * t(my.cache$A)),
             cBind(-tau * my.cache$A, tau * my.cache$D))
  return(QQ)
}

mu = function() {
  return(numeric(0))
}

log.norm.const = function() {
  val = (n-order) * (-0.5 * log(2 * pi) + 0.5 * log(p$kappa)) +
    (my.cache$nn * (-0.5 * log(2 * pi) + 0.5 * log(tau)))
  return(val)
}

log.prior = function() {
  val = dgamma(p$kappa, shape = shape0, rate = rate0, log = TRUE) + theta[1]
  return(val)
}
```

```
    initial = function() {
      return(4)
    }

    quit = function() {
      return(invisible())
    }

    ## as some calls to this function does not define 'theta',  its convenient to have to
    ## defined still (like in the graph-function)
    if (is.null(theta))
      theta = initial()

    p = interpret.theta()
    val = do.call(match.arg(cmd), args = list())
    return(val)
   }

   ## ----------------------------------------------------------
    ## New definition of the yearly + multi-year Q structure
    ## ----------------------------------------------------------
    iid.new = function(cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"),

    envir = environment(sys.call()[[1]])

    if (!exists("my.cache", envir = envir, mode = "list")) {
      nn = n %/% m
      stopifnot (nn == as.integer(n/m))
      R = Diagonal(n, x = rep(1, n))
      A = matrix(0, nn, n)
      j = 1
      for(i in 1:nn) {
        A[i, j:(j+m-1)] = 1/m
        j = j + m
      }
      A = inla.as.sparse(A)
      D = Diagonal(nn, x=1)
      assign("my.cache", list(R=R, A=A, D=D, nn=nn), envir = envir)
    }

    interpret.theta = function() {
      return(list(kappa = exp(theta[1L])))
    }

    graph = function() {
      return (Q())
    }

    Q = function() {
      QQ = rBind(cBind(p$kappa * my.cache$R + tau * t(my.cache$A) %*% my.cache$A,
                       -tau * t(my.cache$A)),
                 cBind(-tau * my.cache$A, tau * my.cache$D))
      return(QQ)
    }

    mu = function() {
      return(numeric(0))
    }

    log.norm.const = function() {
      val = (n * (-0.5 * log(2 * pi) + 0.5 * log(p$kappa)) +
        (my.cache$nn * (-0.5 * log(2 * pi) + 0.5 * log(tau))))
      return(val)
    }

    log.prior = function() {
      val = dgamma(p$kappa, shape = shape0, rate = rate0, log = TRUE) + theta[1]
      return(val)
    }

    initial = function() {
      return(4)
    }

    quit = function() {
      return(invisible())
    }

    ## as some calls to this function does not define 'theta',  its convenient to have to
    ## defined still (like in the graph-function)
    if (is.null(theta))
      theta = initial()

    p = interpret.theta()
    val = do.call(match.arg(cmd), args = list())
    return(val)
  }

  ## ---------------------------------------------------------
  ## New definition of the yearly + multi-year structured Q
  ## ---------------------------------------------------------
  st.new = function(cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"), th

  envir = environment(sys.call()[[1]])
  # The new structure takes the following order
  # (x_11, ..., x_1T, ..., x_S1, ..., x_ST, xx_11, ..., xx_1t, ..., xx_S1, ..., xx_St)
  #  x_ij : random effect of region i, year j
  # xx_ik : random effect of region i, period k

  if (!exists("my.cache", envir = envir, mode = "list")) {
    nn = n %/% m
    stopifnot (nn == as.integer(n/m))
    R1 = Diagonal(n, x = rep(1, n))
    R2 = INLA:::inla.rw(n, order = order, scale.model=TRUE, sparse=TRUE)
    R3 = Diagonal(S, x = rep(1, S))
    R4 = Amat
    diag(R4) <- 0
    diag <- apply(R4, 1, sum)
    R4[R4 != 0] <- -1
    diag(R4) <- diag
    R4 <- INLA:::inla.scale.model(R4, constr = list(A=matrix(1,1,dim(R4)[1]), e=0))
    # both independent
    if(type == 1){
        R <- R3 %x% R1
    # AR * independent
    }else if(type == 2){
        R <- R3 %x% R2
    # independent * besag
    }else if(type == 3){
        R <- R4 %x% R1
    # AR * besag
    }else if(type == 4){
        R <- R4 %x% R2
    }

    A = matrix(0, nn*S, n*S)
    j = 1
    for(i in 1:(nn*S)) {
      A[i, j:(j+m-1)] = 1/m
      j = j + m
    }
    A = inla.as.sparse(A)
    D = Diagonal(nn*S, x=1)
    assign("my.cache", list(R=INLA:::inla.as.sparse(R), A=A, D=D, nn=nn), envir = envir)
  }

  interpret.theta = function() {
    return(list(kappa = exp(theta[1L])))
  }

  graph = function() {
    return (Q())
  }

  Q = function() {
    QQ = rBind(cBind(p$kappa * my.cache$R + tau * t(my.cache$A) %*% my.cache$A,
                     -tau * t(my.cache$A)),
               cBind(-tau * my.cache$A, tau * my.cache$D))
    return(QQ)
  }

  mu = function() {
    return(numeric(0))
  }
  ## Type I   : S * n
  ## Type II  : S * (n - order)
  ## Type III : (S-1) * n
  ## Type IV  : (S-1) * (n - order)
  log.norm.const = function() {
    df <- S * n
    if(type == 2){
      df <- S * (n - order)
    }else if(type == 3){
      df <- (S-1) * n
    }else if(type == 4){
      df <- (S-1) * (n - order)
    }
    val = (df * (-0.5 * log(2 * pi) + 0.5 * log(p$kappa)) +
      (S * my.cache$nn * (-0.5 * log(2 * pi) + 0.5 * log(tau))))
    return(val)
  }

  log.prior = function() {
    val = dgamma(p$kappa, shape = shape0, rate = rate0, log = TRUE) + theta[1]
    return(val)
  }

  initial = function() {
    return(4)
  }

  quit = function() {
    return(invisible())
  }

  ## as some calls to this function does not define 'theta',  its convenient to have to
  ## defined still (like in the graph-function)
  if (is.null(theta))
    theta = initial()

  p = interpret.theta()
  val = do.call(match.arg(cmd), args = list())
  return(val)
}

  ## ---------------------------------------------------------
  ## Common Setup
  ## ---------------------------------------------------------
  if(is.null(geo)){
    data <- data[which(data$region == "All"), ]
    if(length(data) == 0){
      stop("No geographics specified and no observation labeled 'All' either.")
    }
  } else{
    data <- data[which(data$region != "All"), ]
  }
  ###################################################################### Re-calculate hyper-priors
  # Todo: make it work with the new Q matrix!!

  if (redo.prior) {
      priors <- simhyper(R = 2, nsamp = 1e+05, nsamp.check = 5000, Amat = Amat, nperiod = length(year
  }
```

```
a.iid <- priors$a.iid
b.iid <- priors$b.iid
a.rw1 <- priors$a.iid
b.rw1 <- priors$a.iid
a.rw2 <- priors$a.iid
b.rw2 <- priors$a.iid
a.icar <- priors$a.iid
b.icar <- priors$a.iid

###################################################################### # remove NA rows? e.g. if no 10
if (na.rm) {
    na.count <- apply(data, 1, function(x) {
        length(which(is.na(x)))
    })
    to_remove <- which(na.count == 6)
    if (length(to_remove) > 0)
        data <- data[-to_remove, ]
}
###################################################################### get the list of region and nume
if(is.null(geo)){
  region_names <- regions <- "All"
  region_count <- S <- 1
  dat <- cbind(data, region_number = 0)
}else{
  region_names <- colnames(Amat)
  region_count <- S <- length(region_names)
  regions <- data.frame(region = region_names, region_number = seq(1, region_count))
      # -- merging in the alphabetical region number -- #
  dat <- merge(data, regions, by = "region")
}

# -- creating IDs for the spatial REs -- #
dat$region.struct <- dat$region.unstruct <- dat$region_number

##################################################################### get the lsit of region and numer
if(is.yearly){
  n <- year_range[2] - year_range[1] + 1
  nn <- n %/% m
  N <- n + nn
  rw.model <- inla.rgeneric.define(model = rw.new,
                                  n = n,
                                  m = m,
                                  order = rw,
                                  tau = exp(10),
                                  shape0 = a.rw2,
                                  rate0 = b.rw2)
  iid.model.time <- inla.rgeneric.define(model = iid.new,
                                  n = n,
                                  m = m,
                                  tau = exp(10),
                                  shape0 = a.iid,
                                  rate0 = b.iid)
  st.model <- inla.rgeneric.define(model = st.new,
                                  n = n,
                                  m = m,
                                  order = rw,
                                  S = region_count,
                                  Amat = Amat,
                                  type = type.st,
                                  tau = exp(10),
                                  shape0 = a.iid,
                                  rate0 = b.iid)

  year_names_new <- c(as.character(c(year_range[1]:year_range[2])), year_names)
  time.index <- cbind.data.frame(idx = 1:N, Year = year_names_new)
  if(rw == 2){
    constr = list(A = matrix(c(rep(1, n), rep(0, nn),
                               1:n, rep(0, nn)), 2, N, byrow=TRUE), e = c(0,0))
  }else{
     constr = list(A = matrix(c(rep(1, n), rep(0, nn)), 1, N), e = 0)
  }

  # AR2 constraints
  if(type.st %in% c(2, 4) && rw == 2){
      tmp <- matrix(0, S * 2, N * S)
      for(i in 1:S){
        tmp[i*2-1, ((i-1)*n + 1) : (i*n)] <- 1
        tmp[i*2, ((i-1)*n + 1) : (i*n)] <- (1:n)
      }
  # AR1 constraints
  }else if(type.st %in% c(2, 4) && rw == 1){
    tmp <- matrix(0, S, N * S)
    for(i in 1:S){
      tmp[i, ((i-1)*n + 1) : (i*n)] <- 1
    }
  }else{
    tmp <- NULL
  }

  # ICAR constraints
  if(type.st %in% c(3, 4)){
    tmp2 <- matrix(0, n, N*S)
    for(i in 1:n){
        tmp2[i , (1:(n*S)) %% n == i-1] <- 1
      }
  }else{
    tmp2 <- NULL
  }
  tmp <- rbind(tmp, tmp2)
  if(is.null(tmp)){
    constr.st <- NULL
  }else{
    constr.st <- list(A = tmp, e = rep(0, dim(tmp)[1]))
  }
  years <- data.frame(year = year_names_new[1:N], year_number = seq(1, N))
}else{
  n <- 0
  N <- nn <- length(year_names)
  years <- data.frame(year = year_names, year_number = seq(1, N))
}

# -- creating IDs for the temporal REs -- #
if(is.yearly){
  dat$time.unstruct <- dat$time.struct <- years[match(dat$years, years[, 1]), 2]
}else{
  dat$time.unstruct <- dat$time.struct <- years[match(dat$years, years[, 1]), 2]
}

################################################################## get the number of surveys
if(sum(!is.na(data$survey)) == 0){
  data$survey <- 1
  nosurvey <- TRUE
}else{
  nosurvey <- FALSE
}
survey_count <- length(table(data$survey))
################################################################## -- these are the time X survey o
x <- expand.grid(1:nn, 1:survey_count)
survey.time <- data.frame(time.unstruct = x[, 1], survey = x[, 2], survey.time = c(1:nrow(x)))

# -- these are the area X survey options -- #
x <- expand.grid(1:region_count, 1:survey_count)
survey.area <- data.frame(region_number = x[, 1], survey = x[, 2], survey.area = c(1:nrow(x)))

# -- these are the area X time options -- #
# The new structure takes the following order
# (x_11, ..., x_1T, ..., x_S1, ..., x_ST, xx_11, ..., xx_1t, ..., xx_S1, ..., xx_St)
#  x_ij : random effect of region i, year j
# xx_ik : random effect of region i, period k
if(is.yearly){
  x <- rbind(expand.grid(1:n, 1:region_count),
             expand.grid((n+1):N, 1:region_count))
}else{
  x <- expand.grid(1:N, 1:region_count)
}
time.area <- data.frame(region_number = x[, 2], time.unstruct = x[, 1], time.area = c(1:nrow(x)))

# -- these are the area X time X survey options -- #
x <- expand.grid(1:region_count, 1:N, 1:survey_count)
survey.time.area <- data.frame(region_number = x[, 1], time.unstruct = x[, 2], survey = x[, 3], sur

# -- merge these all into the data sets -- #
newdata <- dat
if (sum(!is.na(dat$survey)) > 0) {
    newdata <- merge(newdata, survey.time, by = c("time.unstruct", "survey"))
    newdata <- merge(newdata, survey.area, by = c("region_number", "survey"))
    newdata <- merge(newdata, survey.time.area, by = c("region_number", "time.unstruct", "survey"))
}
if(!is.null(geo)){
  newdata <- merge(newdata, time.area, by = c("region_number", "time.unstruct"))
}else{
  newdata$time.area <- NA
}


########################### Model Selection ######

# -- subset of not missing and not direct estimate of 0 -- #
exdat <- newdata
exdat <- exdat[!is.na(exdat$logit.est) && exdat$logit.est > (-20), ]


## ----------------------------------------------------------
## Setup yearly model
## ----------------------------------------------------------
if(is.yearly && (!is.null(geo))){
  if (is.null(formula)) {
    if(rw == 1){
      formula <- logit.est ~ f(time.struct, model = rw.model, diagonal = 1e-6, extraconstr = constr
    }else if(rw == 2 && type.st %in% c(2, 4)){
      formula <- paste('logit.est ~ time.fix+f(time.struct, model = rw.model, diagonal = 1e-6, extra
        paste0("+ time.fix_", 1:S, collapse=" "))
      formula <- as.formula(formula)

    }else if(rw == 2){
      formula <- logit.est ~ time.fix+f(time.struct, model = rw.model, diagonal = 1e-6, extraconstr

    }else{
      stop("Random walk order should be 1 or 2.")
    }
}

## ----------------------------------------------------------
## Setup non-yearly model
## ----------------------------------------------------------
}else if((!is.yearly) && (!is.null(geo))){
  if (is.null(formula)) {
    if(rw == 1){
      formula <- logit.est ~ f(region.unstruct,model="iid",param=c(a.iid,b.iid)) + f(region.struct,
    }else if(rw == 2){
      constr = list(A = matrix(1:nn, 1, nn), e = 0)
      formula <- logit.est ~ time.fix+f(region.unstruct,model="iid",param=c(a.iid,b.iid)) + f(regio
          f(time.unstruct,model="iid",param=c(a.iid,b.iid)) + f(time.area,model="iid", param=c(a.ii
    }else{
      stop("Random walk order should be 1 or 2.")
    }
  }
## ----------------------------------------------------------
## Setup yearly national model
## ----------------------------------------------------------
}else if(is.yearly && is.null(geo)){
  if (is.null(formula)) {
    if(rw == 1){
      formula <- logit.est ~ f(time.struct, model = rw.model, diagonal = 1e-6, extraconstr = constr
    }else if(rw == 2){
      formula <- logit.est ~ time.fix+f(time.struct, model = rw.model, diagonal = 1e-6, extraconstr
    }else{
      stop("Random walk order should be 1 or 2.")
    }
  }

## ----------------------------------------------------------
## Setup non-yearly national model
## ----------------------------------------------------------
}else if((!is.yearly) && (is.null(geo))){
  if (is.null(formula)) {
    if(rw == 1){
      formula <- logit.est ~ f(time.struct,model="rw1",param=c(a.rw1,b.rw1))  + f(time.unstruct,mode
    }else if(rw == 2){
      constr = list(A = matrix(1:nn, 1, nn), e = 0)
      formula <- logit.est ~ time.fix+f(time.struct,model="rw2",param=c(a.rw2,b.rw2), extraconstr =
          f(time.unstruct,model="iid",param=c(a.iid,b.iid))
    }else{
      stop("Random walk order should be 1 or 2.")
    }
  }
}
mod <- formula



## ----------------------------------------------------------
## Subnational lincomb for projection
## ----------------------------------------------------------
if(!is.null(geo)){
  lincombs.info <- data.frame(Index = 1:(region_count*N), District = NA, Year = NA)
  index <- 0
  for(j in 1:region_count){
    for(i in 1:N){
      index <- index + 1
      time <- rep(NA, N)
      # time.old <- rep(NA, m)
      area <- rep(NA, region_count)
      spacetime <- rep(NA, N*region_count)

      space.time.id <- unique(time.area$time.area[time.area$time.unstruct == i & time.area$region_nu
      spacetime[space.time.id] <- 1
      time[i] <- 1
      area[j] <- 1
      time.unstruct <- time
      if(i <= n){
        timefix <- i
      }else{
        # fixed time effect: e.g., 36, ..., 42 -> (36-35-1)*5+2.5, ..., (42-35-1)*5+2.5
        timefix <- (i-n-1)*m + m/2
      }
      if(n == 0) timefix <- i

      object.name <- paste("lc", index, sep = "")

      lincombs.info[index, c("District", "Year")] <- c(j,i)
      if(rw == 1){
        assign(object.name, inla.make.lincomb("(Intercept)" = 1,
                                              time.area = spacetime,
                                              time.struct= time ,
                                              time.unstruct= time,
                                              region.struct = area,
                                              region.unstruct = area))
      }else if(is.yearly && type.st %in% c(2, 4) && rw == 2){
          # the name of the third argument is changed later
          assign(object.name, inla.make.lincomb("(Intercept)" = 1,
                                        time.fix = timefix,
                                        time.fix_temp = timefix,
                                        time.area = spacetime,
                                        time.struct= time ,
                                        time.unstruct= time,
                                        region.struct = area,
                                        region.unstruct = area))
      }else{
        assign(object.name, inla.make.lincomb("(Intercept)" = 1,
                                        time.fix = timefix,
                                        time.area = spacetime,
                                        time.struct= time ,
                                        time.unstruct= time,
                                        region.struct = area,
                                        region.unstruct = area))
      }

      if(index == 1){
        lincombs.yearly <- get(object.name)
        if(is.yearly && type.st %in% c(2, 4) && rw == 2){
          # correct name of the area time effect
          names(lincombs.yearly$lc[[3]]) <- paste0("time.fix_", j)
        }
        names(lincombs.yearly)[index] <- object.name
      }else{
        tmp <- get(object.name)
        if(is.yearly && type.st %in% c(2, 4) && rw == 2){
          names(tmp$lc[[3]]) <- paste0("time.fix_", j)
        }
        lincombs.yearly <- c(lincombs.yearly, tmp)
        names(lincombs.yearly)[index] <- object.name
      }
    }
  }

##------------------------------------------------------------##
## National model lincomb for projection
##------------------------------------------------------------##
```

```
}else{
    lincombs.info <- data.frame(Index = 1:N, District = NA, Year = NA)
    index <- 0
    for(i in 1:N){
       index <- index + 1
       time <- rep(NA, N)
       time[i] <- 1
       time.unstruct <- time
       if(i <= n){
         timefix <- i
       }else{
         timefix <- (i-n-1)*m + m/2
       }
       if(n == 0) timefix <- i

       object.name <- paste("lc", index, sep = "")

       lincombs.info[index, c("District", "Year")] <- c(0,i)
       if(rw == 1){
         assign(object.name, inla.make.lincomb("(Intercept)" = 1,
                                             time.struct= time ,
                                             time.unstruct= time))
       }else{
           assign(object.name, inla.make.lincomb("(Intercept)" = 1,
                                       time.fix = timefix,
                                       time.struct= time ,
                                       time.unstruct= time))
       }

       if(index == 1){
         lincombs.yearly <- get(object.name)
         names(lincombs.yearly)[index] <- object.name
       }else{
         lincombs.yearly <- c(lincombs.yearly, get(object.name))
         names(lincombs.yearly)[index] <- object.name
       }
   }
}

 # if(is.yearly){
   # rbind yearly data with NA for the lincombs
   for(i in 1:N){
     tmp<-exdat[match(unique(data$region), data$region), ]
     tmp$time.unstruct<-tmp$time.struct<- i
     tmp$logit.est<-tmp$logit.prec<-tmp$survey<-NA
     # tmp$survey.time<-tmp$survey.area<-tmp$survey.time.area<-NA
     tmp$time.area<-lincombs.info[lincombs.info[3] == i, 1]
     tmp$years<-years[i, 1]
     tmp$u5m <- tmp$lower <- tmp$upper <- tmp$var.est <- NA
     if("u5m.nohiv" %in% colnames(data)){
      tmp$u5m.nohiv <- tmp$lower.nohiv <- tmp$upper.nohiv <- tmp$var.est.nohiv<- tmp$logit.prec.nohi
     }
     exdat<-rbind(exdat,tmp)
```

```
      }
    # }

    # (n + 1), ..., (n + nn) ->
    if(n > 0){
      exdat$time.fix <- exdat$time.unstruct
      post <- which(exdat$time.unstruct > n)
      exdat$time.fix[post] <- (exdat$time.fix[post]-n-1)*m + m/2
    }else{
      exdat$time.fix <- exdat$time.unstruct
    }
    if(is.yearly && rw == 2){
      for(i in 1:S){
          name <- paste0("time.fix_", i)
          exdat[, name] <- exdat$time.fix
          exdat[exdat$region_number != i, name] <- 0
      }
    }

    # -- fitting the model in INLA -- #

    if (!isTRUE(requireNamespace("INLA", quietly = TRUE))) {
      stop("You need to install the packages 'INLA'. Please run in your R terminal:\n install.packages(
    }
    # If INLA is installed, then attach the Namespace (so that all the relevant functions are available
    if (isTRUE(requireNamespace("INLA", quietly = TRUE))) {
      if (!is.element("INLA", (.packages()))) {
        attachNamespace("INLA")
      }
      inla11 <- INLA::inla(mod, family = "gaussian", control.compute = list(dic = T, mlik = T, cpo = T)
        lincomb = lincombs.yearly)
    }

    return(list(model = mod, fit = inla11, Amat = Amat, newdata = exdat, time = seq(0, N - 1), area = s
        1), survey.time = survey.time, survey.area = survey.area, time.area = time.area, survey.time.ar
        a.iid = a.iid, b.iid = b.iid, a.rw1 = a.rw1, b.rw1 = b.rw1, a.rw2 = a.rw2, b.rw2 = b.rw2, a.ica

}
```