Cluster-Level Models

Andrea Boskovic and Katie Paulson

2023-02-22

In this document, we will review cluster-level modeling techniques. For comparison, we'll start with the smooth weighted (area-level) model, then show the lono-binomial model and the beta-binomial model. We will be using South Africa in this example, which will require the GADM shapefile (on Canvas as `gadm36_ZAF_shp.zip`) for South Africa and an HIV dataset labeled `hiv_df.csv` on Canvas.

Please reference both the SUMMER R notes and the SAE lecture notes for more context and description of these models.

Data Preparation

Before loading the data, make sure your data is in the same directory (folder) as this .Rmd file on your computer, and set your current working directory to that folder. You can do this by moving your mouse to the top of the screen in RStudio, clicking Session -> Set working directory -> To source file location.

```
# Load data
poly_adm1 <- st_read(
  dsn = "gadm36_ZAF_shp/",
  layer = "gadm36_ZAF_1"
)
```

```
## Reading layer `gadm36_ZAF_1' from data source
##   `/Users/kpaulson/Documents/STAT_554/gadm36_ZAF_shp' using driver `ESRI Shapefile'
## Simple feature collection with 9 features and 10 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 16.45189 ymin: -34.83514 xmax: 32.89125 ymax: -22.12503
## Geodetic CRS:  WGS 84
```

```
# Create the adjacency matrix
admin1_mat <- poly2nb(poly_adm1)
nb2INLA(file = "admin1_southafrica.graph", admin1_mat)
admin1_mat <- nb2mat(admin1_mat, zero.policy = TRUE)

# Make row and column names NAME_1
colnames(admin1_mat) <- rownames(admin1_mat) <- poly_adm1$NAME_1
admin1_names <- data.frame(
  GADM = poly_adm1$NAME_1,
  Internal = rownames(admin1_mat)
)

# Read in the dataset containing HIV prevalence information
hiv_df <- read.csv("hiv_df.csv")
```

The following `binom_df` will be used in the lono-binomial and beta-binomial models. It contains binomial

counts at the cluster level.

```
# Create binomial dataframe
binom_df <- hiv_df %>%
  group_by(clustid) %>%
  summarise(y = sum(hiv_ind, na.rm = T),
            N = length(which(!(is.na(hiv_ind)))),
            admin1 = admin1 %>% nth(1),
            admin1_name = admin1_name %>% nth(1)) %>%
  as.data.frame()
```

Smoothed weighted

There are two approaches to creating this model: we can use SUMMER (which is built on INLA) or we can use INLA directly.

SUMMER

We first create our model using `smoothSurvey()` in SUMMER.

```
smoothed_admin1 <- smoothSurvey(
  data = hiv_df %>% filter(!is.na(hiv_ind)),
  geo = poly_adm1, Amat = admin1_mat, responseType = "binary",
  responseVar = "hiv_ind", strataVar = "strata", weightVar = "hiv05",
  regionVar = "admin1_name", clusterVar = "~clustid+id", CI = 0.95
)
```

```
## Warning in inla.model.properties.generic(inla.trim.family(model), mm[names(mm) == : Model 'bym2' in s
##   Use this model with extra care!!! Further warnings are disabled.
```

Accessing the `smooth` attribute of the `smoothed_admin1` object gives us our estimates and variance. We can also see which formula was fit by accessing `formula`.

```
smoothed_admin1$smooth
```

```
##          region      mean          var    median      lower     upper
## 1  Eastern Cape 0.1654109 0.0001851273 0.1649307 0.14010011 0.1933224
## 2    Free State 0.2099268 0.0002764596 0.2093314 0.17879905 0.2440389
## 3       Gauteng 0.1944815 0.0003714209 0.1936603 0.15891180 0.2345057
## 4 KwaZulu-Natal 0.2524215 0.0003939397 0.2519546 0.21508737 0.2927906
## 5       Limpopo 0.1077261 0.0001630674 0.1070842 0.08453202 0.1344657
## 6    Mpumalanga 0.2337600 0.0003089430 0.2333232 0.20055144 0.2695043
## 7    North West 0.2035296 0.0002147030 0.2031589 0.17597030 0.2333662
## 8 Northern Cape 0.1312424 0.0004947340 0.1299727 0.09130159 0.1782767
## 9  Western Cape 0.1377092 0.0007735658 0.1357281 0.08919195 0.1978679
##   logit.mean   logit.var logit.median logit.lower logit.upper
## 1  -1.621754 0.009718325    -1.621989   -1.814459   -1.814459
## 2  -1.328280 0.010058017    -1.328960   -1.524505   -1.524505
## 3  -1.425764 0.015139784    -1.426400   -1.666347   -1.666347
## 4  -1.088479 0.011091488    -1.088215   -1.294528   -1.294528
## 5  -2.121089 0.017643251    -2.120877   -2.382305   -2.382305
## 6  -1.189764 0.009647405    -1.189641   -1.382851   -1.382851
## 7  -1.366802 0.008184527    -1.366667   -1.543891   -1.543891
## 8  -1.904111 0.038480135    -1.901200   -2.297845   -2.297845
## 9  -1.854358 0.055426069    -1.851234   -2.323541   -2.323541
```

```
smoothed_admin1$formula
```

```
## HT.logit.est ~ 1 + f(region.struct, graph = Amat, model = "bym2",
##     hyper = hyperpc2, scale.model = TRUE)
## <environment: 0x1237ca860>
```

INLA

We can also define and fit our model with INLA, which is a more flexible approach. First obtain direct estimates using the `survey` package.

```
design <- svydesign(data = hiv_df, ids = ~clustid + id, strata = ~strata,
                    weights = ~hiv05)
direct_admin1 <- svyby(~hiv_ind, by = ~ admin1_name, design = design, svymean, na.rm = T)
```

Note that we will be modeling prevalence estimates on the logit scale since $\text{logit}(p_i)$ lies on $(-\infty, \infty)$ rather than just $(0, 1)$. However, the direct estimates we've obtained give us the standard error for $\hat{p}_i$ not for $\widehat{\text{logit}(p_i)}$. For fitting the smoothed weighted model in INLA we will need to get the variance of $\widehat{\text{logit}(p_i)}$. A common statistical technique to do this is the delta method, which is implemented below. If the first line of code below is confusing, you can plug in the direct estimate and its corresponding standard error into the `get_logit_var()` function also defined below to produce the transformed (logit) variance for you.

```
# Logit variance
logit_var <- direct_admin1$se^2 / ((direct_admin1$hiv_ind - direct_admin1$hiv_ind^2)^2)

# Logit variance: Alternative method (using this pre-defined function)
get_logit_var <- function(direct_se, direct_est) {
  direct_se^2 / ((direct_est - direct_est^2)^2)
}
get_logit_var(direct_admin1$se, direct_admin1$hiv_ind)
```

```
## [1] 0.010495207 0.011033483 0.017168165 0.011730234 0.018037892 0.010307282
## [7] 0.008688993 0.049788941 0.075122761
```

We now define our model formula fit it using INLA. Make sure that the `region` variable goes from 1 to the total number of regions and that the areas are in alphabetical order!

```
# Define formula
formula <- logit_est ~ 1 + f(region, model = "bym2", graph = "admin1_southafrica.graph")
inla_dat <- data.frame(region = as.numeric(factor(direct_admin1$admin1_name)),
                       logit_est = logit(direct_admin1$hiv_ind),
                       logit_var = logit_var)

# Fit INLA model
result <- inla(
  formula = formula,
  data = inla_dat,
  family = "gaussian",
  control.family = list(hyper = list(prec = list(initial = log(1), fixed = TRUE))),
  scale = 1 / inla_dat$logit_var, # this is the fixed/known variance from the direct estimates
  control.predictor = list(compute = TRUE),
  control.compute = list(return.marginals = TRUE, config = TRUE, return.marginals.predictor = TRUE)
)
```

We can now obtain our posterior samples.

```r
# Get posterior samples
nsamp <- 1000
samp <- inla.posterior.sample(n = nsamp, result = result)

# Matrix to store results
samp_mat <- matrix(0, nrow = length(unique(binom_df$admin1)), ncol = nsamp)

# Fill matrix with results
for (i in 1:nsamp) {
  # first nregion values in samp[[i]]$latent correspond to predictions
  samp_mat[,i] <- samp[[i]]$latent[1:length(unique(binom_df$admin1))]
}
```

Now that we have filled our `samp_mat` with posterior samples, we need to apply an expit transformation to our posterior samples because we used $logit(p_i)$ and need to get back to our original p_i's. Recall that:

$$\text{expit}(\text{logit}(p_i)) = p_i,$$

and

$$\text{expit}(p_i) = \frac{\exp\{p_i\}}{1+\exp\{p_i\}} = \frac{1}{1+\exp\{-p_i\}}.$$

We also add row names so it is clear that each row corresponds to an area. Each column corresponds to a posterior sample, and we print out the first five columns below. We also want to summarize information like the posterior median, mean, standard deviation, and quantiles of our estimates, which we can do with the `apply()` function in R.

```r
# Expit the posterior samples
expit_samp_mat <- expit(samp_mat)
rownames(expit_samp_mat) <- sort(unique(binom_df$admin1_name))
expit_samp_mat[, 1:5]
```

```
##                    [,1]       [,2]       [,3]      [,4]       [,5]
## Eastern Cape  0.1646304 0.15679582 0.16494470 0.1583711 0.18408060
## Free State    0.2111671 0.21338613 0.22342652 0.2127151 0.20410801
## Gauteng       0.2012260 0.18182880 0.24447214 0.1917770 0.16568507
## KwaZulu-Natal 0.2448104 0.27168156 0.26168669 0.2668671 0.28690119
## Limpopo       0.1053218 0.09377182 0.08930234 0.1027407 0.09443568
## Mpumalanga    0.2296898 0.24017142 0.25913488 0.2885982 0.24741026
## North West    0.2201432 0.21591233 0.20096329 0.2275002 0.21464664
## Northern Cape 0.1259763 0.13496686 0.11282788 0.1190875 0.09637278
## Western Cape  0.1182540 0.16820057 0.10019731 0.1522970 0.12812472
```

```r
# Summarize desired posterior information
apply(expit_samp_mat, 1, median)
```

```
##  Eastern Cape    Free State       Gauteng KwaZulu-Natal       Limpopo
##     0.1647962     0.2091867     0.1934597     0.2513814     0.1072008
##    Mpumalanga    North West Northern Cape  Western Cape
##     0.2339021     0.2025434     0.1303546     0.1329746
```

```r
apply(expit_samp_mat, 1, mean)
```

```
##  Eastern Cape    Free State       Gauteng KwaZulu-Natal       Limpopo
##     0.1653827     0.2100692     0.1952760     0.2516385     0.1078150
```

```
##    Mpumalanga    North West Northern Cape  Western Cape
##     0.2344020     0.2035564     0.1313874     0.1355049
```

```
apply(expit_samp_mat, 1, sd)
```

```
##  Eastern Cape    Free State       Gauteng KwaZulu-Natal       Limpopo
##    0.01375602    0.01630674    0.01952355    0.01937730    0.01256112
##    Mpumalanga    North West Northern Cape  Western Cape
##    0.01784631    0.01455838    0.02243642    0.02804873
```

```
apply(expit_samp_mat, 1, quantile, c(0.025, 0.975))
```

```
##       Eastern Cape Free State   Gauteng KwaZulu-Natal    Limpopo Mpumalanga
## 2.5%     0.1405175  0.1796238 0.1603581     0.2130800 0.08501451  0.2008611
## 97.5%    0.1932890  0.2446390 0.2352008     0.2912732 0.13354731  0.2703769
##       North West Northern Cape Western Cape
## 2.5%   0.1764465    0.09282116   0.08664374
## 97.5%  0.2357477    0.17685872   0.19300695
```

```
samp_weighted <- expit_samp_mat
```

Lono-binomial Model

We define our model below, and we define `inla_dat` to be the same as our `binom_df`, i.e., the binomial dataframe at the cluster level. This model involves adding a i.i.d. cluster-level random effect to account for overdispersion, as seen with `f(clustid, model = "iid")`.

```
# Define the formula
formula <- y ~ 1 +
  f(admin1, model = "bym2", graph = "admin1_southafrica.graph") +
  f(clustid, model = "iid")
inla_dat <- binom_df

# Fit INLA model
result <- inla(formula = formula,
               Ntrials = N,
               data = inla_dat,
               family = "binomial",
               control.predictor = list(compute = TRUE),
               control.compute = list(return.marginals = TRUE,
                                      config = TRUE,
                                      return.marginals.predictor = TRUE))
```

Again, we want to draw samples from our posterior. To do so, we need to create matrices to contain posterior samples for each term in our linear predictor, which we'll later need to combine.

```
# Get posterior sample
nsamp <- 1000
samp <- inla.posterior.sample(n = nsamp, result = result)

# Matrix setup
region_idx <- which(rownames(samp[[1]]$latent) %>%
                      str_detect("admin1"))[1:length(unique(binom_df$admin1))]
int_idx <- which(rownames(samp[[1]]$latent) %>% str_detect("Intercept"))
cluster_tau_idx <- which(names(samp[[1]]$hyperpar) %>% str_detect("clustid"))
```

```
region_mat <- matrix(0, nrow = length(region_idx), ncol = nsamp)
int_mat <- matrix(0, nrow = 1, ncol = nsamp)
cluster_tau_mat <- matrix(0, nrow = 1, ncol = nsamp)

# Fill in matrix with posterior samples
for (i in 1:nsamp) {
  region_mat[,i] <- samp[[i]]$latent[region_idx]
  int_mat[,i] <- samp[[i]]$latent[int_idx]
  cluster_tau_mat[,i] <- samp[[i]]$hyperpar[cluster_tau_idx]
}
```

We now add the BYM2 term and intercept to get a matrix of predicted values. Once again, we need to do a correction, but this time we do a lono-binomial correction.

```
# Get predicted values
samp_mat <- region_mat + int_mat[rep(1,nrow(region_mat)),]

# Lono correction
h <- 16 * sqrt(3) / (15 * pi)
cluster_vars <- 1 / cluster_tau_mat
denom_samps <- sqrt(1 + h^2 * cluster_vars)

# Correct fitted samples
for (i in 1:ncol(samp_mat)) {
  samp_mat[,i] <- samp_mat[,i] / denom_samps[,i]
}
```

Again, we expit these values to get back to our p_i's and use the `apply()` function to get posterior summaries.

```
expit_samp_mat <- expit(samp_mat)
rownames(expit_samp_mat) <- sort(unique(binom_df$admin1_name))

#  Obtain summaries by region
apply(expit_samp_mat, 1, median)
```

```
##  Eastern Cape    Free State       Gauteng KwaZulu-Natal       Limpopo
##     0.1721992     0.2074620     0.1826934     0.2604224     0.1103416
##    Mpumalanga    North West Northern Cape  Western Cape
##     0.2316642     0.2079449     0.1217938     0.1187463
```

```
apply(expit_samp_mat, 1, mean)
```

```
##  Eastern Cape    Free State       Gauteng KwaZulu-Natal       Limpopo
##     0.1724878     0.2074631     0.1846955     0.2608114     0.1107707
##    Mpumalanga    North West Northern Cape  Western Cape
##     0.2322166     0.2080904     0.1224753     0.1194190
```

```
apply(expit_samp_mat, 1, sd)
```

```
##  Eastern Cape    Free State       Gauteng KwaZulu-Natal       Limpopo
##    0.01259190    0.01612968    0.01820626    0.01732169    0.01224505
##    Mpumalanga    North West Northern Cape  Western Cape
##    0.01608347    0.01525572    0.01583916    0.01636663
```

```
apply(expit_samp_mat, 1, quantile, c(0.025, 0.975))
```

```
##       Eastern Cape Free State   Gauteng KwaZulu-Natal    Limpopo Mpumalanga
## 2.5%     0.1479347  0.1783548 0.1538346     0.2291504 0.08783535  0.2023270
```

```
## 97.5%    0.1975697  0.2395815 0.2218002     0.2956045 0.13601935  0.2633222
##       North West Northern Cape Western Cape
## 2.5%   0.1782612    0.09248362   0.09098552
## 97.5%  0.2388216    0.15564934   0.15123459
```

```
samp_lono <- expit_samp_mat
```

Beta-Binomial Model

Unlike the lono-binomial model, the beta-binomial model does NOT have a cluster-level random effect to account for overdispersion because with this model, overdispersion is dealt with through its likelihood.

```
# Define the formula
formula <- y ~ 1 +
  f(admin1, model = "bym2", graph = "admin1_southafrica.graph")
inla_dat <- binom_df

# Fit INLA model
result <- inla(formula = formula,
               Ntrials = N,
               data = inla_dat,
               family = "betabinomial",
               control.predictor = list(compute = TRUE),
               control.compute = list(return.marginals = TRUE,
                                      config = TRUE,
                                      return.marginals.predictor = TRUE))
```

We again repeat a similar process to extract posterior samples.

```
# Obtain posterior samples
nsamp <- 1000
samp <- inla.posterior.sample(n = nsamp, result = result)

# Set up matrices
region_idx <- which(rownames(samp[[1]]$latent) %>%
                      str_detect("admin1"))[1:length(unique(binom_df$admin1))]
int_idx <- which(rownames(samp[[1]]$latent) %>% str_detect("Intercept"))
region_mat <- matrix(0, nrow = length(region_idx), ncol = nsamp)
int_mat <- matrix(0, nrow = 1, ncol = nsamp)

# Fill in matrix with posterior samples
for (i in 1:nsamp) {
  region_mat[,i] <- samp[[i]]$latent[region_idx]
  int_mat[,i] <- samp[[i]]$latent[int_idx]
}
```

Again, to get back to the prevalence scale, we expit our values and use those values of p_i to calculate our posterior summary statistics.

```
samp_mat <- region_mat + int_mat[rep(1,nrow(region_mat)),]
expit_samp_mat <- expit(samp_mat)
rownames(expit_samp_mat) <- sort(unique(binom_df$admin1_name))

# Obtain summaries by region
apply(expit_samp_mat, 1, median)
```

```
##  Eastern Cape    Free State       Gauteng KwaZulu-Natal       Limpopo
##     0.1750799     0.2088741     0.1866013     0.2610282     0.1139016
##    Mpumalanga    North West Northern Cape  Western Cape
##     0.2341269     0.2094574     0.1204211     0.1125183
```

```
apply(expit_samp_mat, 1, mean)
```

```
##  Eastern Cape    Free State       Gauteng KwaZulu-Natal       Limpopo
##     0.1753480     0.2085249     0.1866262     0.2624200     0.1146284
##    Mpumalanga    North West Northern Cape  Western Cape
##     0.2347820     0.2102980     0.1217458     0.1136727
```

```
apply(expit_samp_mat, 1, sd)
```

```
##  Eastern Cape    Free State       Gauteng KwaZulu-Natal       Limpopo
##    0.01414481    0.01669501    0.01775484    0.01777095    0.01340178
##    Mpumalanga    North West Northern Cape  Western Cape
##    0.01712785    0.01630845    0.01706533    0.01748011
```

```
apply(expit_samp_mat, 1, quantile, c(0.025, 0.975))
```

```
##       Eastern Cape Free State   Gauteng KwaZulu-Natal    Limpopo Mpumalanga
## 2.5%     0.1491742  0.1774417 0.1540105     0.2295305 0.09079349  0.2015104
## 97.5%    0.2054132  0.2427587 0.2241350     0.2998690 0.14239773  0.2694140
##       North West Northern Cape Western Cape
## 2.5%   0.1792101    0.09157094   0.08204183
## 97.5%  0.2436553    0.15928424   0.14894989
```

```
samp_betabinom <- expit_samp_mat
```

Comparison

```
# Table comparing medians from each model
data.frame(smoothed_weighted = apply(samp_weighted, 1, median) %>% round(2),
           lono = apply(samp_lono, 1, median) %>% round(2),
           betabinom = apply(samp_betabinom, 1, median) %>% round(2)) %>%
  kable("simple")
```

	smoothed_weighted	lono	betabinom
Eastern Cape	0.16	0.17	0.18
Free State	0.21	0.21	0.21
Gauteng	0.19	0.18	0.19
KwaZulu-Natal	0.25	0.26	0.26
Limpopo	0.11	0.11	0.11
Mpumalanga	0.23	0.23	0.23
North West	0.20	0.21	0.21
Northern Cape	0.13	0.12	0.12
Western Cape	0.13	0.12	0.11

Acknowledgements

Credit to Taylor Okonek for the initial version of these examples, as well as Jon Wakefield.