Lesson 22

Clark

Clayton and Kaldor (1987) analyzed lip cancer rates in Scotland in the years 1975-1980 at the county level in order to evaluate the presence of an association between sun exposure and lip cancer. One potential model is:

Here we use $\log(E_i)$ to represent the log of the expected number of cases. Therefore, η_i is interpreted as the log of the *relative risk*. Let's draw a structure diagram of the model.

Recall, when we conduct Bayesian inference, we get posterior distributions of all of our paramters in our model through:

What parameters are we estimating here? Recall that for each paramter we want a posterior for we need to provide a prior distribution for.

Typically, to find the posterior distribution we cannot find a closed form solution to the actual posterior. Therefore, researchers either simulate from the posterior, or they estimate the posterior. Today, we're going to use the later technique. If you take a Bayesian course in the future they likely will spend a lot of time on Markov Chain Monte Carlo (MCMC) techniques for simulating from the posterior. While this is typically really smart to do, it also can be computationally intensive and takes some time to talk about what's actually going on. We're going to side step this today and instead estimate the posterior using a technique called integrated nested Laplace approximation (INLA). The basics are this:

The really nice thing about this, I think is it's *sort of* straight forward to code.

```r
library(INLA)
library(SpatialEpi)
data("scotland")
o <- scotland$data$cases
X <- scotland$data$AFF
E <- scotland$data$expected
LipCancerData <- data.frame(o=o,X=X,E=E,id=seq(1,length(o)))

formula <- o ~ 1 + X + f(id,model = "iid",
                         hyper=list(prec=list(prior="loggamma",
                                              param=c(1,0.00001))))

lip.cancer.model <- inla(formula,family="poisson",
                         data=LipCancerData, offset=log(E),
                         control.predictor=list(compute=TRUE),
                         control.fixed=list(mean=0, prec=0.00001,
                                            mean.intercept=0,
                                            prec.intercept=0.00001))
```

Let’s go through this and then look at some of the printouts.

```
summary(lip.cancer.model)

round(lip.cancer.model$summary.fixed,3)
round(lip.cancer.model$summary.hyperpar,3)

scotland_fitted <- scotland_sf
scotland_fitted$county_effect <- lip.cancer.model$summary.random$id$mean
ggplot() +
  geom_sf(data = scotland_fitted, aes(fill= county_effect))
```

Let’s say instead we want a model where we assume the effect of sunlight differs per each county. We can write:

Our code then becomes:

```
LipCancerData2 <- data.frame(o=o,X=X,E=E,id=seq(1,length(o)),id2=seq(1,length(o)))

formula2 <- o ~ 1 + X + f(id,model = "iid",
                         hyper=list(prec=list(prior="loggamma",
                                              param=c(1,0.00001)))) +
                       f(id2,X,model="iid",
                          hyper=list(prec=list(prior="loggamma",
                                              param=c(1,0.00001))))

lip.cancer.model2 <- inla(formula2,family="poisson",
                         data=LipCancerData2, offset=log(E),
                         control.predictor=list(compute=TRUE),
                         control.fixed=list(mean=0, prec=0.00001,
                                            mean.intercept=0,
                                            prec.intercept=0.00001))
```

Let’s go through the R code I posted and see if we can figure out what’s going on.

```
library(tidyverse)
library(faraway)
library(INLA)
library("spdep")
library("lme4")
library("spatstat")
library("sp")
library("maptools")
library("lattice")




data(penicillin)
summary(penicillin)
```

```
##  treat    blend       yield
##  A:5   Blend1:4   Min.   :77
##  B:5   Blend2:4   1st Qu.:81
##  C:5   Blend3:4   Median :87
##  D:5   Blend4:4   Mean   :86
##        Blend5:4   3rd Qu.:89
##                   Max.   :97
```

```
#Prior for Precision
prec.prior <- list(prec = list(param = c(0.001, 0.001)))

penicillin$treat <- relevel(penicillin$treat, "D")


inla.pen <- inla(yield ~ 1 + treat + f(blend, model = "iid",
                                       hyper = prec.prior),
                 data = penicillin, control.predictor = list(compute = TRUE))
summary(inla.pen)
```

```
##
## Call:
##    c("inla.core(formula = formula, family = family, contrasts = contrasts,
##    ", " data = data, quantiles = quantiles, E = E, offset = offset, ", "
##    scale = scale, weights = weights, Ntrials = Ntrials, strata = strata,
##    ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose =
##    verbose, ", " lincomb = lincomb, selection = selection, control.compute
##    = control.compute, ", " control.predictor = control.predictor,
##    control.family = control.family, ", " control.inla = control.inla,
##    control.fixed = control.fixed, ", " control.mode = control.mode,
##    control.expert = control.expert, ", " control.hazard = control.hazard,
##    control.lincomb = control.lincomb, ", " control.update =
##    control.update, control.lp.scale = control.lp.scale, ", "
##    control.pardiso = control.pardiso, only.hyperparam = only.hyperparam,
##    ", " inla.call = inla.call, inla.arg = inla.arg, num.threads =
##    num.threads, ", " keep = keep, working.directory = working.directory,
##    silent = silent, ", " inla.mode = inla.mode, safe = FALSE, debug =
##    debug, .parent.frame = .parent.frame)" )
## Time used:
##     Pre = 0.485, Running = 0.295, Post = 0.179, Total = 0.959
## Fixed effects:
##               mean    sd 0.025quant 0.5quant 0.975quant   mode kld
## (Intercept) 86.000 2.745     80.552   86.000     91.448 86.000   0
## treatA      -1.992 2.820     -7.592   -1.993      3.611 -1.992   0
## treatB      -0.996 2.820     -6.597   -0.996      4.607 -0.996   0
## treatC       2.988 2.820     -2.616    2.989      8.587  2.989   0
##
```

```
## Random effects:
##   Name     Model
##     blend IID model
##
## Model hyperparameters:
##                                          mean    sd 0.025quant 0.5quant
## Precision for the Gaussian observations 0.064 0.025      0.028    0.059
## Precision for blend                     0.140 0.190      0.012    0.084
##                                         0.975quant  mode
## Precision for the Gaussian observations      0.124 0.052
## Precision for blend                          0.619 0.031
##
## Marginal log-Likelihood:  -84.10
##  is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```
test<-inla.pen$marginals.fixed[[3]]
```

Let's try to write out the model that we are fitting:

```
data("sleepstudy")

sleepstudy$Reaction <- sleepstudy$Reaction / 1000


inla.sleep <- inla(Reaction ~ 1 + Days + f(Subject, model = "iid"),
                   data = sleepstudy)


#sleepstudy %>% ggplot(aes(x=Days,y=Reaction))+geom_point()+
#  stat_smooth(method="lm",se=F)


#sleepstudy %>% ggplot(aes(x=Days,y=Reaction))+geom_point()+
#  stat_smooth(method="lm",se=F)+facet_wrap(~Subject)


inla.sleep.w <- inla(Reaction ~ 1 + f(Subject, Days, model = "iid"),
                     data = sleepstudy, control.predictor = list(compute = TRUE))
summary(inla.sleep.w)
```

```
##
## Call:
##    c("inla.core(formula = formula, family = family, contrasts = contrasts,
##    ", " data = data, quantiles = quantiles, E = E, offset = offset, ", "
##    scale = scale, weights = weights, Ntrials = Ntrials, strata = strata,
##    ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose =
##    verbose, ", " lincomb = lincomb, selection = selection, control.compute
##    = control.compute, ", " control.predictor = control.predictor,
##    control.family = control.family, ", " control.inla = control.inla,
##    control.fixed = control.fixed, ", " control.mode = control.mode,
##    control.expert = control.expert, ", " control.hazard = control.hazard,
##    control.lincomb = control.lincomb, ", " control.update =
##    control.update, control.lp.scale = control.lp.scale, ", "
##    control.pardiso = control.pardiso, only.hyperparam = only.hyperparam,
##    ", " inla.call = inla.call, inla.arg = inla.arg, num.threads =
##    num.threads, ", " keep = keep, working.directory = working.directory,
##    silent = silent, ", " inla.mode = inla.mode, safe = FALSE, debug =
##    debug, .parent.frame = .parent.frame)" )
## Time used:
##     Pre = 0.382, Running = 0.331, Post = 0.087, Total = 0.8
## Fixed effects:
##              mean    sd 0.025quant 0.5quant 0.975quant  mode kld
## (Intercept) 0.255 0.004      0.247    0.255      0.263 0.255   0
##
## Random effects:
##   Name     Model
##     Subject IID model
##
## Model hyperparameters:
##                                             mean      sd 0.025quant 0.5quant
## Precision for the Gaussian observations 1198.80  133.41     954.83  1192.41
## Precision for Subject                   7361.87 2494.99    3562.09  6997.44
##                                         0.975quant    mode
## Precision for the Gaussian observations    1479.39 1181.66
## Precision for Subject                     13263.29 6326.02
##
## Marginal log-Likelihood:  336.13
##  is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

How do these models differ? When would you want to use the second model over the first?

Let's step up our modeling a bit

```r
nyc.stops <- read.table(file = "frisk_with_noise.dat", skip = 6,
                        header = TRUE)

# Add labels to factors
nyc.stops$eth <- as.factor(nyc.stops$eth)
```

```r
levels(nyc.stops$eth) <- c("black", "hispanic", "white")
nyc.stops$eth <- relevel(nyc.stops$eth, "white")

nyc.stops$crime <- as.factor(nyc.stops$crime)
levels(nyc.stops$crime) <- c("violent", "weapons", "property", "drug")


## ----------------------------------------------------------------------------
nyc.stops.agg <- aggregate(cbind(stops, past.arrests, pop) ~ precinct + eth,
                           data = nyc.stops, sum)

# Population is summed 4 times
nyc.stops.agg$pop <- nyc.stops.agg$pop / 4


## ----------------------------------------------------------------------------
nyc.inla <- inla(stops ~ eth + f(precinct, model = "iid"),
                 data = nyc.stops.agg, offset = log((15 / 12) * past.arrests),
                 family = "poisson")




## ----------------------------------------------------------------------------
# Ethnicity precinct index
nyc.stops.agg$ID <- 1:nrow(nyc.stops.agg)
nyc.inla2 <- inla(stops ~ eth + f(precinct, model = "iid") +
                    f(ID, model = "iid"),
                  data = nyc.stops.agg, offset = log((15/12) * past.arrests),
                  family = "poisson")
```

So, we can do this, but how do we know the models are any good? Other than conceptualizing the problem, perhaps we want something similar to AIC or BIC to help us out.

Two possible methods are posterior predictive checks and DIC.

The general idea behind posterior predictive checks (or P values) is:

We can sample from the posterior doing something like:

```r
nyc.inla <- inla(stops ~ eth + f(precinct, model = "iid"),
                 data = nyc.stops.agg, offset = log((15 / 12) * past.arrests),
                 family = "poisson",control.predictor=list(link=1, compute=TRUE),
                 control.compute=list(return.marginals.predictor=TRUE))

nyc.inla2 <- inla(stops ~ eth + f(precinct, model = "iid") +
                    f(ID, model = "iid"),
                  data = nyc.stops.agg, offset = log((15/12) * past.arrests),
                  family = "poisson",control.compute=list(dic=TRUE,config = TRUE,
                                                          return.marginals.predictor=TRUE))



predicted.p.value <- c()
n <- nrow(nyc.stops.agg)
for(i in 1:n){
  predicted.p.value[i] <- inla.pmarginal(q=nyc.stops.agg$stops[i],marginal=nyc.inla$marginals.fitted.val
}
```

We can take a look at:

```
hist(predicted.p.value)
```

Here we see some issues. We can see where we aren't fitting the data well:

```
plot(nyc.stops.agg$stops,nyc.inla$summary.fitted.values$mean,
     xlab="Observed Values", ylab="Mean Post. Pred. Distr.")
```

We can repeat this with the more complex model.

To compare two models we can use DIC or deviance information criterion

```
nyc.inla <- inla(stops ~ eth + f(precinct, model = "iid"),
                 data = nyc.stops.agg, offset = log((15 / 12) * past.arrests),
                 family = "poisson",control.compute = list(dic=TRUE,config = TRUE))


nyc.inla2 <- inla(stops ~ eth + f(precinct, model = "iid") +
                    f(ID, model = "iid"),
                  data = nyc.stops.agg, offset = log((15/12) * past.arrests),
                  family = "poisson",control.compute=list(dic=TRUE,config = TRUE))



nyc.inla$dic$dic
```

```
## [1] 3857.836
```

```
nyc.inla2$dic$dic
```

```
## [1] 2156.08
```

Here we see for a variety of reasons the second model would be preferred to the first.