Lecture 11 - Key

Hierarchical Regression

Recall the hierarchical normal model we used previously.

This model allowed a different mean for each group (school), denoted θ_j.

Now returning to the motivating example, test scores within a school. Suppose there are other factors that affect test scores at the school, specifically how about the socio-economic standing of families in that school district.

We can now write the model as:

where $\tilde{x}_J$ is a vector of socio-economic information about school j, this could also include 1 to account for an intercept.

What priors to we need to fit this model?

Similar to the hierarchical means model, we can obtain full conditional distributions for σ^2, Σ_0, $\tilde{\theta}$, and $\tilde{\mu}$. This allows us to use a Gibbs sampler to draw samples from the joint posterior distribution.

Exercise

1. Write out the model for a hierarchical regression setting. To keep it simple assume we are fitting a different intercept and slope (associated with square footage) for the different zip codes.

The following priors are specified for this setting.

Extract Data

```r
library(readr)
library(dplyr)
library(tidyr)
seattle <- read_csv('http://www.math.montana.edu/ahoegh/teaching/stat532/data/SeattleHousing.csv')
set.seed(11122018)

num.zips <- 10
num.houses <- 20
keep.zips <- sample(unique(seattle$zipcode), num.zips)

seattle.filter <- seattle %>% filter(zipcode %in% keep.zips) %>% group_by(zipcode) %>%
  sample_n(num.houses) %>% arrange(zipcode) %>% select(price, sqft_living, zipcode, id) %>%
  ungroup() %>% mutate(zipcode = as.factor(zipcode), house.num = rep(1:num.houses, num.zips))

price.wide <- seattle.filter %>% select(zipcode, price,house.num) %>%
  spread(key = zipcode, value = price) %>% select(-house.num)

size.wide <- seattle.filter %>% select(zipcode, sqft_living,house.num) %>%
  spread(key = zipcode, value = sqft_living) %>% select(-house.num)


library(ggplot2)
ggplot(data = seattle.filter, aes(y = price, x = sqft_living)) + geom_point() +
  geom_smooth() + facet_wrap(~zipcode, nrow = 5, ncol = 2)
```

2. Compare and contrast the following two models

```
summary(lm(price~ sqft_living , data = seattle.filter))
```

```
library(rjags)
modelstring <- "
model {
    # Model
    for (zip in 1:num.zips) {
        for (house in 1:num.houses) {
            mu[house, zip] <- alpha + beta * (x[house,zip]);
            price.wide[house,zip] ~ dnorm(mu[house,zip], tau.price)
        }
    }
    # Priors
    alpha   ~ dnorm(0, 1/1e16);
    beta    ~ dnorm(0, 1/1e16);
    tau.price     ~ dgamma(.0001, .0001);

    # Transformations
    sigma.price    <- 1.0/sqrt(tau.price);
}
"
writeLines(modelstring, "model.txt")

Data <- list(
    num.zips =   num.zips,
    num.houses =    num.houses,
    price.wide =   price.wide,
    x = size.wide)
mod1 <- jags.model("model.txt", data=Data, n.chains=4, n.adapt=1000)

codaSamples = coda.samples( mod1 , variable.names=c("alpha","beta", 'sigma.price') ,
                            n.iter=10000)

summary(codaSamples)
```

3. Now again, compare and contrast the following two models.

```
summary(lm(price~ sqft_living + zipcode - 1, data = seattle.filter))
```

```
library(rjags)
modelstring <- "
model {
    # Model
    for (zip in 1:num.zips) {
        for (house in 1:num.houses) {
            mu[house, zip] <- alpha[zip] + beta * (x[house,zip]);
            price.wide[house,zip] ~ dnorm(mu[house,zip], tau.price)
        }
        alpha[zip] ~ dnorm(alpha.mu, alpha.tau);
    }
    # Priors
    alpha.mu   ~ dnorm(0, 1/1e16);
    beta    ~ dnorm(0, 1/1e16);
    tau.price     ~ dgamma(.0001, .0001);
    alpha.tau ~ dgamma(.00001, .00001);
    # Transformations
    alpha.sigma  <- 1.0/sqrt(alpha.tau);
    sigma.price    <- 1.0/sqrt(tau.price);
}
"
writeLines(modelstring, "model.txt")

Data <- list(
    num.zips =   num.zips,
    num.houses =    num.houses,
    price.wide =   price.wide,
    x = size.wide)
mod1 <- jags.model("model.txt", data=Data, n.chains=4, n.adapt=1000)

codaSamples = coda.samples( mod1 , variable.names=c("alpha","beta", 'sigma.price', 'alpha.mu') ,
                            n.iter=10000)

summary(codaSamples)
```