Lecture 11 - Key

Hierarchical Regression

Recall the hierarchical normal model we used previously.

$$\begin{aligned} p(y|\theta_j, \sigma^2) &= normal(\theta_j, \sigma^2) \text{ within-group model} \\ p(\theta_j|\mu, \tau) &= normal(\mu, \tau^2) \text{ between-group model} \end{aligned}$$

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:

$$\begin{aligned} p(y|\tilde{x}_j, \theta_j, \sigma^2) &= normal(\tilde{x}_j^T \tilde{\theta}_j, \sigma^2) \text{ within-group model} \\ p(\tilde{\theta}_j|\tilde{\mu}, \Sigma) &= MVN(\tilde{\mu}, \Sigma) \text{ between-group model} \end{aligned}$$

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?

$$\begin{aligned} \sigma^2 &\sim InvGamma(\nu_0/2, \nu_0\sigma_0^2/2) \\ \Sigma &\sim InvWishart(\eta_0/2, \eta_0\tau_0^2/2) \\ \tilde{\mu} &\sim MVN(\tilde{\mu}_0, \Lambda_0) \end{aligned}$$

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.

$$
\begin{aligned}
y_{ij} &\sim N(X_{ij}\widetilde{\theta}_i, \sigma^2) \\
\widetilde{\theta}_i &\sim N(\mu_0, \Sigma_0),
\end{aligned}
$$

where y_{ij} is the price of the j^{th} house in the i^{th} zip code, $\widetilde{\theta}_i$ is the vector of regression coefficients (slope, intercept) for zip code i.

The following priors are specified for this setting.

$$
\begin{aligned}
\sigma^2 &\sim InvGamma(\nu_0/2, \nu_0\sigma_0^2/2) \\
\Sigma &\sim InvWishart(\eta_0/2, \eta_0\tau_0^2/2) \\
\tilde{\mu} &\sim MVN(\tilde{\mu}_0, \Lambda_0)
\end{aligned}
$$

Extract Data

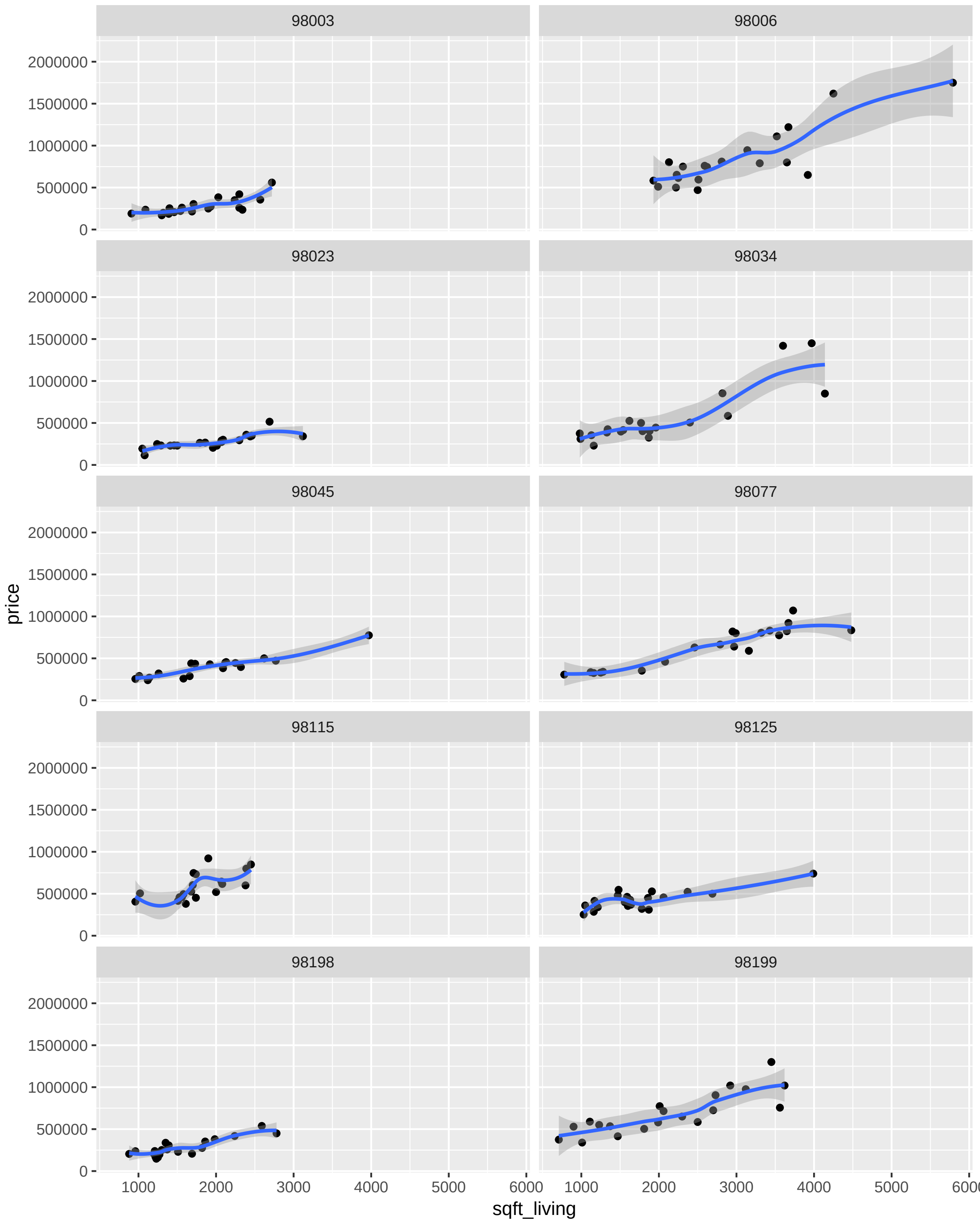
```
library(readr)
library(dplyr)
library(tidyr)
seattle <- read_csv('http://www.math.montana.edu/ahoegh/teaching/stat532/data/SeattleHousing.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_integer(),
##   price = col_double(),
##   bathrooms = col_double(),
##   floors = col_double(),
##   lat = col_double(),
##   long = col_double(),
##   mn_sold = col_character(),
##   day_sold = col_character()
## )

## See spec(...) for full column specifications.
```

```
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)
```

```r
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)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



98003
98006
98023
98034
98045
98077
98115
98125
98198
98199
price
2000000
1500000
1000000
500000
0
1000
2000
3000
4000
5000
6000
sqft_living

2. Compare and contrast the following two models

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

```
##
## Call:
## lm(formula = price ~ sqft_living, data = seattle.filter)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -425974 -123091  -17755   96398  566245
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -22966.8    31222.4  -0.736    0.463
## sqft_living    253.3       14.1  17.971   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 170100 on 198 degrees of freedom
## Multiple R-squared:  0.6199, Adjusted R-squared:  0.618
## F-statistic:   323 on 1 and 198 DF,  p-value: < 2.2e-16
```

```r
library(rjags)
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.0
```

```
## Loaded modules: basemod,bugs
```

```r
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)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 200
##    Unobserved stochastic nodes: 3
##    Total graph size: 698
##
## Initializing model
```

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

summary(codaSamples)
```

```
##
## Iterations = 1:10000
## Thinning interval = 1
## Number of chains = 4
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                 Mean       SD  Naive SE Time-series SE
## alpha       -22797.5 33037.88 165.18938       576.3691
## beta           253.3    14.82   0.07411         0.2613
## sigma.price 170845.4  8722.99  43.61493        45.6073
##
## 2. Quantiles for each variable:
##
##                 2.5%      25%      50%    75%    97.5%
## alpha       -84613.8 -44235.2 -23129.7  -1484  39292.9
## beta           225.3    243.7    253.4    263    281.3
## sigma.price 154936.4 164822.3 170445.3 176414 188763.6
```

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

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

```
##
## Call:
## lm(formula = price ~ sqft_living + zipcode - 1, data = seattle.filter)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -394795  -66941   -1669   61635  515198
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## sqft_living       221.59      11.47  19.314  < 2e-16 ***
## zipcode98003 -118981.48   33955.25  -3.504 0.000572 ***
## zipcode98006  176673.63   43496.25   4.062 7.13e-05 ***
## zipcode98023 -148893.88   34868.56  -4.270 3.09e-05 ***
## zipcode98034  107088.70   35754.98   2.995 0.003111 **
## zipcode98045  -27545.19   34842.92  -0.791 0.430195
## zipcode98077   49943.30   40546.77   1.232 0.219574
## zipcode98115  190047.30   33769.68   5.628 6.51e-08 ***
## zipcode98125   36915.81   33756.66   1.094 0.275528
## zipcode98198  -66314.83   32433.76  -2.045 0.042279 *
## zipcode98199  220501.76   36468.17   6.046 7.76e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 121100 on 189 degrees of freedom
## Multiple R-squared:  0.9567, Adjusted R-squared:  0.9542
## F-statistic: 379.7 on 11 and 189 DF,  p-value: < 2.2e-16
```

```r
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")
```

```r
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)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 200
##    Unobserved stochastic nodes: 14
##    Total graph size: 764
##
## Initializing model
```

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

summary(codaSamples)
```

```
##
## Iterations = 1:10000
## Thinning interval = 1
## Number of chains = 4
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                  Mean       SD  Naive SE Time-series SE
## alpha[1]    -115337.4 35048.06 175.24032       506.4106
## alpha[2]     165489.5 45629.58 228.14789       815.1097
## alpha[3]    -144104.4 35991.71 179.95856       546.5192
## alpha[4]     100680.3 36890.82 184.45408       578.5034
## alpha[5]     -28215.2 36135.17 180.67586       535.8567
## alpha[6]      44708.2 42435.14 212.17571       735.1173
## alpha[7]     180191.4 34915.47 174.57733       536.9964
## alpha[8]      33917.9 34602.07 173.01037       510.3442
## alpha[9]     -64218.0 33400.68 167.00338       450.1471
## alpha[10]    208989.4 38159.23 190.79616       606.9420
## alpha.mu      38143.8 53858.78 269.29389       632.7127
## beta            223.4    12.38   0.06192         0.2648
## sigma.price  121648.4  6429.06  32.14532        37.8737
##
## 2. Quantiles for each variable:
##
##                  2.5%       25%       50%       75%    97.5%
## alpha[1]    -181922.6 -138554.4 -115782.6  -92673.9 -47558.8
## alpha[2]      80614.2  135588.6  164957.8  194500.8 252429.9
## alpha[3]    -212767.3 -167781.5 -144415.8 -120925.3 -75162.7
## alpha[4]      30378.1   76559.9  100249.5  124475.3 171490.8
## alpha[5]     -96490.5  -51979.1  -28745.9   -4927.4  41616.3
```

```
## alpha[6]      -35435.1   17304.3   44096.0   71241.1 125817.9
## alpha[7]      113631.0  157173.1  179797.3  203013.2 247144.8
## alpha[8]      -32120.0   11228.9   33859.1   56025.5 100412.3
## alpha[9]     -128131.1  -86369.8  -64394.3  -42602.1    480.1
## alpha[10]     136748.1  184094.1  208649.7  233319.3 282297.4
## alpha.mu      -66455.5    4085.9   37533.9   71643.1 144783.3
## beta             200.1     215.8     223.7     231.5    246.2
## sigma.price  110072.7  117295.4  121307.4  125652.3 134930.4
```