Bayesian Multilevel Modeling

Contiunuing with our exercise from last class. We want to build a Bayesian multi-level model. First, Load the following libraries

```
library(tidyverse)
library(lme4)
library(rstan)
library(cowplot)
library(ggplot2)
library(lubridate)
library(ggridges)
library(kableExtra)
```

Then load the data like before.

```
SalesTrans =
  read_csv("C:/Users/ellen/Documents/UH/Fall 2020/Class Materials/Section II/Class 1/Data/Sales.csv")
Location =
  read_csv("C:/Users/ellen/Documents/UH/Fall 2020/Class Materials/Section II/Class 1/Data/Location.csv"
MerGroup =
  read_csv("C:/Users/ellen/Documents/UH/Fall 2020/Class Materials/Section II/Class 1/Data/MerGroup.csv"
SalesTrans = SalesTrans %>% inner_join(Location, by = "LocationID")
SalesTrans = SalesTrans %>% inner_join(MerGroup, by = "MerGroup")
LocationID = as.factor(SalesTrans$LocationID)
SalesTrans$ProductID = as.factor(SalesTrans$ProductID)
SalesTrans$Description = as.factor(SalesTrans$Description)
SalesTrans$MerGroup = as.factor(SalesTrans$MerGroup)

# breaking out Q4 to simplify exercise

SalesTrans$Qtr = quarter(SalesTrans$Tdate)
SalesTrans = filter(SalesTrans, Qtr == 4)
```

```
SalesTransSummary = SalesTrans %>%
  group_by(Description, MerGroup, MfgPromo, Wk ) %>%
  summarise(Volume = n(), TotSales = sum(Amount) )
```

Now, we'll run lmer models *(like before)* to get priors:

```
LMERVarIntSlpMod2L = lmer(TotSales ~ Wk + ( Wk | Description) + (Wk | MerGroup),
                          data = SalesTransSummary)
```

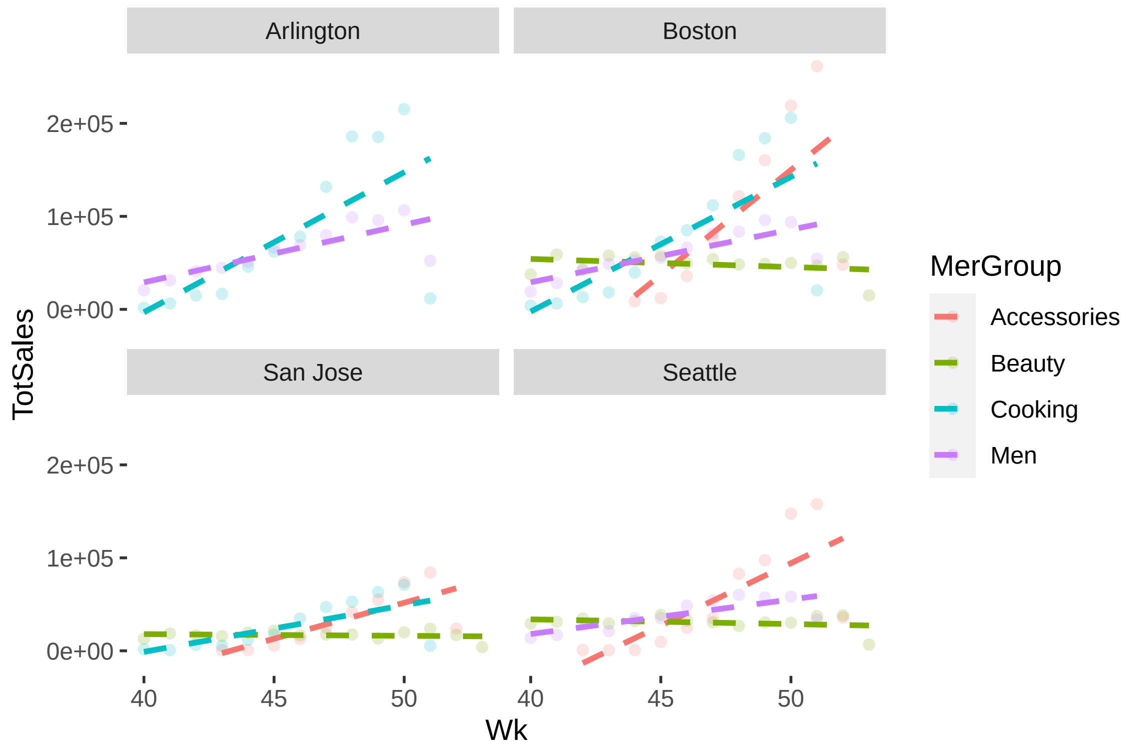
And pull the coefficients like before:

```
# Here, we're getting the effects (intercept and slope) for the Description level

d1 = ranef(LMERVarIntSlpMod2L)$Description %>%
  rownames_to_column("Description") %>%
  select(Description, "I1" = "(Intercept)", "S1" = "Wk")
```

```r
# Here, we're getting the effects (intercept and slope) for the MerGroup level

d2 = ranef(LMERVarIntSlpMod2L)$MerGroup %>%
  rownames_to_column("MerGroup") %>%
  select(MerGroup, "I2" = "(Intercept)", "S2" = "Wk")

# then, we get the fixed effects

I = fixef(LMERVarIntSlpMod2L)[1]
S = fixef(LMERVarIntSlpMod2L)[2]

# and we add them up to get the parameter values:

Coef = crossing(d1, d2) %>% mutate(Intercept = I + I1 + I2, Slope = S + S1 + S2)
```

Let's filter it down to 3 locations and 4 Mergroups so it's easier to see, and generate lm models for baseline:

```r
SalesSummarySub = filter(SalesTransSummary,
                         Description %in% c("Arlington", "Boston", "San Jose", "Seattle"),
                         MerGroup %in% c("Accessories", "Beauty", "Cooking", "Men" ))

p =  ggplot(SalesSummarySub, aes(Wk, TotSales, color = MerGroup)) +
  geom_point(alpha = .2) +
  geom_smooth(method = "lm", se = F, alpha = .05, linetype = "dashed", alpha = .5) +
  facet_wrap(~Description) +
  theme(panel.background = element_rect(fill = "white"))
p
```



Now, we'll build a bayesian model. Notice that we are setting up to intercept parameters *(vector[J] alpha vector[K] alpha2)*, and slope *(vector[J] beta vector[K] beta2)* at two levels *(Description and Merch Groups)*.

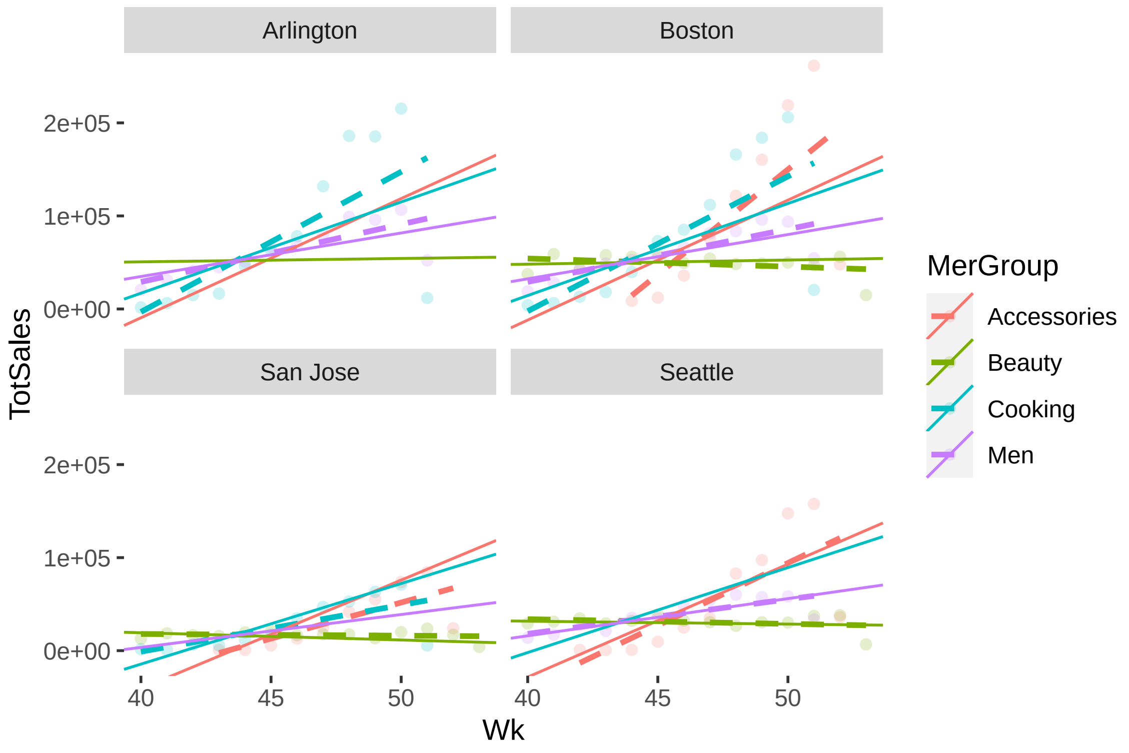
Also notice that all these parameters are modeled separately.

```
# build stan model
stanMod <- '

data {

  int<lower=0> N;
  vector[N] x; // Wk
  vector[N] y; // TotSales

  int<lower=0> J; // Description
  int Description[N];

  int<lower=0> K; // MerGroup
  int MerGroup[N];

  real p_alpha;
  real p_alpha2;

  real<lower=0> p_alphaSigma;
  real<lower=0> p_alphaSigma2;

  real p_beta;
  real p_beta2;

  real<lower=0> p_betaSigma;
  real<lower=0> p_betaSigma2;

}

parameters {

  // random effects we found
  vector[J] alpha; // intercept for Description
  vector[K] alpha2; // intercept for MerGroup

  vector[J] beta; // slope for Description
  vector[K] beta2; // slope for MerGroup

  real<lower=0> sigma; // to control sigma of y_hat

}

transformed parameters {

  vector[N] y_hat;
  for(i in 1:N)
  y_hat[i]=alpha[Description[i]]+alpha2[MerGroup[i]]+(beta[Description[i]]*x[i])+(beta2[MerGroup[i]]*x[
}

model {

  target += normal_lpdf(alpha | p_alpha, p_alphaSigma);
  target += normal_lpdf(alpha2 | p_alpha2, p_alphaSigma2);

  target += normal_lpdf(beta | p_beta, p_betaSigma);
  target += normal_lpdf(beta2 | p_beta2, p_betaSigma2);

  target += normal_lpdf(sigma | 50, 50);

  // y_hat: our prediction
  target += normal_lpdf(y | y_hat, sigma);
}
'
#range(d1$I1)
# [1] -32944.63  32401.78

fit <- stan(model_code = stanMod,  data = list(
  N = nrow(SalesTransSummary),
  y = SalesTransSummary$TotSales,
  x = SalesTransSummary$Wk,
  J = length(unique(SalesTransSummary$Description)),
  K = length(unique(SalesTransSummary$MerGroup)),
  Description = as.numeric(SalesTransSummary$Description),
  MerGroup = as.numeric(SalesTransSummary$MerGroup),

  p_alpha = median(d1$I1), # Description intercept
  p_alpha2 = median(d2$I2), # MerGroup intercept

  #
  p_alphaSigma = 200,
  p_alphaSigma2 = 200,

  p_beta = median(d1$S1), # Description slope (lmer model)
  p_beta2 = median(d2$S2), # MerGroup slope (lmer model)

  p_betaSigma = 50,
  p_betaSigma2 = 50
), refresh = 0)


# Extract coefficients from stan model
sumFit <- data.frame(summary(fit))

# Description
Intercept1 <- summary(fit, pars = c("alpha"), probs = c(0.1, 0.9))$summary
# MerGroup
Intercept2 <- summary(fit, pars = c("alpha2"), probs = c(0.1, 0.9))$summary

# Description slope
Slope1 <- summary(fit, pars = c("beta"), probs = c(0.1, 0.9))$summary
# MerGroup slope
Slope2 <- summary(fit, pars = c("beta2"), probs = c(0.1, 0.9))$summary

# random effects for each Description
CoefMapD <- data.frame(Description = unique(Coef$Description),
                        DIntercept = Intercept1[,1], DSlope = Slope1[,1])
# random effects for each MerGroup
CoefMapM <- data.frame(MerGroup = unique(Coef$MerGroup), MIntercept = Intercept2[,1], MSlope = Slope2[,

# add all intercepts and all slopes cross each row
CoefMapB = crossing(CoefMapD, CoefMapM) %>%
  mutate(Intercept = DIntercept + MIntercept, Slope = DSlope + MSlope)

CoefMapBSub <- filter(CoefMapB,
                      Description %in% c('Los Angeles', "Arlington", "San Francisco", "San Jose", "Bost
                      MerGroup %in% c("Accessories", "Cooking", "Women", "Shoes"))
```

Notice that we're just using a median value for the priors *(p_alpha = median(d1$I1))* which we got from lmer. If we had previous Bayesian models, we would probably use the posterior parameters for our priors. We could also "squeeze" the sigmas and force groups to pool around these priors *(more on that laster)*.

Taking a look at the models on a plot:

```
CoefSub = filter(Coef,
                         Description %in% c("Arlington", "Boston", "San Jose", "Seattle"),
                         MerGroup %in% c("Accessories", "Beauty", "Cooking", "Men" ))
p2 = p + geom_abline(data = CoefSub,
                     aes(intercept = CoefSub$Intercept, slope =  CoefSub$Slope, color = MerGroup))
p2
```



Note that our Bayesian model gives us parameter *effects* for every location and merch group, even where no data exists *(e.g., Arlington data had not Accessories or Beauty products)*. This is a BIG deal - how would you forecast Accessories for Arlington if Nordies was planning on carrying that group next year? We can do this becuase the model quantifies the *EFFECTS* of each location and the *EFFECTS* of each merch group. Then we just add the effect to the base to get the prosterior for a location, merch group of interest.