

Varying Slopes Models

General Principles

To model the relationship between predictor variables and a dependent variable while allowing for varying effects across groups or clusters, we use a *varying slopes* model.

This approach is useful when we expect the relationship between predictors and the dependent variable to differ across groups (e.g., different slopes for different subjects, locations, or time periods). This allows every unit in the data to have its own unique response to any treatment, exposure, or event, while also improving estimates via pooling.

Considerations

Note

- We have the same considerations as for [12. Varying intercepts](#).
- The idea is pretty similar to categorical models, where a slope is specified for each category. However, here, we also estimate relationships between different groups. This leads to a different mathematical approach, as to model these relationships between groups, we model a matrix of covariance .
- The covariance matrix requires a correlation matrix distribution which is modeled using an *LKJcorr* distribution that holds a parameter η . η is usually set to 2 to define a weakly informative prior that is skeptical of extreme correlations near -1 or 1 . When we use *LKJcorr*(1), the prior is flat over all valid correlation matrices. When the value is greater than 1, then extreme correlations are less likely.
- The Half-Cauchy distribution is used when modeling the covariance matrix to specify strictly positive values for the diagonal of the covariance matrix, ensuring positive variances.

Example

Below is an example code snippet demonstrating Bayesian regression with varying effects. This example is based on McElreath (2018).

Simulated data

Python (Raw)

```
from BI import bi
# Setup device-----
m = bi(platform='cpu')

# Import Data & Data Manipulation -----
# Import
from importlib.resources import files
data_path = files('BI.resources.data') / 'Sim data multivariatenormal.csv'
m.data(data_path, sep=',')

# Define model -----
def model(cafe, wait, N_cafes, afternoon):
    a = m.dist.normal(5, 2, name = 'a')
    b = m.dist.normal(-1, 0.5, name = 'b')
    sigma_cafe = m.dist.exponential(1, shape=(2,), name = 'sigma_cafe')
    sigma = m.dist.exponential(1, name = 'sigma')
    Rho = m.dist.lkj(2, 2, name = 'Rho')
    cov = jnp.outer(sigma_cafe, sigma_cafe) * Rho
    a_cafe_b_cafe = m.dist.multivariatenormal(jnp.stack([a, b]), cov, shape = [N_cafes], name = 'a_cafe_b_cafe')

    a_cafe, b_cafe = a_cafe_b_cafe[:, 0], a_cafe_b_cafe[:, 1]
    mu = a_cafe[cafe] + b_cafe[cafe] * afternoon
    m.dist.normal(mu, sigma, obs=wait)

# Run sampler -----
m.fit(model)
```

Python (Build in function)

```
from BI import bi
# Setup device-----
```

```

m = bi(platform='cpu')

# Import Data & Data Manipulation -----
# Import
from importlib.resources import files
data_path = files('BI.resources.data') / 'Sim data multivariatenormal.csv'
m.data(data_path, sep=',')

# Define model -----
def model(cafe, wait, N_cafes, afternoon):
    a = m.dist.normal(5, 2, name = 'a')
    b = m.dist.normal(-1, 0.5, name = 'b')
    sigma = m.dist.exponential( 1, name = 'sigma')

    varying_intercept, varying_slope = m.effects.varying_effects(
        N_group = N_cafes,
        group = cafe,
        global_intercept= a,
        global_slope= b,
        group_name = 'cafe')

    mu = varying_intercept + varying_slope* afternoon
    m.dist.normal(mu, sigma, obs=wait)

# Run sampler -----
m.fit(model)

```

R

```

library(BI)

# Setup platform-----
m=importbi(platform='cpu')

# Import data -----
m$data(paste(system.file(package = "BI"),"/data/Sim data multivariatenormal.csv", sep = ''),
m$data_to_model(list('cafe', 'wait', 'afternoon'))

# Define model -----
model <- function(cafe, afternoon, wait, N_cafes = as.integer(20) ){

```

```

a = bi.dist.normal(5, 2, name = 'a')
b = bi.dist.normal(-1, 0.5, name = 'b')
sigma_cafe = bi.dist.exponential(1, shape= c(2), name = 'sigma_cafe')
sigma = bi.dist.exponential( 1, name = 'sigma')
Rho = bi.dist.lkj(as.integer(2), as.integer(2), name = 'Rho')
cov = jnp$outer(sigma_cafe, sigma_cafe) * Rho

a_cafe_b_cafe = bi.dist.multivariatenormal(
  jnp$squeeze(jnp$stack(list(a, b))),
  cov, shape = c(N_cafes), name = 'a_cafe')

a_cafe = a_cafe_b_cafe[, 0]
b_cafe = a_cafe_b_cafe[, 1]

mu = a_cafe[cafe] + b_cafe[cafe] * afternoon

m$normal(mu, sigma, obs=wait)
}

# Run MCMC -----
m$fit(model) # Optimize model parameters through MCMC sampling

# Summary -----
m$summary() # Get posterior distribution

```

Mathematical Details

Centered parameterization

We can express the Bayesian regression model accounting for prior distributions as follows:

$$Y_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha_{k(i)} + \beta_{k(i)} X_i$$

The varying intercepts (α_k) and slopes (β_k) are modeled using a *Multivariate Normal distribution*:

$$\begin{pmatrix} \alpha_k \\ \beta_k \end{pmatrix} \sim \text{MultivariateNormal} \left(\begin{pmatrix} \bar{\alpha} \\ \bar{\beta} \end{pmatrix}, \text{diag}(\varsigma) \Omega \text{diag}(\varsigma) \right)$$

$$\begin{aligned}
\bar{\alpha} &\sim \text{Normal}(0, 1) \\
\bar{\beta} &\sim \text{Normal}(0, 1) \\
\varsigma &\sim \text{Exponential}(1) \\
\Omega &\sim \text{LKJ}(\eta)
\end{aligned}$$

Where:

- $\begin{pmatrix} \bar{\alpha} \\ \bar{\beta} \end{pmatrix}$ is a vector composed from concatenating a parameter for the global intercept and a parameter vector of the global slopes.
- ς is a vector giving the standard deviation of the random effects for the intercept and slopes across groups.
- Ω is the correlation matrix.

Non-centered parameterization

For computational reasons, it is often better to implement a non-centered parameterization that is equivalent to the *Multivariate Normal distribution* approach:

$$\begin{pmatrix} \alpha_k \\ \beta_k \end{pmatrix} = \begin{pmatrix} \bar{\alpha} \\ \bar{\beta} \end{pmatrix} + \varsigma \circ \left(L \cdot \begin{pmatrix} \hat{\alpha}_k \\ \hat{\beta}_k \end{pmatrix} \right)$$

$$\begin{aligned}
\bar{\alpha} &\sim \text{Normal}(0, 1) \\
\bar{\beta} &\sim \text{Normal}(0, 1) \\
\varsigma &\sim \text{Exponential}(1) \\
L &\sim \text{LKJ Cholesky}(\eta)
\end{aligned}$$

$$\hat{\alpha}_k \sim \text{Exponential}(1)$$

$$\hat{\beta}_k \sim \text{Exponential}(1)$$

- Where:
 - $\sigma_{\alpha} \sim \text{Exponential}(1)$ is the prior standard deviation among intercepts.
 - $\sigma_{\beta} \sim \text{Exponential}(1)$ is the prior standard deviation among slopes.
 - $L \sim \text{LKJcorr}(\eta)$ is the a cholesky factor of the correlation matrix matrix using the Cholesky Factor

Multivariate Model with One Random Slope for Each Variable

We can apply a multivariate model similarly to [Chapter 2](#). In this case, we apply the same principle, but with a covariance matrix with a dimension equal to the number of varying slopes we define. For example, if we want to generate random slopes for i observations in a model with two independent variables X_1 and X_2 , we can define the formula as follows:

$$Y_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \alpha_i + \beta_{k(i)}X_{1i} + \gamma_{k(i)}X_{2i}$$

$$\begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \sim \begin{pmatrix} \bar{\alpha} \\ \bar{\beta} \\ \bar{\gamma} \end{pmatrix} + \varsigma \circ \left(L \cdot \begin{pmatrix} \hat{\alpha}_k \\ \hat{\beta}_k \\ \hat{\gamma}_k \end{pmatrix} \right)$$

$$\bar{\alpha} \sim \text{Normal}(0, 1)$$

$$\bar{\beta} \sim \text{Normal}(0, 1)$$

$$\bar{\gamma} \sim \text{Normal}(0, 1)$$

$$\varsigma \sim \text{Exponential}(1)$$

$$L \sim \text{LKJ Cholesky}(2)$$

$$\hat{\alpha}_k \sim \text{Exponential}(1)$$

$$\hat{\beta}_k \sim \text{Exponential}(1)$$

$$\hat{\gamma}_k \sim \text{Exponential}(1)$$

Multivariate Random Slopes on a Single Variable

For more than two varying effects (k and g), we apply the same principle but with a covariance matrix for each varying effect that is summed to generate the varying intercept and slope. For example, if we want to generate random slopes for i actors and k groups, we can define the formula as follows:

$$Y_i \sim \text{Normal}(\mu_i, \sigma)$$

$$\mu_i = \left(\alpha_{k(i)} + \pi_{g(i)} \right) + \left(\beta_{k(i)} + \gamma_{g(i)} \right) X_i$$

$$\begin{pmatrix} \alpha_k \\ \beta_k \end{pmatrix} \sim \begin{pmatrix} \bar{\alpha} \\ \bar{\beta} \end{pmatrix} \varsigma \circ \left(L_k \cdot \begin{pmatrix} \hat{\alpha}_k \\ \hat{\beta}_k \end{pmatrix} \right)$$

$$\bar{\alpha} \sim \text{Normal}(0, 1)$$

$$\bar{\beta} \sim \text{Normal}(0, 1)$$

$$L_k \sim \text{LKJ Cholesky}(2)$$

$$\hat{\alpha} \sim \text{Exponential}(1)$$

$$\hat{\beta} \sim \text{Exponential}(1)$$

$$\begin{pmatrix} \pi_g \\ \gamma_g \end{pmatrix} \sim \begin{pmatrix} \bar{\pi} \\ \bar{\gamma} \end{pmatrix} + \varsigma \circ \left(L_g \cdot \begin{pmatrix} \hat{\pi}_g \\ \hat{\gamma}_g \end{pmatrix} \right)$$

$$\bar{\pi} \sim \text{Normal}(0, 1)$$

$$\bar{\gamma} \sim \text{Normal}(0, 1)$$

$$L_g \sim \text{LKJ Cholesky}(2)$$

$$\hat{\alpha} \sim \text{Exponential}(1)$$

$$\hat{\beta} \sim \text{Exponential}(1)$$

Notes

Note

- We can apply interaction terms similarly to [Chapter 3](#).
- We can apply categorical variables similarly to [Chapter 4](#).
- We can apply varying slopes with any distribution presented in previous chapters. Below is the formula and the code snippet for a Binomial multivariate model with an interaction between two independent variables X_1 and X_2 and multiple varying effects for each actor and each group.

$$Y_i \sim \text{Binomial}(n = 1, p_i)$$

$$\text{logit}(p_i) = (\alpha_{k(i)} + \pi_{g(i)}) + (\beta_{k(i)} + \gamma_{g(i)}) X_i$$

$$\begin{pmatrix} \alpha_k \\ \beta_k \end{pmatrix} \sim \begin{pmatrix} \bar{\alpha} \\ \bar{\beta} \end{pmatrix} \varsigma \circ \left(L_k \cdot \begin{pmatrix} \hat{\alpha}_k \\ \hat{\beta}_k \end{pmatrix} \right)$$

$$\bar{\alpha} \sim \text{Normal}(0, 1)$$

$$\bar{\beta} \sim \text{Normal}(0, 1)$$

$$L_k \sim \text{LKJ Cholesky}(2)$$

$$\hat{\alpha} \sim \text{Exponential}(1)$$

$$\hat{\beta} \sim \text{Exponential}(1)$$

$$\begin{pmatrix} \pi_g \\ \gamma_g \end{pmatrix} \sim \begin{pmatrix} \bar{\pi} \\ \bar{\gamma} \end{pmatrix} + \varsigma \circ \left(L_g \cdot \begin{pmatrix} \hat{\pi}_g \\ \hat{\gamma}_g \end{pmatrix} \right)$$

$$\bar{\pi} \sim \text{Normal}(0, 1)$$

$$\bar{\gamma} \sim \text{Normal}(0, 1)$$

$$L_g \sim \text{LKJ Cholesky}(2)$$

$$\hat{\alpha} \sim \text{Exponential}(1)$$

$$\hat{\beta} \sim \text{Exponential}(1)$$


```

from main import *
# Setup device-----
m = bi(platform='cpu')
# Import data
m.read_csv("../data/chimpanzees.csv", sep=";")
m.df["block_id"] = m.df.block
m.df["treatment"] = 1 + m.df.prosoc_left + 2 * m.df.condition
m.data_to_model(['pulled_left', 'treatment', 'actor', 'block_id'])

def model(tid, actor, block_id, L=None, link=False):
    # fixed priors
    g = dist.normal(0, 1, name = 'g', shape = (4,))
    sigma_actor = dist.exponential(1, name = 'sigma_actor', shape = (4,))
    L_Rho_actor = dist.lkjcholesky(4, 2, name = "L_Rho_actor")
    sigma_block = dist.exponential(1, name = "sigma_block", shape = (4,))
    L_Rho_block = dist.lkjcholesky(4, 2, name = "L_Rho_block")

    # adaptive priors - non-centered
    z_actor = dist.normal(0, 1, name = "z_actor", shape = (4,7))
    z_block = dist.normal(0, 1, name = "z_block", shape = (4,3))
    alpha = deterministic(
        "alpha", ((sigma_actor[... , None] * L_Rho_actor) @ z_actor).T
    )
    beta = deterministic(
        "beta", ((sigma_block[... , None] * L_Rho_block) @ z_block).T
    )

    logit_p = g[tid] + alpha[actor, tid] + beta[block_id, tid]
    dist("L", dist.Binomial(logits=logit_p), obs=L)

    # compute ordinary correlation matrices from Cholesky factors
    if link:
        deterministic("Rho_actor", L_Rho_actor @ L_Rho_actor.T)
        deterministic("Rho_block", L_Rho_block @ L_Rho_block.T)
        deterministic("p", expit(logit_p))

# Run mcmc -----
m.fit(model)

# Summary -----
m.sampler.print_summary(0.89)

```

Reference(s)