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

Caution

- 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)

```python
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[:, 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)

```python
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

Formula

We model the relationship between the independent variable X and the outcome variable Y with varying intercepts (α) and varying slopes (β) for each group (k) using the following equation:

$$Y_{ik} = \alpha_k + \beta_k X_{ik} + \sigma$$

Where:

- Y_{ik} is the outcome variable for observation i in group k.
- X_{ik} is the independent variable for observation i in group k.
- α_k is the varying intercept for group k.

- β_k is the varying regression coefficient for group k.
- σ is the error term, assumed to be strictly positive.

Bayesian Model

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

$$
\begin{aligned}
Y_{ik} &\sim \text{Normal}(\mu_{ik}, \sigma) \\
\mu_{ik} &= \alpha_k + \beta_k X_{ik} \\
\alpha_k &\sim Normal(0, 1) \\
\beta_k &\sim Normal(0, 1) \\
\sigma &\sim Exponential(1)
\end{aligned}
$$

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

$$
\begin{pmatrix} \alpha_k \\ \beta_k \end{pmatrix} \sim \text{MultivariateNormal}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma_\alpha^2 & \rho_{\alpha\beta}\sigma_\alpha\sigma_\beta \\ \rho_{\alpha\beta}\sigma_\alpha\sigma_\beta & \sigma_\beta^2 \end{pmatrix} \right)
$$

Where:

- $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ is the prior for the average intercept.
- $\begin{pmatrix} \sigma_\alpha^2 & \rho_{\alpha\beta}\sigma_\alpha\sigma_\beta \\ \rho_{\alpha\beta}\sigma_\alpha\sigma_\beta & \sigma_\beta^2 \end{pmatrix}$ is the covariance matrix which specifies the variance and covariance of α_k and β_k,
- where:
  - σ_α^2 is the variance of α_k.
  - σ_β^2 is the variance of β_k.
  - $\rho_{\alpha\beta}\sigma_\alpha\sigma_\beta$ is the covariance between α_k and β_k.

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

$$
\begin{pmatrix} \alpha_k \\ \beta_k \end{pmatrix} \sim \begin{pmatrix} \sigma_\alpha \\ \sigma_\beta \end{pmatrix} \circ L \cdot \begin{pmatrix} \hat{\alpha}_k \\ \hat{\beta}_k \end{pmatrix}
$$

- Where:

- $\sigma_\alpha \sim$ Exponential(1) is the prior standard deviation among intercepts.
- $\sigma_\beta \sim$ Exponential(1) is the prior standard deviation among slopes.
- $L \sim$ LKJcorr(η) is the prior for the correlation matrix using the Cholesky Factor

The full non-centered version of the model is thus:

$$Y_{ik} \sim \text{Normal}(\mu_{ik}, \sigma)$$

$$\mu_{ik} = \alpha_k + \beta_k X_{ik}$$

$$\begin{pmatrix} \alpha_k \\ \beta_k \end{pmatrix} \sim \begin{pmatrix} \sigma_\alpha \\ \sigma_\beta \end{pmatrix} \circ L \cdot \begin{pmatrix} \hat{\alpha}_k \\ \hat{\beta}_k \end{pmatrix}$$

$$\alpha \sim \text{Normal}(0, 1)$$
$$\beta \sim \text{Normal}(0, 1)$$
$$\sigma_\alpha \sim \text{Exponential}(1)$$

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

$$L \sim \text{LKJcorr}(2)$$

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:

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

$$\mu_i = \alpha_i + \beta_{1i} X_{1i} + \beta_{2i} X_{2i}$$

$$\begin{pmatrix} \alpha_i \\ \beta_{1i} \\ \beta_{2i} \end{pmatrix} \sim \begin{pmatrix} \sigma_\alpha \\ \sigma_{\beta_1} \\ \sigma_{\beta_2} \end{pmatrix} \circ L \cdot \begin{pmatrix} \hat{\alpha}_i \\ \hat{\beta}_{1i} \\ \hat{\beta}_{2i} \end{pmatrix}$$

$$\sigma_\alpha \sim \text{Exponential}(1)$$
$$\sigma_{\beta_1} \sim \text{Exponential}(1)$$
$$\sigma_{\beta_2} \sim \text{Exponential}(1)$$

$$L \sim \text{LKJcorr}(2)$$

Multivariate Random Slopes on a Single Variable

For more than two varying effects, 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:

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

$$\mu_i = \alpha_i + \beta_i X_i$$
$$\alpha_i = \alpha + \alpha_{actor[i]} + \alpha_{group[i]}$$
$$\beta_i = \beta + \beta_{actor[i]} + \beta_{group[i]}$$

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

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

$$\begin{pmatrix} \alpha_{\text{actor}} \\ \beta_{\text{actor}} \end{pmatrix} \sim \begin{pmatrix} \sigma_{\alpha a} \\ \sigma_{\beta a} \end{pmatrix} \circ L_a \cdot \begin{pmatrix} \hat{\alpha}_{ka} \\ \hat{\beta}_{ka} \end{pmatrix}$$

$$\sigma_{\alpha a} \sim \text{Exponential}(1)$$
$$\sigma_{\beta a} \sim \text{Exponential}(1)$$

$$L_a \sim \text{LKJcorr}(2)$$

$$\begin{pmatrix} \alpha_{\text{group}} \\ \beta_{\text{group}} \end{pmatrix} \sim \begin{pmatrix} \sigma_{\alpha g} \\ \sigma_{\beta g} \end{pmatrix} \circ L_g \cdot \begin{pmatrix} \hat{\alpha}_{kg} \\ \hat{\beta}_{kg} \end{pmatrix}$$

$$\begin{aligned} \sigma_{\alpha g} &\sim \text{Exponential}(1) \\ \sigma_{\beta g} &\sim \text{Exponential}(1) \\ L_g &\sim LKJcorr(2) \end{aligned}$$

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.

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

$$\begin{aligned} \text{logit}(p_i) &= \alpha_i + \beta_{1i} X_{1i} + \beta_{2i} X_{1i} X_{2i} \\ \alpha_i &= \alpha + \alpha_{actor[i]} + \alpha_{group[i]} \\ \beta_{1i} &= \beta_1 + \beta_{1,actor[i]} + \beta_{1,group[i]} \\ \beta_{2i} &= \beta_2 + \beta_{2,actor[i]} + \beta_{2,group[i]} \end{aligned}$$

$$\begin{aligned} \alpha &\sim \text{Normal}(0, 1) \\ \beta_1 &\sim \text{Normal}(0, 1) \\ \beta_2 &\sim \text{Normal}(0, 1) \end{aligned}$$

$$\begin{pmatrix} \alpha_{\text{actor}} \\ \beta_{1,\text{actor}} \\ \beta_{2,\text{actor}} \end{pmatrix} \sim \begin{pmatrix} \sigma_{\alpha a} \\ \sigma_{\beta_1 a} \\ \sigma_{\beta_2 a} \end{pmatrix} \circ L_a \cdot \begin{pmatrix} \hat{\alpha}_{ka} \\ \hat{\beta}_{1,ka} \\ \hat{\beta}_{2,ka} \end{pmatrix}$$

$$\sigma_{\alpha a} \sim \text{Exponential}(1)$$

$$\sigma_{\beta_1 a} \sim \text{Exponential}(1)$$
$$\sigma_{\beta_2 a} \sim \text{Exponential}(1)$$
$$L_a \sim LKJcorr(2)$$

$$\begin{pmatrix} \alpha_{\text{group}} \\ \beta_{1,\text{group}} \\ \beta_{2,\text{group}} \end{pmatrix} \sim \begin{pmatrix} \sigma_{\alpha g} \\ \sigma_{\beta_1 g} \\ \sigma_{\beta_2 g} \end{pmatrix} \circ L_g \cdot \begin{pmatrix} \hat{\alpha}_{kg} \\ \hat{\beta}_{1,kg} \\ \hat{\beta}_{2,kg} \end{pmatrix}$$

$$\sigma_{\alpha g} \sim \text{Exponential}(1)$$
$$\sigma_{\beta_1 g} \sim \text{Exponential}(1)$$
$$\sigma_{\beta_2 g} \sim \text{Exponential}(1)$$
$$L_g \sim LKJcorr(2)$$

```
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)