Nested Varying Effects

General Principles

To model hierarchies where groups are themselves nested within larger units (e.g., students within classes, which are within schools), we use *Nested Varying Effects*. This allows the model to share information across different levels of the hierarchy, improving estimates through multilevel pooling.

Note

> **Note**
>
> - We have the same considerations as for Varying Intercepts and Varying Slopes.
> - In a nested model, a parameter at one level (e.g., a group-level intercept) becomes the mean for the distribution of parameters at a lower level (e.g., individual-level intercepts).
> - This structure is often represented using multiple indices: j for the group and $k(j)$ for the super-group (e.g., region) that group j belongs to.
> - To capture the correlation between multiple varying effects (e.g., intercept and slope) at any level of the hierarchy, we use a **Multivariate Normal (MVN)** distribution.
> - Non-centered parameterization is highly recommended for nested models to avoid "funnel" geometries that can hinder MCMC sampling.

Example

Below is an example of a nested model with both **varying intercepts** and **varying slopes**. We model the outcome y for individuals in `groups` nested within `regions`. The relationship between x and y varies at both levels.

Python (Centered)

```python
from BayesForge import bf
import jax.numpy as jnp
import numpy as np

# Setup device ------------------------------------------------
m = bf(platform='cpu')

# Load data - column args are mapped automatically from the DataFrame.
# N_groups and N_regions are dataset-specific; provide them as defaults.
data_path = m.load.sim_nested_effects(only_path=True)
m.data(data_path)

# Define model ------------------------------------------------
def model_nested(y, x, group_id, region_id, N_groups=20, N_regions=5):
    sigma = m.dist.exponential(1, name='sigma')

    # 1. Region level
    mu_global = jnp.stack([m.dist.normal(5, 2,  name='global_intercept'),
                           m.dist.normal(-1, 1, name='global_beta')])
    sigma_reg  = m.dist.exponential(1, shape=(2,), name='sigma_region')
    corr_reg   = m.dist.lkj(2, 2, name='corr_region')
    cov_reg    = jnp.diag(sigma_reg) @ corr_reg @ jnp.diag(sigma_reg)
    region_effects = m.dist.multivariate_normal(
        mu_global, cov_reg, shape=(N_regions,), name='region_effects'
    )

    # 2. Group level - parent mapping via JAX scatter (traceable under pmap)
    group_to_region = jnp.zeros(N_groups, dtype=jnp.int32).at[group_id].set(region_id)
    sigma_grp  = m.dist.exponential(1, shape=(2,), name='sigma_group')
    corr_grp   = m.dist.lkj(2, 2, name='corr_group')
    cov_grp    = jnp.diag(sigma_grp) @ corr_grp @ jnp.diag(sigma_grp)
    group_effects = m.dist.multivariate_normal(
        region_effects[group_to_region], cov_grp, name='group_effects'
    )

    mu_est = group_effects[group_id, 0] + group_effects[group_id, 1] * x
    m.dist.normal(mu_est, sigma, obs=y)

# Run sampler ------------------------------------------------
m.fit(model_nested, num_samples=1000, num_warmup=500, num_chains=1)
```

```
m.summary()
```

```
bf v 0.0.48 package loaded

E0526 16:17:20.745109 1063742 cuda_dnn.cc:523] Loaded runtime CuDNN library: 9.1.0 but source
E0526 16:17:20.746990 1063742 cuda_dnn.cc:523] Loaded runtime CuDNN library: 9.1.0 but source

jax.local_device_count 32

/home/sosa/.local/lib/python3.10/site-packages/jax/_src/ops/scatter.py:108: FutureWarning:

scatter inputs have incompatible types: cannot safely cast value from dtype=int64 to dtype=in

  0%|          | 0/1500 [00:00<?, ?it/s]warmup:   0%|          | 1/1500 [00:01<42:38,  1.71s/
/home/sosa/work/BF/BayesForge/Diagnostic/jax_diagnostics.py:214: RuntimeWarning:

invalid value encountered in scalar divide
```

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_ha
corr_group[0, 0]	1.00	0.00	1.00	1.00	0.00	0.00	3000.00	3000.00	NaN
corr_group[0, 1]	0.42	0.23	0.06	0.77	0.01	0.01	446.31	458.13	NaN
corr_group[1, 0]	0.42	0.23	0.06	0.77	0.01	0.01	446.31	458.13	NaN
corr_group[1, 1]	1.00	0.00	1.00	1.00	0.00	0.00	996.33	963.91	NaN
corr_region[0, 0]	1.00	0.00	1.00	1.00	0.00	0.00	3000.00	3000.00	NaN
...	...	...	...	...	...	...	...	...	...
sigma	0.49	0.02	0.46	0.52	0.00	0.00	1167.56	640.41	NaN
sigma_group[0]	0.42	0.09	0.30	0.55	0.00	0.00	875.35	741.04	NaN
sigma_group[1]	0.25	0.06	0.16	0.35	0.00	0.00	342.37	524.02	NaN
sigma_region[0]	1.05	0.42	0.54	1.66	0.02	0.01	539.86	741.76	NaN
sigma_region[1]	0.47	0.24	0.15	0.75	0.01	0.01	531.66	569.97	NaN

Python (Non-Centered)

```
# Data and m object carry over from the Centered tab above.

def model_nested_noncentered(y, x, group_id, region_id, N_groups=20, N_regions=5):
    sigma = m.dist.exponential(1, name='sigma')
```

```
    # 1. Region level
    mu_global    = jnp.stack([m.dist.normal(5, 2,  name='global_intercept'),
                              m.dist.normal(-1, 1, name='global_beta')])
    sigma_region = m.dist.exponential(1, shape=(2,), name='sigma_region')
    L_region     = m.dist.lkj_cholesky(2, 2, name='L_region')
    z_region     = m.dist.normal(0, 1, name='z_region', shape=(2, N_regions))
    region_effects = mu_global + ((sigma_region[..., None] * L_region) @ z_region).T

    # 2. Group level - parent mapping via JAX scatter (traceable under pmap)
    pid          = jnp.zeros(N_groups, dtype=jnp.int32).at[group_id].set(region_id)
    sigma_group  = m.dist.exponential(1, shape=(2,), name='sigma_group')
    L_group      = m.dist.lkj_cholesky(2, 2, name='L_group')
    z_group      = m.dist.normal(0, 1, name='z_group', shape=(2, N_groups))
    group_effects = region_effects[pid] + ((sigma_group[..., None] * L_group) @ z_group).T

    mu_est = group_effects[group_id, 0] + group_effects[group_id, 1] * x
    m.dist.normal(mu_est, sigma, obs=y)

m.fit(model_nested_noncentered, num_samples=1000, num_warmup=500, num_chains=1)
m.summary()
```

```
/home/sosa/.local/lib/python3.10/site-packages/jax/_src/ops/scatter.py:108: FutureWarning:

scatter inputs have incompatible types: cannot safely cast value from dtype=int64 to dtype=i

  0%|          | 0/1500 [00:00<?, ?it/s]warmup:   0%|          | 1/1500 [00:01<29:18,  1.17s/
/home/sosa/work/BF/BayesForge/Diagnostic/jax_diagnostics.py:214: RuntimeWarning:

invalid value encountered in scalar divide
```

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
L_group[0, 0]	1.00	0.00	1.00	1.00	0.00	0.00	3000.00	3000.00	NaN
L_group[0, 1]	0.00	0.00	0.00	0.00	0.00	0.00	3000.00	3000.00	NaN
L_group[1, 0]	0.42	0.23	0.03	0.75	0.01	0.01	584.84	675.29	NaN
L_group[1, 1]	0.87	0.11	0.72	1.00	0.00	0.00	581.67	675.29	NaN
L_region[0, 0]	1.00	0.00	1.00	1.00	0.00	0.00	3000.00	3000.00	NaN
...	...	...	...	...	...	...	...	...	...
z_region[1, 0]	-0.71	0.66	-1.71	0.38	0.02	0.02	831.50	720.19	NaN
z_region[1, 1]	1.05	0.65	0.09	2.11	0.02	0.02	934.02	733.74	NaN
z_region[1, 2]	-0.58	0.66	-1.62	0.42	0.03	0.02	661.19	724.94	NaN

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
z_region[1, 3]	0.39	0.79	-0.75	1.73	0.02	0.02	1060.03	744.14	NaN
z_region[1, 4]	-0.02	0.70	-1.10	1.08	0.02	0.02	884.18	604.35	NaN

Python (Built-in)

```
from BayesForge import bf
import jax.numpy as jnp

# Setup device -----------------------------------------------
m = bf(platform='cpu')

# Load data - column args are mapped automatically from the DataFrame.
# N_groups and N_regions are dataset-specific; provide them as defaults.
data_path = m.load.sim_nested_effects(only_path=True)
m.data(data_path)

# group_ids: one obs-level index array per level, top to bottom.
# Parent structure is derived automatically from these arrays.
def model_nested_builtin(y, x, group_id, region_id, N_groups=20, N_regions=5):
    sigma = m.dist.exponential(1, name='sigma')

    a_g_est, b_g_est = m.effects.nested_varying_effects(
        N_vars=2,
        names=["region", "group"],
        N_groups=[N_regions, N_groups],
        group_ids=[region_id, group_id],
        centered=False,
    )

    mu_est = a_g_est + b_g_est * x
    m.dist.normal(mu_est, sigma, obs=y)

m.fit(model_nested_builtin, num_samples=1000, num_warmup=500, num_chains=1)
m.summary()
```

```
jax.local_device_count 32
```

```
/home/sosa/.local/lib/python3.10/site-packages/jax/_src/ops/scatter.py:108: FutureWarning:
```

```
scatter inputs have incompatible types: cannot safely cast value from dtype=int64 to dtype=in

  0%|          | 0/1500 [00:00<?, ?it/s]warmup:   0%|          | 1/1500 [00:01<28:15,  1.13s/
/home/sosa/work/BF/BayesForge/Diagnostic/jax_diagnostics.py:214: RuntimeWarning:

invalid value encountered in scalar divide
```

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r
L_corr_group[0, 0]	1.00	0.00	1.00	1.00	0.00	0.00	3000.00	3000.00	N
L_corr_group[0, 1]	0.00	0.00	0.00	0.00	0.00	0.00	3000.00	3000.00	N
L_corr_group[1, 0]	0.42	0.23	0.03	0.75	0.01	0.01	584.84	675.29	N
L_corr_group[1, 1]	0.87	0.11	0.72	1.00	0.00	0.00	581.67	675.29	N
L_corr_region[0, 0]	1.00	0.00	1.00	1.00	0.00	0.00	3000.00	3000.00	N
...	...	...	...	...	...	...	...	...	..
z_region[1, 0]	-0.71	0.66	-1.71	0.38	0.02	0.02	831.50	720.19	N
z_region[1, 1]	1.05	0.65	0.09	2.11	0.02	0.02	934.02	733.74	N
z_region[1, 2]	-0.58	0.66	-1.62	0.42	0.03	0.02	661.19	724.94	N
z_region[1, 3]	0.39	0.79	-0.75	1.73	0.02	0.02	1060.03	744.14	N
z_region[1, 4]	-0.02	0.70	-1.10	1.08	0.02	0.02	884.18	604.35	N

Python (Built-in Centered)

```python
from BayesForge import bf
import jax.numpy as jnp

# Setup device ---------------------------------------------
m = bf(platform='cpu')

# Load data - column args are mapped automatically from the DataFrame.
# N_groups and N_regions are dataset-specific; provide them as defaults.
data_path = m.load.sim_nested_effects(only_path=True)
m.data(data_path)

def model_nested_builtin_centered(y, x, group_id, region_id, N_groups=20, N_regions=5):
    sigma = m.dist.exponential(1, name='sigma')

    a_g_est, b_g_est = m.effects.nested_varying_effects(
        N_vars=2,
        names=["region", "group"],
```

```
        N_groups=[N_regions, N_groups],
        group_ids=[region_id, group_id],
        centered=True,
    )

    mu_est = a_g_est + b_g_est * x
    m.dist.normal(mu_est, sigma, obs=y)

m.fit(model_nested_builtin_centered, num_samples=1000, num_warmup=500, num_chains=1)
m.summary()
```

```
/home/sosa/.local/lib/python3.10/site-packages/jax/_src/ops/scatter.py:108: FutureWarning:

scatter inputs have incompatible types: cannot safely cast value from dtype=int64 to dtype=in

jax.local_device_count 32

  0%|          | 0/1500 [00:00<?, ?it/s]warmup:   0%|          | 1/1500 [00:01<42:58,  1.72s/
/home/sosa/work/BF/BayesForge/Diagnostic/jax_diagnostics.py:214: RuntimeWarning:

invalid value encountered in scalar divide
```

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_ha
corr_group[0, 0]	1.00	0.00	1.00	1.00	0.00	0.00	3000.00	3000.00	NaN
corr_group[0, 1]	0.40	0.24	0.08	0.80	0.01	0.01	605.50	556.22	NaN
corr_group[1, 0]	0.40	0.24	0.08	0.80	0.01	0.01	605.50	556.22	NaN
corr_group[1, 1]	1.00	0.00	1.00	1.00	0.00	0.00	1023.28	1031.23	NaN
corr_region[0, 0]	1.00	0.00	1.00	1.00	0.00	0.00	3000.00	3000.00	NaN
...	...	...	...	...	...	...	...	...	...
sigma	0.49	0.02	0.46	0.53	0.00	0.00	1073.98	760.39	NaN
sigma_group[0]	0.42	0.08	0.29	0.54	0.00	0.00	919.03	737.85	NaN
sigma_group[1]	0.25	0.07	0.15	0.34	0.00	0.00	457.05	510.07	NaN
sigma_region[0]	1.04	0.41	0.47	1.56	0.01	0.01	758.53	766.58	NaN
sigma_region[1]	0.46	0.24	0.17	0.75	0.01	0.01	542.06	631.21	NaN

Julia

```
using BayesForge

# Setup device ------------------------------------------------
m = importBF(platform="cpu")

# Load data - column args mapped automatically from the DataFrame.
# N_groups and N_regions are dataset-specific; provide them as defaults.
data_path = m.load.sim_nested_effects(only_path=true)
m.data(data_path)

# Define model ------------------------------------------------
@BF function model(y, x, group_id, region_id, N_groups=20, N_regions=5)
    sigma = m.dist.exponential(1, name="sigma")

    a_g_est, b_g_est = m.effects.nested_varying_effects(
        N_vars    = 2,
        names     = ["region", "group"],
        N_groups  = [N_regions, N_groups],
        group_ids = [region_id, group_id],
        centered  = false,
    )

    mu = a_g_est .+ b_g_est .* x
    m.dist.normal(mu, sigma, obs=y)
end

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

R

```
library(BayesForge)

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

# Load data - column args mapped automatically from the DataFrame.
# N_groups and N_regions are dataset-specific; provide them as defaults.
data_path = m$load$sim_nested_effects(only_path=TRUE)
```

```
m$data(data_path)

# Define model ------------------------------------------------
model <- function(y, x, group_id, region_id, N_groups=20L, N_regions=5L) {
  sigma = m$dist$exponential(1, name='sigma')

  veff = m$effects$nested_varying_effects(
    N_vars    = 2L,
    names     = list('region', 'group'),
    N_groups  = list(N_regions, N_groups),
    group_ids = list(region_id, group_id),
    centered  = FALSE,
  )
  a_g_est = veff[[1]]
  b_g_est = veff[[2]]

  mu = a_g_est + b_g_est * x
  m$dist$normal(mu, sigma, obs=y)
}

# Run MCMC ----------------------------------------------------
m$fit(model)
m$summary()
```

Mathematical Details

We model the outcome Y_i for observation i. The intercept α_j and slope β_j for group $j(i)$ are nested within region $k(j)$.

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

$$\mu_i = \alpha_{[j(i)]} + \beta_{[j(i)]} X_i$$

Centered Parameterization

The vector of group-level effects follows a Multivariate Normal distribution centered on the region-level effects:

$$\begin{bmatrix} \alpha_j \\ \beta_j \end{bmatrix} \sim \text{MVNormal}\left(\begin{bmatrix} \gamma_{\alpha,[k(j)]} \\ \gamma_{\beta,[k(j)]} \end{bmatrix}, \Sigma_{group} \right)$$

The region-level effects themselves follow their own Multivariate Normal distribution:

$$\begin{bmatrix} \gamma_{\alpha,k} \\ \gamma_{\beta,k} \end{bmatrix} \sim \text{MVNormal}\left(\begin{bmatrix} \bar{\gamma}_\alpha \\ \bar{\gamma}_\beta \end{bmatrix}, \Sigma_{region} \right)$$

Where: - $\Sigma = \mathbf{SRS}$ (diagonal matrix of standard deviations $\mathbf{S}$ and correlation matrix $\mathbf{R}$). - $\bar{\gamma}_\alpha, \bar{\gamma}_\beta$ are the global average intercept and slope. - $\Sigma_{region}, \Sigma_{group}$ are the covariance matrices for each hierarchical level.

Non-Centered Parameterization

To avoid funnel geometries, the non-centered formulation instead uses Cholesky factors L_{region} and L_{group} alongside standardized, non-correlated normal variables:

$$\begin{bmatrix} z_{\alpha,k} \\ z_{\beta,k} \end{bmatrix} \sim \text{Normal}(0,1), \quad \begin{bmatrix} z_{\alpha,j} \\ z_{\beta,j} \end{bmatrix} \sim \text{Normal}(0,1)$$

The region-level and group-level distributions are linearly transformed:

$$\begin{bmatrix} \gamma_{\alpha,k} \\ \gamma_{\beta,k} \end{bmatrix} = \begin{bmatrix} \bar{\gamma}_\alpha \\ \bar{\gamma}_\beta \end{bmatrix} + \text{diag}(\mathbf{S}_{region}) L_{region} \begin{bmatrix} z_{\alpha,k} \\ z_{\beta,k} \end{bmatrix}$$

$$\begin{bmatrix} \alpha_j \\ \beta_j \end{bmatrix} = \begin{bmatrix} \gamma_{\alpha,[k(j)]} \\ \gamma_{\beta,[k(j)]} \end{bmatrix} + \text{diag}(\mathbf{S}_{group}) L_{group} \begin{bmatrix} z_{\alpha,j} \\ z_{\beta,j} \end{bmatrix}$$

Notes

> **Note**
>
> - By nesting the **MVN** prior, we allow the model to learn how the correlation between intercepts and slopes persists across different levels of the hierarchy.
> - The non-centered parameterization (used by default in the `m.effects.nested_varying_effects` built-in function) is critical for convergence in these complex models.
> - This approach can be generalized to N levels and M variables by increasing the dimensionality of the vectors and matrices.

Reference(s)