# Latent Variable Models (WIP)

## General Principles

In some scenarios, the observed data does not directly reflect the underlying structure or factors influencing the outcome. Instead, *latent variables*—variables that are not directly observed but are inferred from the data—can help model this hidden structure. These latent variables capture unobserved factors that affect the relationship between predictors ($X$) and the outcome ($Y$).

We model the relationship between the predictor variables ($X$) and the outcome variable ($Y$) with a latent variable ($Z$) as follows:

$$Y = f(X, Z) + \epsilon$$

Where: - $Y$ is the observed outcome variable. - $X$ is the observed predictor variable(s). - $Z$ is the latent (unobserved) variable, which we aim to infer. - *f(X, Z)* is the function that relates $X$ and $Z$ to $Y$. - is the error term, typically assumed to be normally distributed with mean 0 and variance ^2.

The latent variable $Z$ can represent various phenomena, such as group-level effects, time-varying trends, or individual-level factors, that are not captured by the observed predictors alone.

## Considerations

In Bayesian regression with latent variables, we consider the uncertainty in both the observed and latent variables. We declare prior distributions for the latent variables, in addition to the usual priors for regression coefficients and intercepts. These latent variables are often modeled using Gaussian distributions (*Normal* priors) or more flexible distributions such as *Multivariate Normal* for correlations among the latent variables.

The goal is to infer the posterior distribution over both the parameters and the latent variables, given the observed data.

## Example

Below is an example code snippet demonstrating Bayesian regression with latent variables using TensorFlow Probability:

## Python

```
from BI import bi, jnp


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

# Data Simulation ----------------------------------------------
NY = 4  # Number of dependent variables or outcomes (e.g., dimensions for latent variables)
NV = 8  # Number of observations or individual-level data points (e.g., subjects)
N = 100
K = 5
a = 0.5
# Generate the means and offsets for the data
# means: Generate random normal means for each of the NY outcomes
# offsets: Generate random normal offsets for each of the NV observations
means = m.dist.normal(0, 1, shape=(NY,), sample=True, seed=10)
offsets = m.dist.normal(0, 1, shape=(NV, 1), sample=True, seed=20)

Y2 = offsets + means

# Simulate individual-level random effects (e.g., random slopes or intercepts)
# b_individual: A matrix of size (N, K) where N is the number of individuals and K is the nu
b_individual = m.dist.normal(0, 1, shape=(N, K), sample=True, seed=0)

# mu: Add an additional effect 'a' to the individual-level random effects 'b_individual'
# 'a' could represent a population-level effect or a baseline
mu = b_individual + a

# Convert Y2 to a JAX array for further computation in a JAX-based framework
Y2 = jnp.array(Y2)


# Set data -----------------------------------------------
dat = dict(
```

```
    NY = NY,
    NV = NV,
    Y2 = Y2
)
m.data_on_model = dat

# Define model ------------------------------------------------
def model(NY, NV, Y2):
    means = m.dist.normal(0, 1, shape=(NY,), name='means')
    offset = m.dist.normal(0, 1, shape=(NV, 1), name='offset')
    sigma = m.dist.exponential(1, shape=(NY,), name='sigma')
    tmp = jnp.tile(means, (NV, 1)).reshape(NV, NY)
    mu_l = tmp + offset
    m.dist.normal(mu_l, jnp.tile(sigma, [NV, 1]), obs=Y2)

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

# Summary -----------------------------------------------------
m.summary()
```

```
jax.local_device_count 16
```

```
  0%|          | 0/1000 [00:00<?, ?it/s]warmup:   0%|          | 1/1000 [00:01<17:52,  1.07s/
arviz - WARNING - Shape validation failed: input_shape: (1, 500), minimum_shape: (chains=2, d
```

| | mean | sd | hdi_5.5% | hdi_94.5% | mcse_mean | mcse_sd | ess_bulk | ess_tail | r_hat |
|---|---|---|---|---|---|---|---|---|---|
| means[0] | -1.45 | 0.0 | -1.45 | -1.45 | 0.0 | 0.0 | 1.35 | 14.79 | NaN |
| means[1] | -0.73 | 0.0 | -0.73 | -0.72 | 0.0 | 0.0 | 1.35 | 14.79 | NaN |
| means[2] | 1.24 | 0.0 | 1.24 | 1.25 | 0.0 | 0.0 | 1.35 | 14.79 | NaN |
| means[3] | -0.60 | 0.0 | -0.61 | -0.60 | 0.0 | 0.0 | 1.34 | 15.50 | NaN |
| offset[0, 0] | -0.64 | 0.0 | -0.64 | -0.64 | 0.0 | 0.0 | 1.35 | 14.79 | NaN |
| offset[1, 0] | 0.75 | 0.0 | 0.75 | 0.76 | 0.0 | 0.0 | 1.34 | 14.79 | NaN |
| offset[2, 0] | -1.14 | 0.0 | -1.14 | -1.13 | 0.0 | 0.0 | 1.35 | 14.79 | NaN |
| offset[3, 0] | 0.10 | 0.0 | 0.09 | 0.10 | 0.0 | 0.0 | 1.35 | 14.79 | NaN |
| offset[4, 0] | -0.29 | 0.0 | -0.30 | -0.29 | 0.0 | 0.0 | 1.35 | 14.79 | NaN |
| offset[5, 0] | -0.80 | 0.0 | -0.81 | -0.80 | 0.0 | 0.0 | 1.34 | 14.79 | NaN |
| offset[6, 0] | 0.17 | 0.0 | 0.17 | 0.17 | 0.0 | 0.0 | 1.35 | 14.79 | NaN |
| offset[7, 0] | 0.75 | 0.0 | 0.75 | 0.75 | 0.0 | 0.0 | 1.35 | 14.79 | NaN |
| sigma[0] | 0.00 | 0.0 | 0.00 | 0.00 | 0.0 | 0.0 | 1.41 | 10.90 | NaN |

| | mean | sd | hdi_5.5% | hdi_94.5% | mcse_mean | mcse_sd | ess_bulk | ess_tail | r_hat |
|---|---|---|---|---|---|---|---|---|---|
| sigma[1] | 0.00 | 0.0 | 0.00 | 0.00 | 0.0 | 0.0 | 12.57 | 10.86 | NaN |
| sigma[2] | 0.00 | 0.0 | 0.00 | 0.00 | 0.0 | 0.0 | 17.85 | 12.09 | NaN |
| sigma[3] | 0.00 | 0.0 | 0.00 | 0.00 | 0.0 | 0.0 | 1.64 | 11.56 | NaN |

**R**

## Mathematical Details