

# Varying Intercepts Models

## General Principles

To model the relationship between a dependent variable and an independent variable while allowing for different intercepts across groups or clusters, we can use a *Varying Intercepts* model. This approach is particularly useful when data are grouped (e.g., by subject, location, or time period) and we expect the baseline level of the outcome to vary across these groups.

## Considerations

### Note

- We have the same considerations as for [Regression for a continuous variable](#).
- The main idea of varying intercepts is to generate an intercept for each group, allowing each group to start at different levels. Thus, the intercept  $\alpha_k$  is defined uniquely for each of the  $K$  declared groups.
- In the code below, the intercept `alpha` for each of the  $k$  declared groups shares two priors, `a_bar` and `sigma`, which are respectively modeled by a Normal and an Exponential distribution.

## Example

Below is an example code snippet demonstrating Bayesian regression with varying intercepts using the Bayesian Inference (BI) package. The data consists of a dependent variable representing individuals' survival (`surv`) and an independent categorical variable (`tank`), which indicates the tank where the individual was born, with a total of 48 tanks. This example is based on McElreath (2018).

## Python (Raw)

```
from BI import bi, jnp

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

# Import Data & Data Manipulation -----
# Import
from importlib.resources import files
data_path = files('BI.resources.data') / 'reedfrogs.csv'
m.data(data_path, sep=';')
# Manipulate
m.df["tank"] = jnp.arange(m.df.shape[0])

# Define model -----
def model(tank, surv, density):
    sigma = m.dist.exponential( 1, name = 'sigma')
    a_bar = m.dist.normal( 0., 1.5, name = 'a_bar')
    alpha = m.dist.normal( a_bar, sigma, shape= tank.shape, name = 'alpha')
    p = alpha[tank]
    m.dist.binomial(total_count = density, logits = p, obs=surv)

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

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

jax.local\_device\_count 16

0%| 0/1000 [00:00<?, ?it/s] warmup: 0%| 1/1000 [00:01<26:54, 1.62s, arviz - WARNING - Shape validation failed: input\_shape: (1, 500), minimum\_shape: (chains=2, 0)

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a_bar	1.35	0.27	0.95	1.79	0.01	0.01	467.13	297.56	NaN
alpha[0]	2.13	0.85	0.69	3.34	0.04	0.04	514.75	388.20	NaN
alpha[1]	3.09	1.09	1.30	4.47	0.05	0.05	565.53	436.09	NaN
alpha[2]	0.99	0.68	0.06	2.14	0.03	0.04	676.80	327.00	NaN
alpha[3]	3.12	1.10	1.27	4.66	0.05	0.06	594.03	428.01	NaN

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
alpha[4]	2.15	0.89	0.82	3.57	0.03	0.04	726.38	463.33	NaN
alpha[5]	2.19	0.91	0.87	3.66	0.04	0.05	771.96	371.60	NaN
alpha[6]	3.12	1.07	1.57	4.72	0.05	0.04	520.47	347.62	NaN
alpha[7]	2.14	0.87	0.73	3.38	0.03	0.05	1221.40	424.56	NaN
alpha[8]	-0.16	0.62	-1.02	0.94	0.02	0.04	811.97	219.25	NaN
alpha[9]	2.10	0.86	0.67	3.33	0.04	0.04	513.47	370.85	NaN
alpha[10]	0.99	0.69	-0.13	2.03	0.02	0.03	927.72	411.04	NaN
alpha[11]	0.62	0.65	-0.49	1.61	0.03	0.03	561.63	290.42	NaN
alpha[12]	1.05	0.72	-0.12	2.05	0.03	0.04	708.09	317.39	NaN
alpha[13]	0.18	0.62	-0.83	1.18	0.02	0.03	680.04	406.36	NaN
alpha[14]	2.16	0.84	0.69	3.32	0.03	0.04	843.95	375.45	NaN
alpha[15]	2.15	0.91	0.80	3.65	0.04	0.04	499.23	283.02	NaN
alpha[16]	2.91	0.79	1.42	3.86	0.03	0.04	716.71	353.12	NaN
alpha[17]	2.40	0.67	1.30	3.38	0.03	0.04	727.24	247.66	NaN
alpha[18]	2.03	0.60	1.10	2.91	0.03	0.03	642.62	326.17	NaN
alpha[19]	3.75	0.95	2.17	5.10	0.04	0.04	582.43	361.94	NaN
alpha[20]	2.35	0.61	1.41	3.24	0.02	0.03	824.43	261.58	NaN
alpha[21]	2.36	0.66	1.34	3.37	0.02	0.03	761.56	320.99	NaN
alpha[22]	2.40	0.62	1.37	3.31	0.03	0.03	573.35	352.38	NaN
alpha[23]	1.70	0.53	0.93	2.56	0.02	0.02	839.61	423.04	NaN
alpha[24]	-1.03	0.45	-1.69	-0.33	0.02	0.02	731.40	365.09	NaN
alpha[25]	0.16	0.42	-0.48	0.83	0.02	0.02	651.44	369.19	NaN
alpha[26]	-1.41	0.45	-2.15	-0.71	0.02	0.02	597.70	314.86	NaN
alpha[27]	-0.48	0.42	-1.08	0.19	0.02	0.02	537.11	388.83	NaN
alpha[28]	0.16	0.40	-0.61	0.67	0.02	0.02	431.36	392.36	NaN
alpha[29]	1.46	0.48	0.75	2.27	0.02	0.02	755.76	404.80	NaN
alpha[30]	-0.62	0.43	-1.18	0.20	0.02	0.02	706.26	394.05	NaN
alpha[31]	-0.30	0.43	-0.90	0.41	0.02	0.02	692.89	408.34	NaN
alpha[32]	3.20	0.76	2.04	4.48	0.03	0.03	563.22	379.37	NaN
alpha[33]	2.70	0.64	1.83	3.88	0.02	0.03	758.59	428.71	NaN
alpha[34]	2.74	0.65	1.65	3.71	0.03	0.03	545.98	387.72	NaN
alpha[35]	2.06	0.49	1.36	2.94	0.02	0.02	721.44	392.52	NaN
alpha[36]	2.09	0.57	1.25	2.95	0.02	0.03	732.34	371.66	NaN
alpha[37]	3.89	0.95	2.29	5.11	0.04	0.05	508.53	294.22	NaN
alpha[38]	2.70	0.70	1.66	3.76	0.03	0.05	696.43	275.91	NaN
alpha[39]	2.38	0.54	1.46	3.19	0.02	0.03	588.18	352.03	NaN
alpha[40]	-1.77	0.47	-2.50	-1.03	0.02	0.02	965.46	269.98	NaN
alpha[41]	-0.59	0.31	-1.09	-0.13	0.01	0.01	729.57	401.86	NaN
alpha[42]	-0.44	0.37	-1.04	0.16	0.01	0.02	744.12	373.34	NaN
alpha[43]	-0.33	0.36	-0.90	0.23	0.01	0.02	818.64	313.40	NaN
alpha[44]	0.57	0.34	0.07	1.09	0.02	0.01	487.25	426.63	NaN

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
alpha[45]	-0.58	0.38	-1.25	-0.04	0.01	0.02	1084.06	421.32	NaN
alpha[46]	2.06	0.46	1.34	2.75	0.02	0.02	569.86	389.45	NaN
alpha[47]	0.02	0.35	-0.50	0.61	0.01	0.02	710.85	353.56	NaN
sigma	1.62	0.21	1.30	1.92	0.01	0.01	339.24	267.98	NaN

## Python (Build in function)

```

from BI import bi, jnp

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

# Import Data & Data Manipulation -----
# Import
from importlib.resources import files
data_path = files('BI.resources.data') / 'reedfrogs.csv'
m.data(data_path, sep=';')
# Manipulate
m.df["tank"] = jnp.arange(m.df.shape[0])

# Define model -----
def model(tank, surv, density):
    alpha = m.effects.varying_intercept(N_groups=48, group_id=tank, group_name = 'tank')
    m.dist.binomial(total_count = density, logits = alpha, obs=surv)

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

# Diagnostic -----
m.summary()

```

jax.local\_device\_count 16

0%| 0/1000 [00:00<?, ?it/s] warmup: 0%| 1/1000 [00:01<27:50, 1.67s, 0.00it/s] sampling: 0%| 999/1000 [00:01<27:50, 1.67s, 0.00it/s] arviz - WARNING - Shape validation failed: input\_shape: (1, 500), minimum\_shape: (chains=2, 500)

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail
global_intercept_tank	1.36	0.26	0.94	1.77	0.01	0.01	563.88	182.64
intercept_tank[0]	2.10	0.86	0.72	3.38	0.03	0.04	811.23	369.49
intercept_tank[1]	3.04	1.11	1.31	4.69	0.05	0.06	592.21	290.26
intercept_tank[2]	1.00	0.65	-0.06	2.06	0.03	0.04	647.60	343.02
intercept_tank[3]	3.06	1.10	1.16	4.67	0.05	0.07	547.75	178.68
intercept_tank[4]	2.14	0.89	0.80	3.44	0.03	0.05	953.17	307.45
intercept_tank[5]	2.15	0.89	0.91	3.72	0.03	0.04	850.84	421.61
intercept_tank[6]	3.13	1.11	1.40	4.93	0.06	0.06	455.95	282.69
intercept_tank[7]	2.18	0.89	1.00	3.73	0.04	0.04	504.63	338.18
intercept_tank[8]	-0.16	0.62	-1.02	0.89	0.02	0.03	602.28	387.08
intercept_tank[9]	2.15	0.92	0.81	3.47	0.04	0.06	588.68	335.17
intercept_tank[10]	1.00	0.70	0.03	2.17	0.02	0.04	921.27	305.21
intercept_tank[11]	0.58	0.61	-0.31	1.64	0.02	0.03	718.11	320.40
intercept_tank[12]	1.07	0.69	-0.06	2.08	0.03	0.03	511.78	369.19
intercept_tank[13]	0.20	0.60	-0.69	1.20	0.02	0.02	646.21	339.77
intercept_tank[14]	2.12	0.83	0.90	3.52	0.04	0.04	563.92	353.53
intercept_tank[15]	2.15	0.91	0.69	3.50	0.04	0.05	565.23	355.31
intercept_tank[16]	2.87	0.76	1.74	4.10	0.03	0.03	918.26	411.31
intercept_tank[17]	2.42	0.68	1.36	3.40	0.03	0.05	642.69	209.59
intercept_tank[18]	2.03	0.60	0.95	2.83	0.03	0.02	460.37	346.04
intercept_tank[19]	3.73	1.04	2.02	5.18	0.05	0.05	587.35	381.98
intercept_tank[20]	2.38	0.64	1.47	3.46	0.03	0.03	592.59	386.77
intercept_tank[21]	2.39	0.69	1.30	3.37	0.03	0.03	622.27	322.91
intercept_tank[22]	2.42	0.65	1.26	3.35	0.03	0.04	435.39	247.89
intercept_tank[23]	1.69	0.53	0.88	2.48	0.02	0.03	676.05	393.95
intercept_tank[24]	-1.00	0.48	-1.69	-0.26	0.02	0.02	792.69	344.82
intercept_tank[25]	0.14	0.40	-0.44	0.81	0.02	0.02	570.16	385.30
intercept_tank[26]	-1.42	0.48	-2.04	-0.54	0.02	0.03	699.69	349.23
intercept_tank[27]	-0.50	0.42	-1.12	0.17	0.02	0.02	482.27	381.98
intercept_tank[28]	0.15	0.39	-0.39	0.82	0.02	0.02	655.74	295.17
intercept_tank[29]	1.44	0.48	0.66	2.19	0.02	0.02	491.85	336.45
intercept_tank[30]	-0.62	0.42	-1.24	0.11	0.02	0.02	601.10	369.15
intercept_tank[31]	-0.28	0.44	-0.96	0.37	0.02	0.02	587.55	285.26
intercept_tank[32]	3.18	0.72	2.05	4.19	0.03	0.03	481.03	336.93
intercept_tank[33]	2.67	0.62	1.57	3.49	0.02	0.03	735.44	289.00
intercept_tank[34]	2.75	0.62	1.76	3.73	0.03	0.03	388.61	347.85
intercept_tank[35]	2.06	0.47	1.27	2.80	0.02	0.02	840.45	343.02
intercept_tank[36]	2.09	0.53	1.32	2.91	0.02	0.02	756.53	393.66
intercept_tank[37]	3.90	0.95	2.47	5.27	0.05	0.04	336.15	304.76
intercept_tank[38]	2.70	0.67	1.72	3.72	0.03	0.04	665.74	311.83
intercept_tank[39]	2.34	0.53	1.45	3.15	0.02	0.03	738.32	281.88

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail
intercept_tank[40]	-1.78	0.45	-2.48	-1.06	0.01	0.02	913.76	357.43
intercept_tank[41]	-0.58	0.34	-1.14	-0.07	0.01	0.01	730.83	425.09
intercept_tank[42]	-0.44	0.39	-1.03	0.21	0.01	0.02	693.24	438.47
intercept_tank[43]	-0.33	0.34	-0.86	0.20	0.01	0.02	549.94	352.19
intercept_tank[44]	0.58	0.33	0.06	1.09	0.01	0.01	653.17	339.77
intercept_tank[45]	-0.59	0.38	-1.24	-0.02	0.02	0.02	620.34	294.19
intercept_tank[46]	2.03	0.48	1.30	2.77	0.02	0.03	657.69	351.35
intercept_tank[47]	0.01	0.33	-0.52	0.54	0.01	0.01	735.81	303.89
sd_tank	1.61	0.20	1.28	1.92	0.01	0.01	223.71	348.72

## R

```

library(BI)

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

# Import data -----
m$data(paste(system.file(package = "BI"),"/data/reedfrogs.csv", sep = ''), sep=';')
m$df$tank = c(0:(nrow(m$df)-1)) # Manipulate
m$data_to_model(list('tank', 'surv', 'density')) # Manipulate
m$data_on_model$tank = m$data_on_model$tank$astype(jnp$int32) # Manipulate
m$data_on_model$surv = m$data_on_model$surv$astype(jnp$int32) # Manipulate

# Define model -----
model <- function(tank, surv, density){
  # Parameter prior distributions
  sigma = bi.dist.exponential( 1, name = 'sigma',shape=c(1))
  a_bar = bi.dist.normal(0, 1.5, name='a_bar',shape=c(1))
  alpha = bi.dist.normal(a_bar, sigma, name='alpha', shape =c(48))
  p = alpha[tank]
  # Likelihood
  m$binomial(total_count = density, logits = p, obs=surv)
}

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

```

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

## Mathematical Details

We model the relationship between the independent variable  $X$  and the outcome variable  $Y$  while accounting for varying intercepts  $\alpha$  for each group where  $k(i)$  give us group belonging for observation  $i$ , using the following equation:

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

$$\mu_i = \alpha_{[k(i)]} + \beta X_i$$

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

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

$$\alpha_{[k]} \sim \text{Normal}(\bar{\alpha}, \varsigma)$$

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

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

Where:

- $Y_i$  is the outcome variable for observation  $i$ .
- $\alpha_{[k(i)]}$  is the varying intercept corresponding to the group  $k$  of observation  $i$ .
- $\beta$  is the regression coefficient.
- $\sigma$  is a standard deviation parameter, which here has an Exponential prior that constrains it to be positive.
- $\bar{\alpha}$  is the overall mean intercept.
- $\varsigma$  is the variance of the intercepts across groups.

## Notes

### Note

- We can apply multiple variables similarly to [Chapter 2](#).
- We can apply interaction terms similarly to [Chapter 3](#).
- We can apply categorical variables similarly to [Chapter 4](#).
- We can apply varying intercepts with any distribution developed in previous chapters.

## Reference(s)

McElreath, Richard. 2018. *Statistical Rethinking: A Bayesian course with examples in R and Stan*. Chapman; Hall/CRC.