Regression with a Categorical Independent Variable

General Principles

To study the relationship between a categorical independent variable and a continuous dependent variable, we use a *Categorical model* which applies *stratification*.

Stratification involves modeling how the k different categories of the independent variable affect the target continuous variable by performing a regression for each k category and assigning a regression coefficient for each category. To implement stratification, categorical variables are often encoded using one-hot encoding or by converting categories to indices .

Considerations

> **Note**
>
> - We have the same considerations as for Regression for a Continuous Variable.
> - As we generate regression coefficients for each k category, we need to specify a prior with a shape equal to the number of categories k in the code (see comments in the code).
> - To compare differences between categories, we need to compute the distribution of the differences between categories, known as the contrast distribution. **Never compare confidence intervals or p-values directly**.

Example

Below is an example of code that demonstrates Bayesian regression with an independent categorical variable using the Bayesian Inference (BI) package. The data consist of one continuous dependent variable (*kcal_per_g*), representing the caloric value of milk per gram, a categorical

independent variable (*index_clade*), representing species clade membership, and a continuous independent variable (*mass*), representing the mass of individuals in the clade. The goal is to estimate the differences in milk calories between clades. This example is based on McElreath (2018).

Python

```
from BI import bi

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

# Import Data & Data Manipulation ------------------------------------------------
# Import
from importlib.resources import files
data_path = m.load.milk(only_path = True)
m.data(data_path, sep=';')
m.index(["clade"]) # Convert clade names into index
m.scale(['kcal_per_g']) # Scale

# Define model -------------------------------------------------
def model(kcal_per_g, index_clade, mass):
    a = m.dist.normal(0, 0.5, shape=(4,), name = 'a') # shape based on the number of clades
    b = m.dist.normal(0, 0.5, shape=(4,), name = 'b')
    s = m.dist.exponential( 1, name = 's')
    mu = a[index_clade]+b[index_clade]*mass
    m.dist.normal(mu, s, obs=kcal_per_g)


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

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

```
jax.local_device_count 16
```

```
  0%|          | 0/1000 [00:00<?, ?it/s]warmup:   0%|          | 1/1000 [00:01<19:07,  1.15s/
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
a[0]	-0.32	0.35	-0.79	0.34	0.02	0.02	430.93	365.09	NaN
a[1]	0.59	0.30	0.10	1.06	0.01	0.01	412.79	283.95	NaN
a[2]	0.31	0.37	-0.28	0.87	0.02	0.02	404.55	368.44	NaN
a[3]	-0.17	0.49	-0.92	0.64	0.03	0.02	376.94	232.82	NaN
b[0]	-0.00	0.01	-0.02	0.01	0.00	0.00	456.57	350.35	NaN
b[1]	-0.17	0.12	-0.36	0.04	0.01	0.01	356.03	342.61	NaN
b[2]	0.08	0.07	-0.02	0.19	0.00	0.00	371.61	388.81	NaN
b[3]	-0.27	0.27	-0.72	0.13	0.01	0.01	352.82	217.58	NaN
s	0.80	0.12	0.59	0.97	0.01	0.00	386.61	438.95	NaN

R

```r
library(BayesianInference)
m=importBI(platform='cpu')

# Load csv file
m$data(m$load$milk(only_path = T), sep=';')
m$scale(list('kcal.per.g')) # Manipulate
m$index(list('clade')) # Scale
m$data_to_model(list('kcal_per_g', 'index_clade')) # Send to model (convert to jax array)

# Define model ------------------------------------------------
model <- function(kcal_per_g, index_clade){
  # Parameter prior distributions
  beta = bi.dist.normal( 0, 0.5, name = 'beta', shape = c(4))  # shape based on the number o
  sigma = bi.dist.exponential(1, name = 's')
  # Likelihood
  bi.dist.normal(beta[index_clade], sigma, obs=kcal_per_g)
}

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

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

Julia

```
using BayesianInference

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

# Import Data & Data Manipulation ---------------------------------------------
# Import
data_path = m.load.milk(only_path = true)
m.data(data_path, sep=';')
m.index("clade") # Convert clade names into index
m.scale(["kcal_per_g"]) # Scale

# Define model ----------------------------------------------
@BI function model(kcal_per_g, index_clade, mass)
    a = m.dist.normal(0, 0.5, shape=(4,), name = "a") # shape based on the number of clades
    b = m.dist.normal(0, 0.5, shape=(4,), name = "b")
    s = m.dist.exponential( 1, name = 's')
    mu = a[index_clade]+b[index_clade]*mass
    m.dist.normal(mu, s, obs=kcal_per_g)
end

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

# Summary ----------------------------------------------
m.summary() # Get posterior distributions
```

> **Caution**
>
> For R users, when working with indices you have to ensure 1) that indices are intergers (i.e. `as.integer(index_clade)`) and, 2) that indices start at 0 (i.e. `as.integer(index_clade)-1`).

Mathematical Details

Frequentist formulation

We model the relationship between the categorical input feature (X) and the target variable (Y) using the following equation:

$$Y_i = \alpha + \beta_k X_i + \sigma$$

Where:

- Y_i is the dependent variable for observation i.
- α is the intercept term.
- β_k are the regression coefficients for each k category.
- X_i is the encoded categorical input variable for observation i.
- σ is the error term.

We can interpret β_i as the effect of each category on Y relative to the baseline (usually one of the categories or the intercept).

Bayesian formulation

In the Bayesian formulation, we define each parameter with priors . We can express the Bayesian regression model accounting for prior distributions as follows:

$$Y \sim \text{Normal}(\alpha + \beta_K X, \sigma)$$

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

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

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

Where:

- Y_i is the dependent variable for observation i.
- α is the intercept term, which in this case has a unit-normal prior.
- β_K are slope coefficients for the K distinct independent variables categories, which also have unit-normal priors.
- X_i is the encoded categorical input variable for observation i.
- σ is a standard deviation parameter, which here has a Exponential prior that constrains it to be positive.

Notes

> **Note**
>
> - We can apply multiple variables similarly to Chapter 2: Multiple Continuous Variables.
> - We can apply interaction terms similarly to Chapter 3: Interaction between Continuous Variables.

Reference(s)

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