Name : Axel Ivanda Tanjung – PACMANN Batch 09

Applying Bayesian Hierarchical Regression to Reevaluate Concrete Strength Forecasting

Link Github : https://github.com/axeltanjung/concrete_strengh_bayes/

1. Introduction & Background

A. Introduction

Concrete is a fundamental material in modern construction, underpinning the structural integrity of buildings, bridges, roads, and various infrastructure projects. Accurate forecasting of concrete strength, an essential aspect of quality control in construction, has significant implications for both the safety and economic efficiency of construction projects. Traditionally, concrete strength prediction has relied on empirical formulas and simple regression models, but these methods often fall short in capturing the complex variability inherent in concrete properties due to diverse factors such as mix proportions, curing conditions, and material quality.

In recent years, advances in statistical modeling have offered new approaches to improve the accuracy of concrete strength forecasting. One such advancement is the application of Bayesian hierarchical regression models. Bayesian methods, with their ability to incorporate prior information and quantify uncertainty, provide a robust framework for handling the complexity and variability present in concrete strength data. Hierarchical models, in particular, are adept at managing data with multiple levels of variation, making them well-suited for concrete strength prediction where data is often collected across different batches, sites, and time periods.

Bayesian hierarchical regression offers several advantages over traditional methods. Firstly, it allows for the incorporation of prior knowledge about concrete properties and mix designs, which can enhance the predictive accuracy of the model. This is particularly valuable in cases where historical data is sparse or where new materials or techniques are introduced. By updating prior distributions with observed data, Bayesian hierarchical models provide a more flexible approach to forecasting that can adapt to new information as it becomes available.

Moreover, hierarchical regression models enable the decomposition of variance into different levels, such as within-site and between-site variability. This granular insight is crucial for understanding the factors that contribute to concrete strength variability and for making informed decisions about mix design and quality control procedures. For instance, a hierarchical model can help distinguish whether observed differences in strength are due to variations in materials, mixing processes, or curing conditions, and quantify the impact of each factor on the overall prediction.

Despite these advantages, the application of Bayesian hierarchical regression to concrete strength forecasting is not without challenges. Implementing these models requires expertise in Bayesian statistics and computational methods, as well as access to appropriate software and tools. Furthermore, the quality of the forecasts depends on the quality of the data used for model training and validation. Data collection processes must be robust and comprehensive to ensure that the hierarchical model can capture the true underlying patterns in concrete strength.

This journal aims to explore the application of Bayesian hierarchical regression in the context of concrete strength forecasting, highlighting its potential benefits and addressing the practical considerations involved in its implementation. Through a detailed examination of case studies and empirical data, we will assess how Bayesian hierarchical models compare to traditional forecasting methods in terms of accuracy, reliability, and interpretability. The insights gained from this study will contribute to the development of more effective strategies for predicting concrete strength, ultimately advancing the field of construction engineering and enhancing the safety and durability of concrete structures.

B. Objective

The primary objective of this journal is to explore the application of Bayesian hierarchical regression models in concrete strength forecasting, aiming to reevaluate and enhance traditional predictive methods. Concrete strength is critical for ensuring the safety and durability of structures, and accurate forecasting is essential for effective material utilization and quality control.

- **Assess Model Performance**: The first objective is to evaluate the performance of Bayesian hierarchical regression models compared to conventional regression techniques. We will measure improvements in prediction accuracy, precision, and reliability offered by Bayesian methods, considering their ability to model variability at multiple levels (e.g., mix, batch, site).
- **Analyze Hierarchical Structure Impact**: This study will examine how the hierarchical structure inherent in Bayesian models contributes to forecasting accuracy. By isolating site-specific, batch-specific, and mix-specific effects, we aim to determine how well the hierarchical approach captures the complex variability in concrete strength data.
- **Utilize Prior Knowledge**: We seek to investigate the role of prior distributions in enhancing forecast accuracy. This involves incorporating prior knowledge or historical data into Bayesian models and evaluating how these priors are updated with new information to refine predictions.
- **Address Implementation Challenges**: The study will identify practical challenges in applying Bayesian hierarchical models, including computational demands and data quality requirements. Recommendations will be provided for effectively integrating these models into real-world forecasting practices.
- **Guide Future Research**: Finally, based on our findings, we will offer recommendations for further research and practical applications to improve concrete strength forecasting methodologiee.

2. Dataset

Given are the variable name, variable type, the measurement unit and a brief description. The concrete compressive strength is the regression problem. The order of this listing corresponds to the order of numerals along the rows of the database.

- **Name** -- Data Type -- Measurement -- Description
- **Cement** (component 1) -- quantitative -- kg in a m3 mixture -- Input Variable
- **Blast Furnace Slag** (component 2) -- quantitative -- kg in a m3 mixture -- Input Variable
- **Fly Ash** (component 3) -- quantitative -- kg in a m3 mixture -- Input Variable
- **Water** (component 4) -- quantitative -- kg in a m3 mixture -- Input Variable
- **Superplasticizer** (component 5) -- quantitative -- kg in a m3 mixture -- Input Variable
- **Coarse Aggregate** (component 6) -- quantitative -- kg in a m3 mixture -- Input Variable
- **Fine Aggregate** (component 7) -- quantitative -- kg in a m3 mixture -- Input Variable
- **Age** -- quantitative -- Day (1~365) -- Input Variable
- **Concrete compressive strength** -- quantitative -- MPa -- Output Variable

Source of original dataset can be access through this link:
https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength

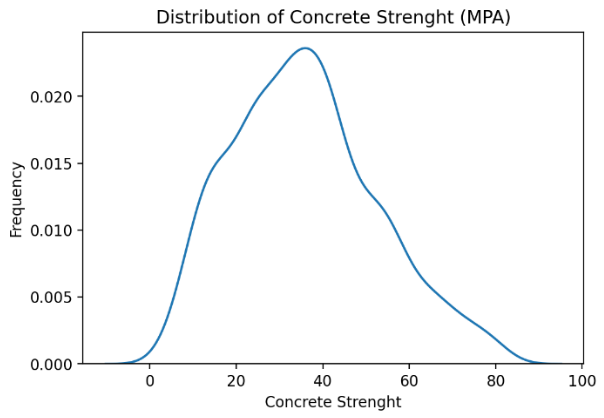
3. Methods

a. Exploratory Data Analysis

We can plot the concrete stenght with frequency. The chart shows the distribution of concrete strength in megapascals (MPA). The x-axis represents the concrete strength, while the y-axis represents the frequency. Here are some insights from the chart:

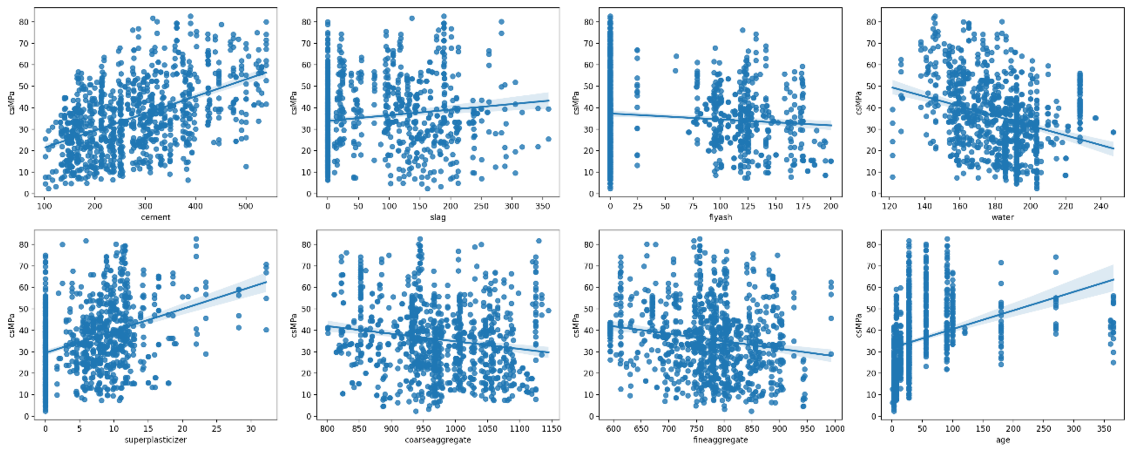
- Skewness: The distribution is skewed to the right, meaning there is a tail in the positive direction. This indicates that there are a few samples with very high concrete strengths.
- Kurtosis: The distribution is leptokurtic, meaning it is more peaked than a normal distribution. This suggests that there are more extreme values (both high and low) than would be expected in a normal distribution.
- Mode: The mode of the distribution is around 40 MPA, which is the most common concrete strength.
- Range: The range of concrete strengths is approximately 0 to 100 MPA.
- Central tendency: The median and mean of the distribution are likely to be higher than the mode due to the right-skewness.

Overall, the chart shows that the concrete strength is highly variable, with a few samples having very high strengths.



We also can plot the csMPa to variable that correlated with concrete strenght. The chart shows scatter plots of compressive strength (csMPa) against various independent variables, along with regression lines. These plots are likely part of a regression analysis to understand how different factors influence concrete strength. Here are some insights from the charts:

- Cement: There seems to be a positive linear relationship between cement content and compressive strength. This suggests that increasing the amount of cement generally leads to stronger concrete.
- Slag: The relationship between slag content and compressive strength is less clear. There might be a slight positive or negative correlation, but it's not as strong as the relationship with cement.
- Flyash: There seems to be a weak negative relationship between flyash content and compressive strength. This could indicate that adding too much flyash might slightly decrease concrete strength.
- Water: There seems to be a negative linear relationship between water content and compressive strength. This is expected, as adding more water can reduce the strength of concrete.
- Superplasticizer: The relationship between superplasticizer content and compressive strength is complex. There might be a slight positive or negative correlation, depending on the specific dosage.
- Coarse aggregate: There seems to be a very weak or no relationship between coarse aggregate content and compressive strength.
- Fine aggregate: There seems to be a very weak or no relationship between fine aggregate content and compressive strength.
- Age: There seems to be a strong positive linear relationship between age and compressive strength. This indicates that concrete strength generally increases over time.
  Overall, the charts suggest that cement content, water content, and age are the most significant factors influencing compressive strength. Other variables like slag, flyash, superplasticizer, coarse aggregate, and fine aggregate might have some influence, but their effects are less pronounced.

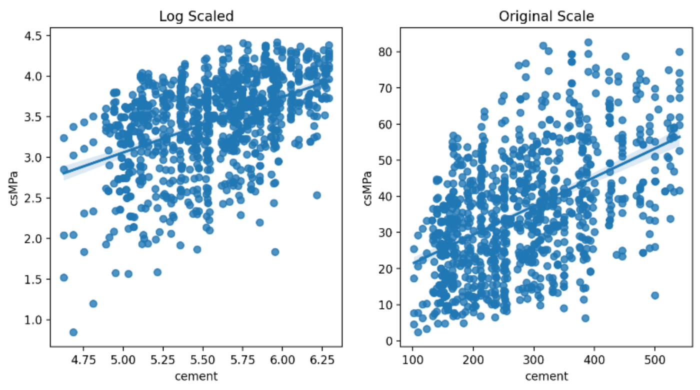


b. Choosing Initial Model

There are lot of varity of model we can use, however ar first we should try the simplest one at first, for example we can use the cement to predict the concrete strenght

Why then scaling needed ?
For easier choose of prior we can perform scaling, for example if we normalized our data into X → Normal(mu=1,sigma=1)

1. Scale Free --> ease at spesifying prior , we can use such standard normal prior
2. Interpretation --> for example we want to scale the outcome as we want to explain / interpret the regression coefficient in terms of log scale which are more intuitive



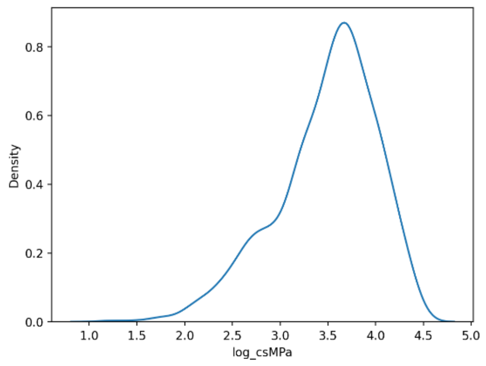
Here are some insights from the charts:

- Log Scale:
  - The relationship between cement content and compressive strength appears to be more linear in log scale. This suggests that the relationship might be exponential or power-law in nature.
  - The regression line in log scale fits the data better than the regression line in original scale, indicating a stronger linear relationship in log scale.
- Original Scale:
  - The relationship between cement content and compressive strength is still apparent, but it's not as linear as in log scale.
  - The regression line in original scale shows a slight curvature, suggesting that the relationship might not be perfectly linear.

$$
\begin{aligned}
log(rides) &\sim Normal(\mu, \sigma) \text{ [likelihood]} \\
\mu &= \alpha + \beta * log(\text{cement}_i) \\
\alpha &\sim Normal(\mu = 2, \sigma = 1) \text{ [prior]} \\
\beta &\sim Normal(\mu = 2, \sigma = 1) \text{ [prior]} \\
\sigma &\sim Expon(\lambda = 2) \text{ [prior]}
\end{aligned}
$$

c. Likelihood

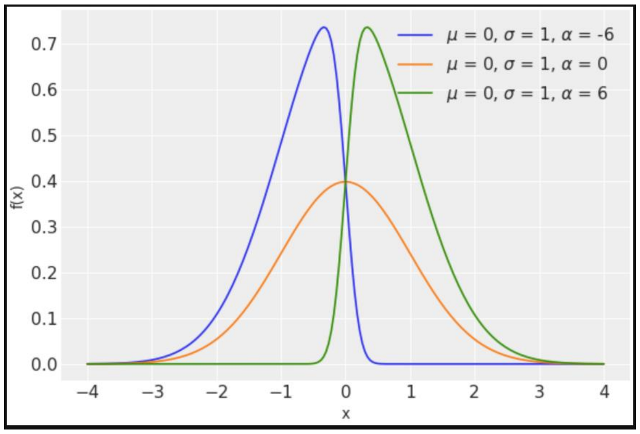
We need to decide log likelihood we are using, we can check logcsmpa distribution



Here are some insights from the plot:

- Distribution: The distribution appears to be approximately normal, with a peak around 3.5. This suggests that the log-transformed compressive strength is normally distributed, which is a common assumption in many statistical models.
- Skewness: The distribution is slightly skewed to the left, indicating that there are a few lower values of log_csMPa that are more extreme than the higher values.
- Kurtosis: The distribution is slightly leptokurtic, meaning it is more peaked than a normal distribution. This suggests that there are more extreme values (both high and low) than would be expected in a normal distribution.
-

Skew Normal Distribution



A Skew Normal distribution is a generalization of the normal (Gaussian) distribution that allows for skewness, meaning it can represent asymmetrical data. While the standard normal distribution is symmetric (bell-shaped) and has a skewness of 0, the skew normal distribution introduces a parameter to control the skewness.
The probability density function (PDF) of a skew normal distribution is given by:

$$f(x;\xi,\omega,\alpha) = \frac{2}{\omega}\phi\left(\frac{x-\xi}{\omega}\right)\Phi\left(\alpha\frac{x-\xi}{\omega}\right)$$

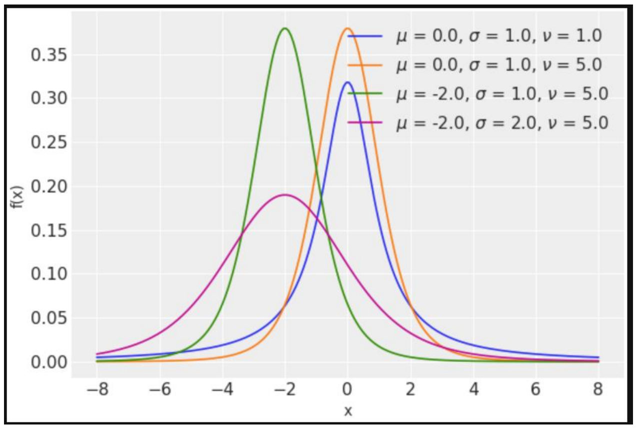
Where:

- ξ (xi) is the location parameter (mean),
- ω>0 (omega) is the scale parameter (similar to standard deviation),
- α (alpha) is the shape parameter (controls skewness),
- ϕ is the standard normal PDF, and
- Φ is the cumulative distribution function (CDF) of the standard normal distribution.

Key Characteristics:

- When α=0, the distribution reduces to the standard normal distribution.
- Positive α values introduce right skewness (longer right tail), while negative α values introduce left skewness (longer left tail).
- It is used in cases where the data exhibits asymmetry, and normal distribution assumptions are not valid.
Applications:
- Modeling financial returns, biological data, and other real-world phenomena that are not symmetric but still resemble the normal distribution.

Student T Distribution



The Student's T distribution is a continuous probability distribution used when estimating the mean of a normally distributed population in situations where the sample size is small and the population's standard deviation is unknown. It is often used in place of the normal distribution when working with smaller sample sizes because it accounts for additional uncertainty in the sample mean.

The probability density function (PDF) for the Student's T distribution is given by:

$$f(x;\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

Where:

- x is the variable,
- v is the degrees of freedom (related to sample size),
- Γ(·) is the gamma function.

Key Characteristics:

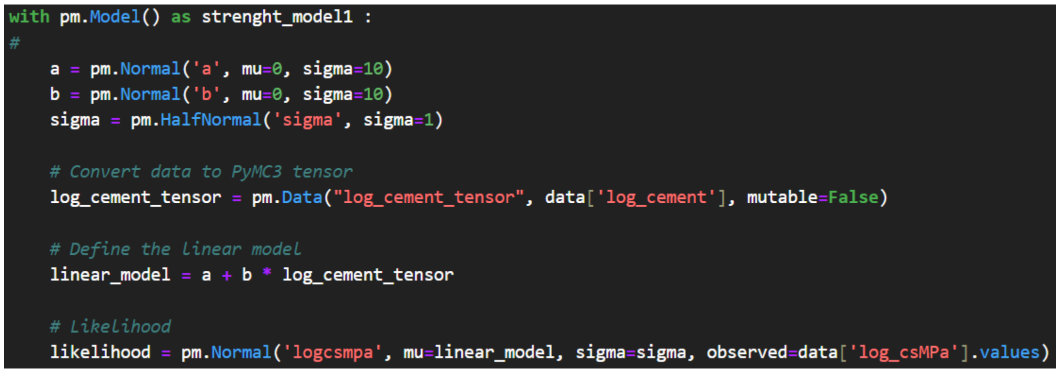
- The degrees of freedom (v) control the shape of the distribution. As v increases, the T-distribution becomes more like a normal distribution.
- For small v, the T-distribution has heavier tails than the normal distribution, meaning it accounts for more variability or uncertainty.
- When v=∞, the T-distribution is exactly the same as the normal distribution.

Applications:

- Used in T-tests (e.g., one-sample, two-sample) to assess hypotheses about means.
- Common in regression analysis when dealing with small sample sizes.
- It is ideal for small datasets where the standard deviation of the population is unknown and needs to be estimated from the data.

Key Differences:

- Skew Normal Distribution: A normal distribution that can be skewed (asymmetric).
- Student's T Distribution: Used for small sample sizes to account for additional variability, resembling a normal distribution but with heavier tails depending on degrees of freedom.



```
with pm.Model() as strenght_model1 :
#
    a = pm.Normal('a', mu=0, sigma=10)
    b = pm.Normal('b', mu=0, sigma=10)
    sigma = pm.HalfNormal('sigma', sigma=1)

    # Convert data to PyMC3 tensor
    log_cement_tensor = pm.Data("log_cement_tensor", data['log_cement'], mutable=False)

    # Define the linear model
    linear_model = a + b * log_cement_tensor

    # Likelihood
    likelihood = pm.Normal('logcsmpa', mu=linear_model, sigma=sigma, observed=data['log_csMPa'].values)
```

This is a simple Bayesian linear regression model that attempts to explain the relationship between log_cement and log_csMPa (likely log-transformed compressive strength of concrete) using PyMC3. Let's break down the different components and provide a more detailed explanation of each:

1. Model Context (with pm.Model() as model):

- The model is created inside a with pm.Model() as strenght_model1: context. In PyMC3, this context allows you to define a probabilistic model.
- strenght_model1 is the name given to the model. It will encapsulate all the distributions and variables used for inference.

2. Prior Distributions:

The prior distributions encode our beliefs about the parameters of the model before seeing the data.

- Intercept (a):

a = pm.Normal('a', mu=0, sigma=10)

This defines a normal prior for the intercept a (the y-axis intercept when log_cement is 0). The intercept represents the expected value of log_csMPa when log_cement is 0. The prior assumes a is normally distributed with a mean of 0 and standard deviation of 10, indicating an initial belief that the intercept is around 0, but with considerable uncertainty (since the standard deviation is large).

- Slope (b):

b = pm.Normal('b', mu=0, sigma=10)

This defines a normal prior for the slope b, which represents the strength of the linear relationship between log_cement (the independent variable) and log_csMPa (the dependent variable). A mean of 0 implies that before seeing the data, we have no strong belief that log_cement has an effect (positive or negative) on log_csMPa. The standard deviation of 10 again reflects uncertainty in the slope.

- Error Term (sigma):

sigma = pm.HalfNormal('sigma', sigma=1)

The standard deviation of the residuals (error term) is modeled with a HalfNormal distribution. The HalfNormal distribution ensures that the standard deviation is strictly positive (because standard deviation can't be negative). The prior has a mean of 0 and a standard deviation of 1, indicating that the model expects small residuals, but it's flexible enough to allow for larger errors.

3. Data Conversion (log_cement_tensor):

log_cement_tensor = pm.Data("log_cement_tensor", data['log_cement'], mutable=False)

Here, the log_cement column from your data is converted into a PyMC3 tensor (which allows PyMC3 to use it internally for sampling). The pm.Data() function is used, which allows the data to be passed into the model and be optionally updated during the inference process. In this case, mutable=False means the data won't change during inference.

4. Linear Model:

linear_model = a + b * log_cement_tensor

This is the equation of the linear model. The dependent variable (log_csMPa) is modeled as a linear function of the independent variable (log_cement). Specifically, log_csMPa is predicted as:

\mu = a + b * log_cement

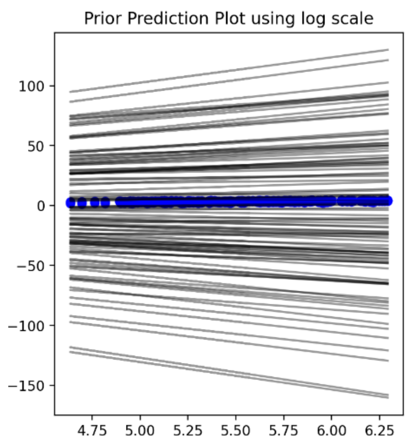
- a is the intercept.
- b is the slope of the line.
- log_cement_tensor is the input data (the independent variable).
  The equation describes how the expected value of log_csMPa changes with respect to log_cement.

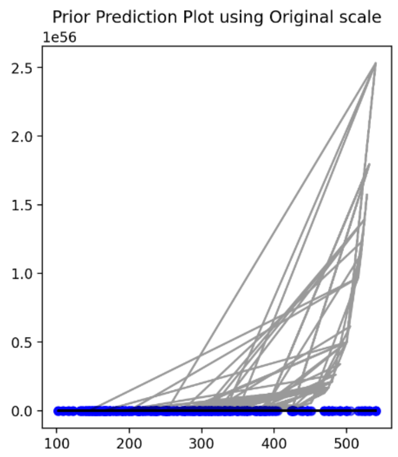
5. Likelihood:

likelihood = pm.Normal('logcsmpa', mu=linear_model, sigma=sigma, observed=data['log_csMPa'].values)

This is the likelihood function, which defines how the observed data (log_csMPa) is generated, given the model parameters. The likelihood assumes that the observed log_csMPa follows a normal distribution with:

- Mean (mu): Given by the linear model (a + b * log_cement_tensor), which is the predicted value of log_csMPa.
- Standard deviation (sigma): Given by the error term, which controls the spread of the residuals.
  The observed=data['log_csMPa'].values part tells PyMC3 that the actual observed values are coming from the data['log_csMPa'] column.





The chart you provided shows two prior prediction plots, one using log scale and the other using original scale. These plots are often used in Bayesian analysis to visualize the relationship between prior beliefs and posterior predictions.
Here are some insights from the plots:

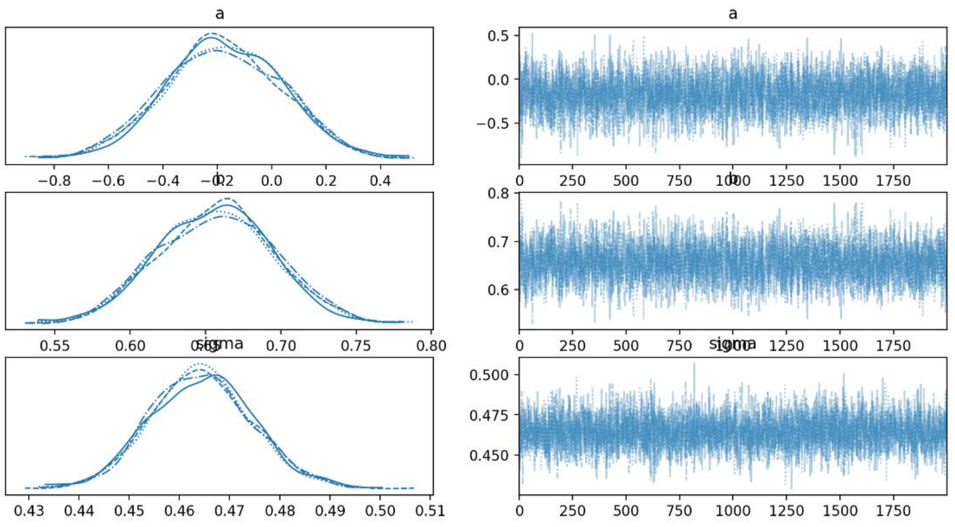
- Log Scale:
  - The prior predictions are more evenly spaced in log scale, which suggests that the prior distribution is more uniform in log space.
  - The posterior predictions (blue dots) are concentrated around a specific range of values, indicating that the data has provided strong evidence for this range.

- Original Scale:
  - The prior predictions are clustered towards the lower values in original scale, suggesting that the prior distribution is skewed towards lower values.
  - The posterior predictions are still concentrated around a specific range of values, but the clustering is less pronounced than in log scale.

  Overall, the plots suggest that the prior distribution is more informative in log scale, and that the data has provided strong evidence for a specific range of values

d. Checking Model Inference Process

One of the core powerful tool that can estimate posterior distribution is our sampling algorithm.
The Inference / Posterior Sampling Process is not simply like `model.fit()` .
Our MCMC algorithm may lead to suboptimal result. Hence it's important to check whether our sampling / trace works properly.
So what is the definition of proper?
To check whether during there is / are problem when fitting the model we can check the trace of each sample generated for our parameters
To do so , we can use `arviz.plot_trace`



Posterior Distributions:

- a: The posterior distribution for a appears to be centered around 0, with a relatively narrow spread. This suggests that the data provides strong evidence for a value of a close to 0.
- sigma: The posterior distribution for sigma is centered around 0.65, with a slightly wider spread than the distribution for a. This indicates that there is some uncertainty in the estimation of sigma, but the data still provides strong evidence for a value around 0.65.
- logsigma: The posterior distribution for logsigma is centered around -0.6, with a similar spread to the distribution for sigma. This suggests that the transformation to log scale has not significantly changed the uncertainty in the estimation of sigma.

Trace Plots:

- The trace plots show the sequence of values sampled for each parameter during the Markov Chain Monte Carlo (MCMC) process. These plots can help assess the convergence of the MCMC algorithm and identify any potential issues.
- The trace plots for a, sigma, and logsigma appear to be mixing well, suggesting that the MCMC algorithm is converging to the posterior distribution. There are no obvious patterns or trends in the plots, which is a good sign.

So what is it ?

Left Figure : Density of our posterior, if you see there is multiple lines , it comes from chain in pymc, chain term it self refer to independent sampling process

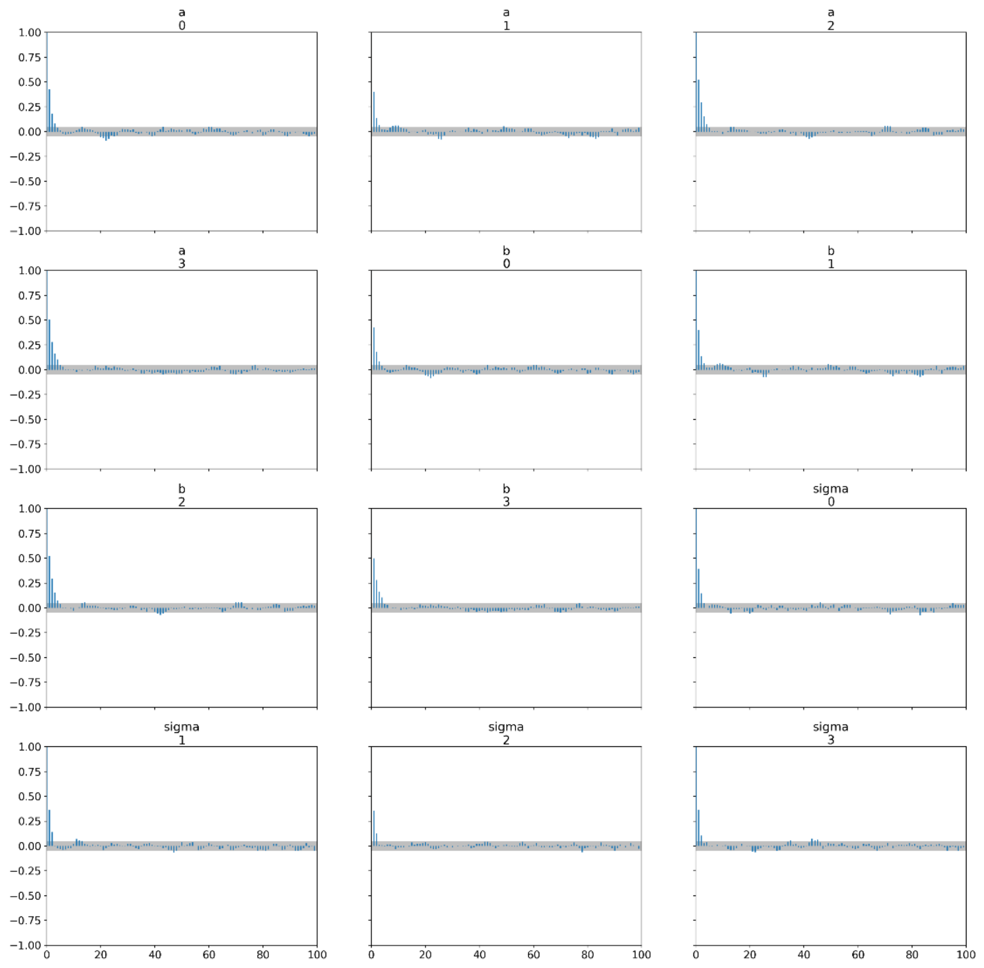
Right Figure : The changes from our parameter values, during posterior sampling process

From our result you might curious is it bad or good ?

Good Definition: During posterior sampling , our parameter value converge towards some value, here we can see for example alpha converge towards zero, or not increase / decrease too exterme

The next thing we could check about our sampling process is that we can checks it autocorrelation

To check autocorrelation we can use `arviz.plot_autocorr(<trace>)`



Here are some insights from the plots:

- a0, a1, a2, a3, b0, b1, b2, b3: The posterior distributions for these parameters are all centered around 0, with relatively narrow spreads. This suggests that the data provides strong evidence for values of these parameters close to 0.

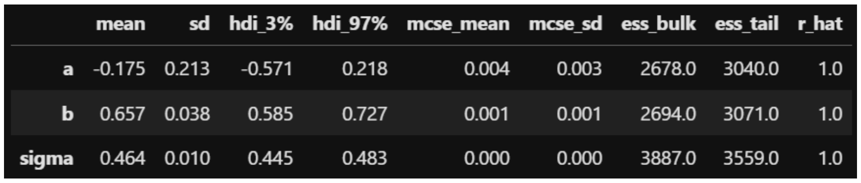
- sigma0, sigma1, sigma2, sigma3: The posterior distributions for these parameters are centered around different values, with varying spreads. This indicates that there is some uncertainty in the estimation of these parameters, but the data still provides evidence for specific ranges of values.

Good/Ideal Condition:

During finding the posterior, we start from initial values, if for example in the next iteration we stuck in the same value over and over again making we never reach optimal posterior, hence good when , autocorrelation are low

Summarizing Model Output

Since your bayesian model return distribution from posterior , it has many range of values



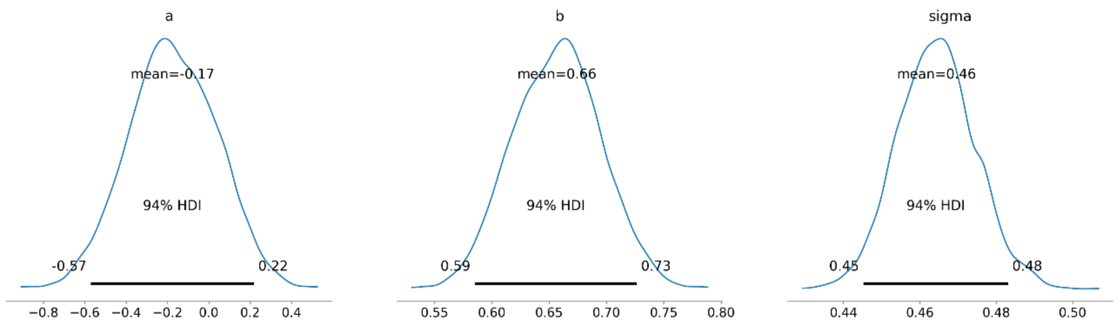
	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a	-0.175	0.213	-0.571	0.218	0.004	0.003	2678.0	3040.0	1.0
b	0.657	0.038	0.585	0.727	0.001	0.001	2694.0	3071.0	1.0
sigma	0.464	0.010	0.445	0.483	0.000	0.000	3887.0	3559.0	1.0

Here's a breakdown of the columns:

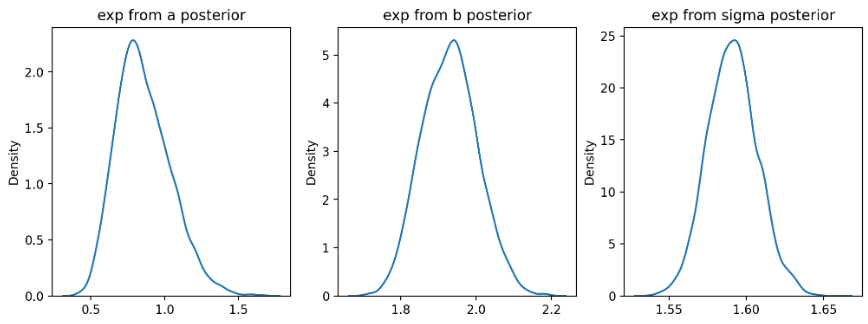
- mean: The average value of the parameter.
- sd: The standard deviation of the parameter.
- hdi_3%: The 3% highest density interval (HDI) for the parameter. This represents a range of values that contains 97% of the posterior probability mass.
- hdi_97%: The 97% highest density interval (HDI) for the parameter. This represents a range of values that contains 3% of the posterior probability mass.
- mcse_mean: The Monte Carlo standard error of the mean. This is a measure of the precision of the estimated mean.
- mcse_sd: The Monte Carlo standard error of the standard deviation. This is a measure of the precision of the estimated standard deviation.
- ess_bulk: The effective sample size for the bulk of the distribution. This is a measure of how efficiently the MCMC algorithm sampled the posterior distribution.
- ess_tail: The effective sample size for the tails of the distribution. This is a measure of how efficiently the MCMC algorithm sampled the extreme values of the posterior distribution.
- r_hat: The R-hat statistic. This is a measure of convergence of the MCMC algorithm. A value of 1 indicates good convergence.

Here are some insights from the table:

- a: The estimated mean of a is -0.175, with a standard deviation of 0.213. The 95% HDI for a is (-0.571, 0.218), indicating that there is a wide range of plausible values for a.
- b: The estimated mean of b is 0.657, with a standard deviation of 0.038. The 95% HDI for b is (0.585, 0.727), indicating that there is a narrower range of plausible values for b compared to a.
- sigma: The estimated mean of sigma is 0.464, with a standard deviation of 0.010. The 95% HDI for sigma is (0.445, 0.483), indicating a relatively narrow range of plausible values for sigma.
- Convergence: The R-hat statistic for all three parameters is 1, indicating that the MCMC algorithm has converged well.
-



- a: The posterior distribution for a is centered around -0.17, with a relatively wide spread. The 94% HDI for a is (-0.57, 0.22), indicating that there is a wide range of plausible values for a.
- b: The posterior distribution for b is centered around 0.66, with a narrower spread than the distribution for a. The 94% HDI for b is (0.59, 0.73), indicating a narrower range of plausible values for b compared to a.
- sigma: The posterior distribution for sigma is centered around 0.46, with a very narrow spread. The 94% HDI for sigma is (0.44, 0.48), indicating a very narrow range of plausible values for sigma.



The chart you provided shows the posterior distributions for three parameters: exp from a posterior, exp from b posterior, and exp from sigma posterior. These distributions represent the range of plausible values for each parameter, given the data and the prior assumptions.

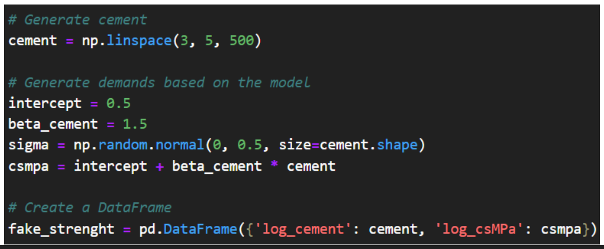
Here are some insights from the plots:

- exp from a posterior: The posterior distribution for exp from a posterior is centered around 1.0, with a relatively wide spread. This suggests that there is a wide range of plausible values for this parameter.
- exp from b posterior: The posterior distribution for exp from b posterior is centered around 2.0, with a narrower spread than the distribution for exp from a posterior. This indicates that there is a narrower range of plausible values for this parameter compared to exp from a posterior.
- exp from sigma posterior: The posterior distribution for exp from sigma posterior is centered around 1.6, with a very narrow spread. This suggests that there is a very narrow range of plausible values for this parameter.

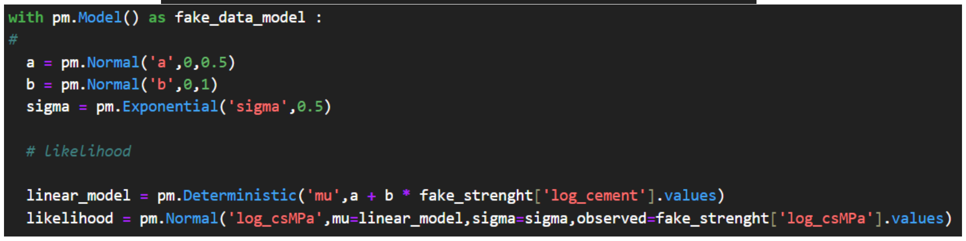
e. Constructing Data Based on Model Fit (Posterior)

The basic idea is to check whether our procedure recovers the correct parameter values when fitting fake data. The procedure is to simulate some fake data from the model with fixed, known parameters and then see whether our method comes close to reproducing the known truth. We can look at point estimates and also the coverage of posterior intervals.

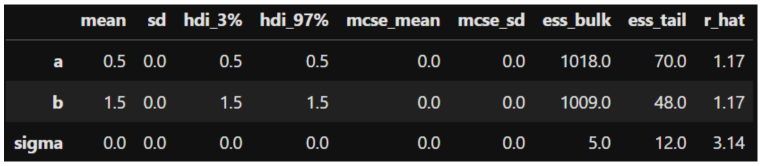
Fitting Model using Fake Data
Generate fake data first



```python
# Generate cement
cement = np.linspace(3, 5, 500)

# Generate demands based on the model
intercept = 0.5
beta_cement = 1.5
sigma = np.random.normal(0, 0.5, size=cement.shape)
csmpa = intercept + beta_cement * cement

# Create a DataFrame
fake_strenght = pd.DataFrame({'log_cement': cement, 'log_csMPa': csmpa})
```



```python
with pm.Model() as fake_data_model :
#
  a = pm.Normal('a',0,0.5)
  b = pm.Normal('b',0,1)
  sigma = pm.Exponential('sigma',0.5)

  # likelihood

  linear_model = pm.Deterministic('mu',a + b * fake_strenght['log_cement'].values)
  likelihood = pm.Normal('log_csMPa',mu=linear_model,sigma=sigma,observed=fake_strenght['log_csMPa'].values)
```



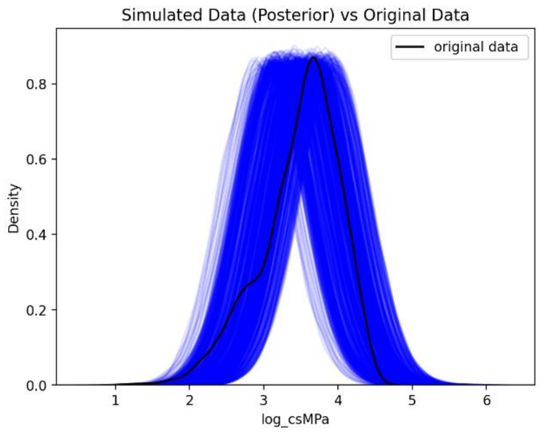
	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a	0.5	0.0	0.5	0.5	0.0	0.0	1018.0	70.0	1.17
b	1.5	0.0	1.5	1.5	0.0	0.0	1009.0	48.0	1.17
sigma	0.0	0.0	0.0	0.0	0.0	0.0	5.0	12.0	3.14

now its time to compare the true parameter and fitted parameter
Actual Parameter
1. intercept = 0.5
2. beta_cement = 1.5
3. sigma = np.random.normal(0, 0.5, size=cement.shape)

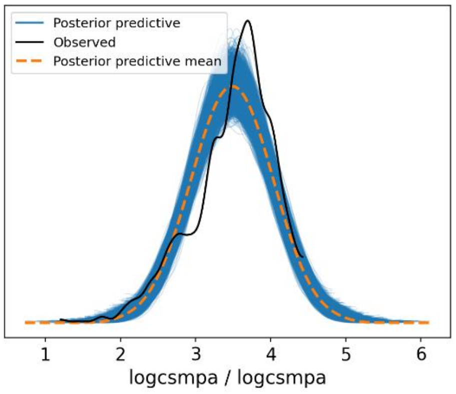
Our Fitted Model Posterior
1. intercept = 0.348
2. beta_cement = 1.538
3. sigma = 0.499

f. Evaluating Model

Posterior Predictive Check
We can do some check based on our model fit, we can simulate dataset from our posterior. First we can sample our posterior from our parameter. The next step is we need to evaluate whether our model working properly



Here are some insights from the plot:
- Similarity: The simulated data (posterior) closely resembles the original data. This suggests that the Bayesian model is able to capture the key features of the original data distribution.
- Variability: The simulated data shows some variability, indicating the uncertainty in the model's predictions.
- Coverage: The simulated data covers the range of the original data, suggesting that the model is able to capture the full range of possible values.



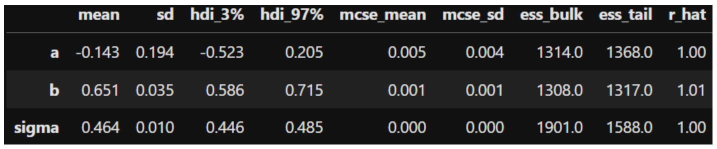
- Posterior Predictive Distribution: The blue lines represent the posterior predictive distribution, which is a distribution of simulated data generated from the model's posterior parameters. This distribution represents the uncertainty in the model's predictions.
- Observed Data: The black line represents the observed data. This is the actual data that the model is trying to fit.
- Posterior Predictive Mean: The orange dashed line represents the posterior predictive mean, which is the average value of the simulated data. This line represents the model's best prediction for the data.

As we can see, our model still have some limitation based on synthetic or generated data, where it did not have long tail on left side, so what is our alternative here ?
The key is **Better at capturing tails**
Our models using simulated data and show its posterior, is quite weird, there are several reson for this
- we use simulated data by taking its mean

g. Cross Validation

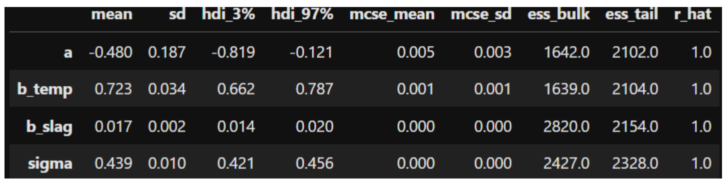
Cross Validation term in machine learning is quite good example to measure model fit, instead of predicting test set directly we can use average metrics from the cross validation , why ? because if we predict test set only there is tendency that the metrics is overestimated

1. Model 1



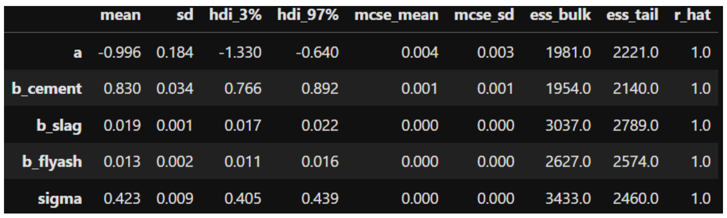
	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a	-0.143	0.194	-0.523	0.205	0.005	0.004	1314.0	1368.0	1.00
b	0.651	0.035	0.586	0.715	0.001	0.001	1308.0	1317.0	1.01
sigma	0.464	0.010	0.446	0.485	0.000	0.000	1901.0	1588.0	1.00

2. Model 2



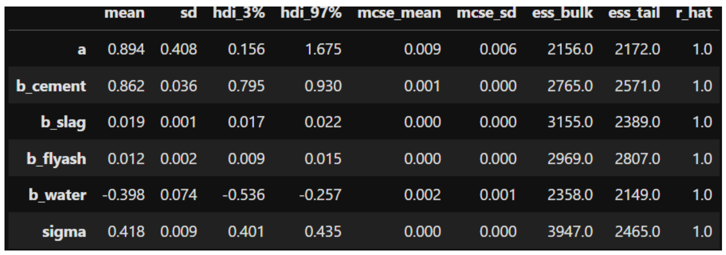
	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a	-0.480	0.187	-0.819	-0.121	0.005	0.003	1642.0	2102.0	1.0
b_temp	0.723	0.034	0.662	0.787	0.001	0.001	1639.0	2104.0	1.0
b_slag	0.017	0.002	0.014	0.020	0.000	0.000	2820.0	2154.0	1.0
sigma	0.439	0.010	0.421	0.456	0.000	0.000	2427.0	2328.0	1.0

3. Model 3



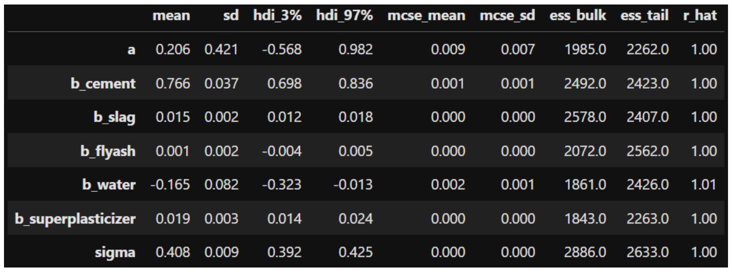
	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a	-0.996	0.184	-1.330	-0.640	0.004	0.003	1981.0	2221.0	1.0
b_cement	0.830	0.034	0.766	0.892	0.001	0.001	1954.0	2140.0	1.0
b_slag	0.019	0.001	0.017	0.022	0.000	0.000	3037.0	2789.0	1.0
b_flyash	0.013	0.002	0.011	0.016	0.000	0.000	2627.0	2574.0	1.0
sigma	0.423	0.009	0.405	0.439	0.000	0.000	3433.0	2460.0	1.0

4. Model 4



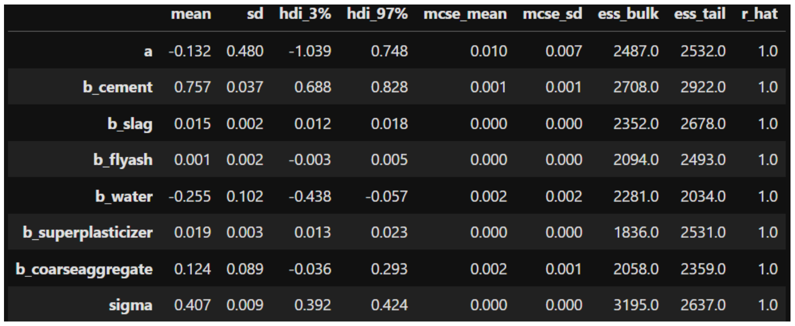
	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a	0.894	0.408	0.156	1.675	0.009	0.006	2156.0	2172.0	1.0
b_cement	0.862	0.036	0.795	0.930	0.001	0.000	2765.0	2571.0	1.0
b_slag	0.019	0.001	0.017	0.022	0.000	0.000	3155.0	2389.0	1.0
b_flyash	0.012	0.002	0.009	0.015	0.000	0.000	2969.0	2807.0	1.0
b_water	-0.398	0.074	-0.536	-0.257	0.002	0.001	2358.0	2149.0	1.0
sigma	0.418	0.009	0.401	0.435	0.000	0.000	3947.0	2465.0	1.0

5. Model 5



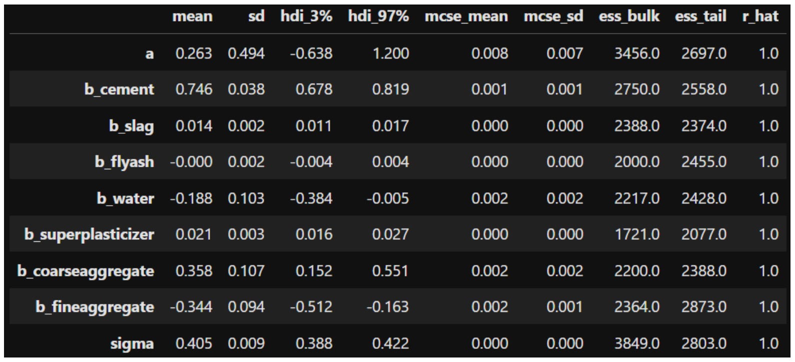
	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a	0.206	0.421	-0.568	0.982	0.009	0.007	1985.0	2262.0	1.00
b_cement	0.766	0.037	0.698	0.836	0.001	0.001	2492.0	2423.0	1.00
b_slag	0.015	0.002	0.012	0.018	0.000	0.000	2578.0	2407.0	1.00
b_flyash	0.001	0.002	-0.004	0.005	0.000	0.000	2072.0	2562.0	1.00
b_water	-0.165	0.082	-0.323	-0.013	0.002	0.001	1861.0	2426.0	1.01
b_superplasticizer	0.019	0.003	0.014	0.024	0.000	0.000	1843.0	2263.0	1.00
sigma	0.408	0.009	0.392	0.425	0.000	0.000	2886.0	2633.0	1.00

6. Model 6



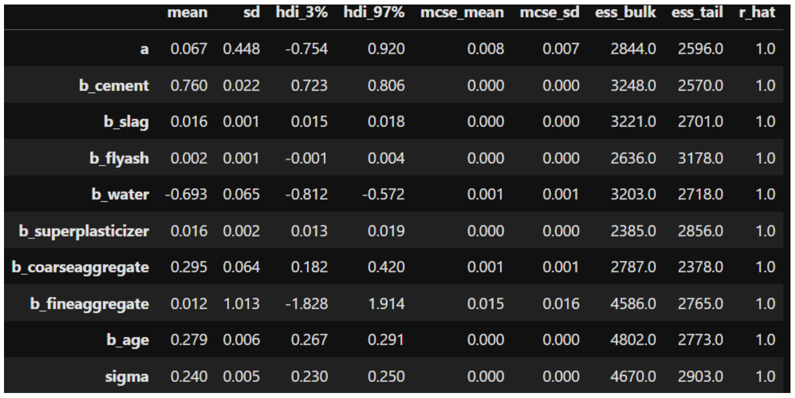
	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a	-0.132	0.480	-1.039	0.748	0.010	0.007	2487.0	2532.0	1.0
b_cement	0.757	0.037	0.688	0.828	0.001	0.001	2708.0	2922.0	1.0
b_slag	0.015	0.002	0.012	0.018	0.000	0.000	2352.0	2678.0	1.0
b_flyash	0.001	0.002	-0.003	0.005	0.000	0.000	2094.0	2493.0	1.0
b_water	-0.255	0.102	-0.438	-0.057	0.002	0.002	2281.0	2034.0	1.0
b_superplasticizer	0.019	0.003	0.013	0.023	0.000	0.000	1836.0	2531.0	1.0
b_coarseaggregate	0.124	0.089	-0.036	0.293	0.002	0.001	2058.0	2359.0	1.0
sigma	0.407	0.009	0.392	0.424	0.000	0.000	3195.0	2637.0	1.0

7. Model 7

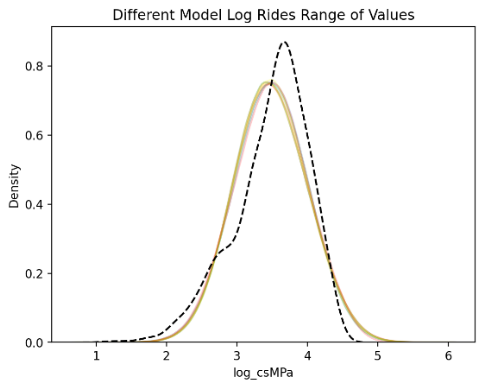


	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a	0.263	0.494	-0.638	1.200	0.008	0.007	3456.0	2697.0	1.0
b_cement	0.746	0.038	0.678	0.819	0.001	0.001	2750.0	2558.0	1.0
b_slag	0.014	0.002	0.011	0.017	0.000	0.000	2388.0	2374.0	1.0
b_flyash	-0.000	0.002	-0.004	0.004	0.000	0.000	2000.0	2455.0	1.0
b_water	-0.188	0.103	-0.384	-0.005	0.002	0.002	2217.0	2428.0	1.0
b_superplasticizer	0.021	0.003	0.016	0.027	0.000	0.000	1721.0	2077.0	1.0
b_coarseaggregate	0.358	0.107	0.152	0.551	0.002	0.002	2200.0	2388.0	1.0
b_fineaggregate	-0.344	0.094	-0.512	-0.163	0.002	0.001	2364.0	2873.0	1.0
sigma	0.405	0.009	0.388	0.422	0.000	0.000	3849.0	2803.0	1.0

8. Model 8



	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a	0.067	0.448	-0.754	0.920	0.008	0.007	2844.0	2596.0	1.0
b_cement	0.760	0.022	0.723	0.806	0.000	0.000	3248.0	2570.0	1.0
b_slag	0.016	0.001	0.015	0.018	0.000	0.000	3221.0	2701.0	1.0
b_flyash	0.002	0.001	-0.001	0.004	0.000	0.000	2636.0	3178.0	1.0
b_water	-0.693	0.065	-0.812	-0.572	0.001	0.001	3203.0	2718.0	1.0
b_superplasticizer	0.016	0.002	0.013	0.019	0.000	0.000	2385.0	2856.0	1.0
b_coarseaggregate	0.295	0.064	0.182	0.420	0.001	0.001	2787.0	2378.0	1.0
b_fineaggregate	0.012	1.013	-1.828	1.914	0.015	0.016	4586.0	2765.0	1.0
b_age	0.279	0.006	0.267	0.291	0.000	0.000	4802.0	2773.0	1.0
sigma	0.240	0.005	0.230	0.250	0.000	0.000	4670.0	2903.0	1.0



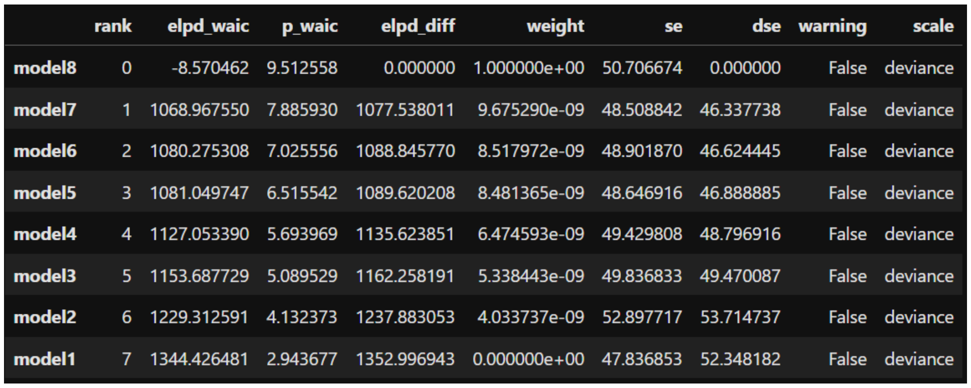
we see that using different model spesification, does not change how the values for posterior, so what do you think by adding complexity ? can you observe any significant changes ?

What if you have numerous models to compare at the same time ?

4. Experiment, Result, Evaluating and Comparing Model

There are many ways to calculate model performance. One of the robust way is to calculate model metrics through Leave One Out Cross Validation

Since most of the models requires log likelihood, we need to calculate it first



	rank	elpd_waic	p_waic	elpd_diff	weight	se	dse	warning	scale
model8	0	-8.570462	9.512558	0.000000	1.000000e+00	50.706674	0.000000	False	deviance
model7	1	1068.967550	7.885930	1077.538011	9.675290e-09	48.508842	46.337738	False	deviance
model6	2	1080.275308	7.025556	1088.845770	8.517972e-09	48.901870	46.624445	False	deviance
model5	3	1081.049747	6.515542	1089.620208	8.481365e-09	48.646916	46.888885	False	deviance
model4	4	1127.053390	5.693969	1135.623851	6.474593e-09	49.429808	48.796916	False	deviance
model3	5	1153.687729	5.089529	1162.258191	5.338443e-09	49.836833	49.470087	False	deviance
model2	6	1229.312591	4.132373	1237.883053	4.033737e-09	52.897717	53.714737	False	deviance
model1	7	1344.426481	2.943677	1352.996943	0.000000e+00	47.836853	52.348182	False	deviance

The table you provided appears to contain information about the performance of different statistical models. The columns include:

- rank: This shows the ranking of the models based on the WAIC score. A lower rank is better because it indicates that the model has a lower WAIC score, meaning it's more likely to generalize well to unseen data.
- elpd_waic: This stands for Expected Log Predictive Density from WAIC. A higher value of elpd_waic is better, as it indicates better predictive performance. However, model8 has a negative value, which suggests it's performing very poorly compared to the other models.
- p_waic: This represents the effective number of parameters. A higher p_waic indicates a more complex model with more parameters. However, complexity can lead to overfitting, so we generally prefer models with a lower p_waic as long as they perform well.
- elpd_diff: This shows the difference in elpd_waic between each model and the top-ranked model (model8 in this case). The best model (model8) has an elpd_diff of 0, and as you go down the list, the differences increase, indicating worse performance relative to model8.
- weight: This is the model weight, which gives a normalized score indicating how likely each model is to be the best given the data. The model with the highest weight is considered the best. In this case, model8 has the highest weight of 1.00000, indicating that it's considered the best among the tested models.
- se: The standard error of the WAIC estimate, which reflects the uncertainty in the model comparison. Lower standard errors are generally better, as they indicate more stable estimates of the WAIC.
- dse: The difference standard error. This is the standard error for the differences in elpd_waic between models. It tells you how much variability there is in the comparison of the models.
- warning: A flag indicating whether any warnings were generated during the computation of WAIC. If True, it would mean there are issues that may affect the reliability of the results. Here, all models have False warnings, indicating no issues.
- scale: This is the scale used to compute WAIC. In this case, all models are using the deviance scale, which is common for model comparison in Bayesian analysis.

Insights:

- Model8 is ranked first, but it has a negative elpd_waic and very high standard error (se), indicating that while it is ranked first, its predictive performance is poor, and it might be highly unstable. The difference between elpd_waic for model8 and the other models is large, suggesting that the other models have better predictive density.
- Model7, Model6, and Model5 have similar performance, with elpd_waic values just over 1000, and lower standard errors compared to model8. These models are ranked right below model8 and seem to have more stable and reliable performance compared to model8, despite the ranking.
- Model1 (ranked last) has the highest elpd_diff, meaning it performs the worst compared to model8. However, it has a significantly higher elpd_waic, suggesting it might actually be a better model in terms of predictive performance than model8 (given the very negative elpd_waic of model8).

Model Complexity:

- Models with lower p_waic (like model1 and model2) are simpler, meaning they have fewer effective parameters.
- Model8 has a very high p_waic, indicating it might be overfitting or unnecessarily complex
.

Conclusion:

- While model8 is ranked first by WAIC and has the highest weight, its poor elpd_waic score and large standard error suggest it might not actually be the best model. Model7, Model6, and Model5 offer more stable performance with much better elpd_waic values, and they may be preferable depending on the application's requirements for stability and predictive power.

5. Reference

- I-Cheng Yeh, "Modeling of strength of high performance concrete using artificial neural networks," Cement and Concrete Research, Vol. 28, No. 12, pp. 1797-1808 (1998).
- I-Cheng Yeh, "Modeling Concrete Strength with Augment-Neuron Networks," J. of Materials in Civil Engineering, ASCE, Vol. 10, No. 4, pp. 263-268 (1998).
- I-Cheng Yeh, "Design of High Performance Concrete Mixture Using Neural Networks," J. of Computing in Civil Engineering, ASCE, Vol. 13, No. 1, pp. 36-42 (1999).
- I-Cheng Yeh, "Prediction of Strength of Fly Ash and Slag Concrete By The Use of Artificial Neural Networks," Journal of the Chinese Institute of Civil and Hydraulic Engineering, Vol. 15, No. 4, pp. 659-663 (2003).
- I-Cheng Yeh, "A mix Proportioning Methodology for Fly Ash and Slag Concrete Using Artificial Neural Networks," Chung Hua Journal of Science and Engineering, Vol. 1, No. 1, pp. 77-84 (2003).
- Yeh, I-Cheng, "Analysis of strength of concrete using design of experiments and neural networks," Journal of Materials in Civil Engineering, ASCE, Vol.18, No.4, pp.597-604 (2006).