Question 1 (30 points). Multilevel models can be useful for modeling time series.

In a time series, the observations cluster by entities that have continuity through time, such as individuals. Since observations within individuals are likely highly correlated, the multilevel structure can help quite a lot.

You'll use the data in *Data/Oxboys.csv*, which is 234 height measurements on 26 boys from an Oxford Boys Club, at 9 different (standardized) ages per boy.

You'll predict height, using age as a predictor, clustered by Subject (individual boy).

- Fit a model with varying intercepts and slopes (on age), clustered by Subject. This means your model will have as many intercepts and slopes as Subjects in the dataset.
- **Define population mean parameters for both the intercepts and slopes** as estimating both parameters will be useful for Question 3.
- Define your model such that you are able to estimate the correlation between intercepts and slopes shared across the Subjects in the dataset.
- Use `pymc.LKJCorr(4)` for the prior of the correlation matrix.
- **There is no need to perform a prior predictive simulation for this assignment.**

Note: Using a centered parameterization for your model should result in a good posterior approximation, so there is no need to define your model using the non-centering trick (shown in lecture) for this model.

```
In [ ]: df = pd.read_csv("Data/Oxboys.csv", delimiter=",")
        df["sub_idx"] = df["Subject"].astype("category").cat.codes
        height = df["height"].values
        age = standardize(df["age"].values)
        sub_idx = df["sub_idx"].values
        num_subjects = df["sub_idx"].nunique()
        with pm.Model() as model:
            a = pm.Normal("a", 150, 10)
            b = pm.Normal("b", 0, 1)
            mu = pm.math.stack([a, b])
            sigma = pm.HalfNormal("sigma", 5)
            # LKJ prior for intercept-slope correlation
            sd_dist = pm.HalfNormal.dist(1)
            chol, correlation, sigmas = pm.LKJCholeskyCov(
                "chol_sub", n=2, eta=4, sd_dist=sd_dist, compute_corr=True
            )
            z = pm.Normal("z", 0, 1, shape=(num_subjects, 2))
            # correlated intercepts and slopes
            res = pm.Deterministic("res", mu + pm.math.dot(z, chol.T))
            subject_a = res[:, 0]
            subject_b = res[:, 1]
            # linear predictor
            mu_height = subject_a[sub_idx] + subject_b[sub_idx] * age
            height_info = pm.Normal("height", mu=mu_height, sigma=sigma, observed=height)
            idata_model = pm.sample(2000, tune=1000,target_accept=0.94, random_seed=42) # do 2000 sampl
```

```
az.summary(idata_model, var_names=["a", "b", "chol_sub_corr", "sigma"])

# Plot Intercepts
```

Plot the parameter estimates for the intercepts and slopes for the Subjects and **interpret** these estimates. **Limit your interpretation to 10 sentences.**

Hint: A good way to represent these parameter estimates is with a bar that represents the high density interval for the posterior distribution of the intercept and slope ***for each Subject****. You can then add the posterior mean estimate of the intercept and slope for each Subject to the plot. Examples for plotting these distributions can be found in the lecture (and accompany code). One such example is the plot of the distribution of the probability of survival in each tank from the introduction to multilevel models lecture. Another example is the plot of the distributions of the district effects on contraception from the lecture on correlated features.*

```
In [ ]: intercepts = extract["chol_subject"][:, :, 0]  # shape: (samples, subjects)
        slopes = extract["chol_subject"][:, :, 1]

        # Plot for Intercepts
        az.plot_forest({"Intercepts": intercepts},
                       credible_interval=0.94,
                       combined=True,
                       ridgeplot_alpha=0.5,
                       r_hat=True)
        plt.title("Posterior Estimates for Subject Intercepts")
        plt.show()

        # Plot for Slopes
        az.plot_forest({"Slopes": slopes},
                       credible_interval=0.94,
                       combined=True,
                       ridgeplot_alpha=0.5,
                       r_hat=True)
        plt.title("Posterior Estimates for Subject Slopes")
        plt.show()
```

The estimates for the intercepts and slopes across different subjects indicate different individual growths. Each subject started at different heights, demonstrated by the various intercepts a subject starts at. Some individuals grew faster than others in a shorter amount of time, for example having steeper slopes for growth. The most dense correlation in the graph is around ~0.5 which indicates that boys who started taller grow at a steeper/quicker rate.

- For which parameters (the intercepts or the slopes) does the model have more certainty? *Base your response to this question on posterior estimates for the population intercept standard deviation and the population slope standard deviation.* **Limit your response to 5 sentences.**

```
In [ ]:
```

The slope estimates are more certain than the intercept estimates. This is evident in the posterior distributions, where the standard deviation for the slopes is smaller than that for the intercepts. A smaller standard deviation indicates less variation across individuals, which shows consistency in growth rates. The larger variation in intercepts suggests more uncertainty in estimating each subject's starting height. So the model provides more precise estimates for growth rates than for initial height levels.

Question 2 (10 points). Now consider the estimated correlation between the varying intercepts and slopes. Can you explain its distribution in the context of the problem? **Limit your response to 10 sentences.**

Hint: Your answer should refer to what the correlation implies about expected heights and growth rates for the subjects.

```
In [ ]:
```

The estimated correlation between the varying intercepts and slopes is mostly positive, the most dense around ~0.5. This could show that subjects who started at a higher height tend to grow at a faster rate. This means that at the start of the study, taller boys usually grow quicker in a shorter time. The positive correlation at ~0.5 means that the differences in height among boys are not closing over time but potentially widening. The variation isn't just in where boys start, but also in how fast they grow. A negative correlation would indicate that shorter boys grow faster. A correlation around ~0 would have suggested a weak to no relationship between starting height and growth rate.

Given some observed heights for a new sample of boys, how would you expect this estimated correlation to influence your predictions for their heights at unobserved ages? Feel free to base your answer off of a visualization of the entire posterior estimate of the correlation or use the mean value from a summary table. However, clearly state which estimated value you are using. **Limit your response to 5 sentences.**

The estimated mean of about ~0.5 between the intercepts and slopes could mean that boys who are recorded to be taller at younger ages continue to grow at a faster rate. This means that when predicting heights at unknown ages, the correlation impacts the values by accounting for steeper growth rates for taller boys. Shorter boys would be expected to grow more slowly based on this relationship.

Question 3 (10 points). Use `scipy.stats.multivariate_normal.rvs()` to simulate a new sample of boys (represented by their observed heights at each age), based upon the posterior mean values of the parameter estimates above. That is, try to simulate varying intercepts and slopes, using the relevant parameter estimates, and then **plot the predicted trends of observed height on age with one trend for each simulated boy you produce**. If you used standardized heights in your model, make sure that the y-axis represents height measured in centimeters.

A sample of 10 simulated boys is plenty, to illustrate the lesson. You can ignore uncertainty in the posterior, just to make the problem a little easier. But if you want to include the uncertainty about the parameters, go for it!

Note that you can construct a variance-covariance matrix to pass as the `cov` argument to `scipy.stats.multivariate_normal.rvs()` using code similar to the following:

```
S = numpy.array( [[ sa**2 , sa*sb*rho] , [sa*sb*rho , sb**2] ])
```

where `sa` is the standard deviation of the first variable, `sb` is the standard deviation of the second variable, and `rho` is the correlation between them.

This question does not require any discussion component.

```
In [ ]: intercept_mean = idata_model.posterior["a"].mean().values
        slope_mean = idata_model.posterior["b"].mean().values
        intercept_sd = idata_model.posterior["sigmas"].sel(sigmas_dim_0=0).mean().values
        slope_sd = idata_model.posterior["sigmas"].sel(sigmas_dim_0=1).mean().values
        rho = idata_model.posterior["correlation"].sel(correlation_dim_0=0, correlation_dim_1=1).mean()

        # Correlation between intercept and slope
        rho = idata_model.posterior["correlation"].sel(correlation_dim_0=0, correlation_dim_1=1).mean()

        cov_matrix = np.array([[intercept_sd**2, intercept_sd*slope_sd*rho],
                               [intercept_sd*slope_sd*rho, slope_sd**2]])
        num_samples = 10 # sample size
        # calculate intercepts and slopes for each boy
        simulated_parameters = multivariate_normal.rvs(mean=[intercept_mean, slope_mean],
                                                       cov=cov_matrix, size=num_samples)
        # calculate heights for each boy at different ages
        ages = np.arange(6, 19)  # ages from 6 to 18 inclusive
        # Plot the trends for each simulated boy
        plt.figure(figsize=(10, 6))
        for i in range(num_samples):
            intercept = simulated_parameters[i, 0]
            slope = simulated_parameters[i, 1]
            heights = intercept + slope * (ages - 6)
            plt.plot(ages, heights, label=f'Boy {i+1}')

        plt.xlabel('Age (years)')
        plt.ylabel('Height (cm)')
        plt.title('Simulated Heights for 10 Boys')
        plt.grid(True)
        plt.show()
```

```
In [ ]: grader.check("q3")
```