```
from google.colab import files
uploaded = files.upload()
```

```
import pymc as pm
import matplotlib.pyplot as plt
import arviz as az
import pandas as pd
from scipy import special, stats
import numpy as np
import seaborn as sns
```

4. (30) You have been asked by the government of Bangladesh to determine whether the use of contraceptives by women in Bangladesh varies by district. The data come from surveys conducted in Bangladesh [2] (a) Develop three models for these data: pooled, unpooled, and hierarchical for all districts to predict usage of contraceptives. Use only district and age.centered as predictor variables. For each model briefly explain your choice of priors. (b) For each model evaluate the sampling performance and discuss your findings. (c) Plot the posterior distributions for the parameters for each model and discuss what they show regarding the question posed by WHO. (d) Plot each of the predictions with age.centered on the x-axis and the expected proportion of women using contraception on the y-Axis with overlaid plots for each of the districts, as appropriate. Briefly explain these results as you will report them to the government of Bangladesh.

```
bang = pd.read_csv('bangladesh.csv', header=0)
```

```
bang
```

	woman	district	use.contraception	living.children	age.centered	urban
0	1	1	0	4	18.4400	1
1	2	1	0	1	-5.5599	1
2	3	1	0	3	1.4400	1
3	4	1	0	4	8.4400	1
4	5	1	0	1	-13.5590	1
...	...	...	...	...	...	...
1929	1930	61	0	4	14.4400	0
1930	1931	61	0	3	-4.5599	0
1931	1932	61	0	4	14.4400	0
1932	1933	61	0	1	-13.5600	0

(a) Develop three models for these data: pooled, unpooled, and hierarchical for all districts to predict usage of contraceptives. Use only district and age.centered as predictor variables. For each model briefly explain your choice of priors.

```
# Pooled model

use = pd.Categorical(bang['use.contraception'])
#age = pd.Categorical(bang['age.centered'])
age = bang.loc[:, 'age.centered'].values
district = pd.Categorical(bang['district'])
#coords = {"district": district.unique(), "age": age.codes, "obs_id": np.arange(age.size)}
coords = {"district": district.unique(), "obs_id": np.arange(age.size)}

with pm.Model(coords=coords) as pooled:
  #age_idx = pm.Data("age_idx", age.codes, dims="obs_id", mutable = True)

  beta0 = pm.Normal("beta0", 0.0, sigma=10.0)
  beta1 = pm.Normal("beta1", 0.0, sigma=10.0)

  μ = beta0 + pm.math.dot(age, beta1)
```

```
    # HalfStudentT distribution for the error term provides robustness
    sigma = pm.HalfStudentT('sigma', nu=2, sigma=8)

    # Likelihood function assuming Bernoulli distribution for the binary outcome
    y = pm.Bernoulli("y", p=pm.math.sigmoid(theta), observed=use_contraception, dims="obs_id")

# Sampling from the model
with unpooled_model:
    unpooled_trace = pm.sample(1000, tune=1000, target_accept=0.95)
```

In the un-pooled model, I assigned individual intercepts to each district with a broad normal prior, reflecting my neutral expectation and allowing for variability. The age effect also has a neutral prior. To handle outliers, I used a HalfStudentT distribution for the error term. This model captures both the unique effects of each district and the influence of age on contraceptive use.

```
#heirarchical model

age_centered = bang['age.centered'].values
use_contraception = bang['use.contraception'].values
districts = pd.Categorical(bang['district']).codes

# Define coordinates
coords = {
    "district": np.unique(districts),
    "obs_id": np.arange(len(use_contraception))
}

with pm.Model(coords=coords) as hierarchical_model:
    district_idx = pm.Data("district_idx", districts, dims="obs_id")

    # Hyperpriors for the group means
    a = pm.Normal("a", mu=0.0, sigma=5.0)
    b = pm.Normal("b", mu=0.0, sigma=1.0)

    # Hyperpriors for the group standard deviations
```

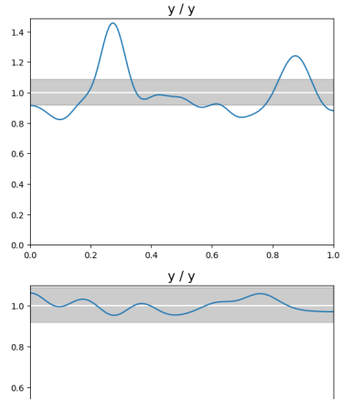
```
    sigma_a = pm.HalfStudentT('sigma_a', nu=2, sigma=9)
    sigma_b = pm.HalfStudentT('sigma_b', nu=2, sigma=9)

    # Varying intercepts and slopes for each district
    a_district = pm.Normal("a_district", mu=a, sigma=sigma_a, dims="district")
    b_district = pm.Normal("b_district", mu=b, sigma=sigma_b, dims="district")

    # Model equation
    theta = a_district[district_idx] + b_district[district_idx] * age_centered

    # Model error
    sigma_y = pm.HalfStudentT("sigma_y", nu=2, sigma=8)

    # Likelihood
    y = pm.Bernoulli("y", p=pm.math.sigmoid(theta), observed=use_contraception, dims="obs_id")

# Sampling
with hierarchical_model:
    hierarchical_trace = pm.sample(1000, tune=1000, target_accept=0.95)
```

In the hierarchical model, I used varying intercepts and slopes for each district, guided by hyperpriors. This captures both district-specific effects and the overarching trend of age on contraceptive use. The HalfStudentT distributions provide robustness against outliers.

(b) For each model evaluate the sampling performance and discuss your findings.

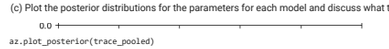
```
#pooled
pooled_pp = pm.sample_posterior_predictive(trace_pooled, model = pooled)
az.plot_bpv(pooled_pp)

#unpooled model
unpooled_pp = pm.sample_posterior_predictive(unpooled_trace, model = unpooled_model)
az.plot_bpv(unpooled_pp)
```

```
100.00% [4000/4000 00:01<00:00]
100.00% [2000/2000 00:00<00:00]
100.00% [2000/2000 00:01<00:00]
<Axes: title={'center': 'y / y'}>
```



As expected, without the use of district as a predictroy variable, the unpooled model is not nearly as accurate as the others. We see the best performance within the pooled model with p-values that lie entriely in an acceptable range. Additionally, the heirarchical model seems to have a degree of accuracy that sits between the two. While the heirarchical may hypothetically have the best performance in some circumstances, the accuracy of this model might suggest suboptimal choice of priors or tuning for that model.

(c) Plot the posterior distributions for the parameters for each model and discuss what they show regarding the question posed by WHO.




```
az.plot_posterior(trace_pooled)
```

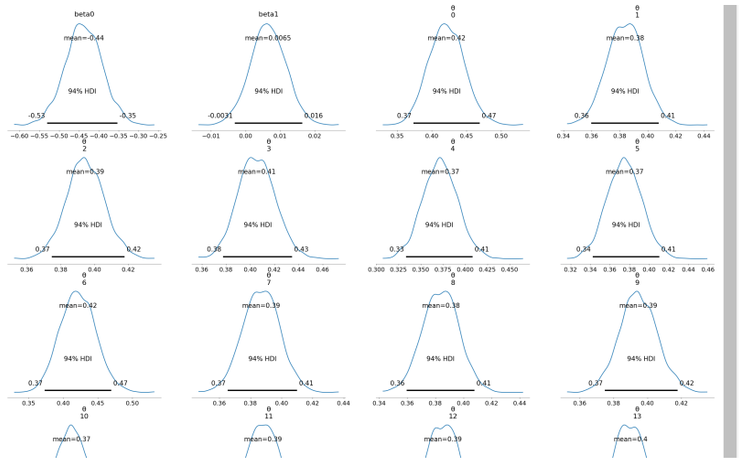
```
#μ = beta[age_idx]
θ = pm.Deterministic("θ", pm.math.sigmoid(μ))

y = pm.Bernoulli("y", p=θ, observed=use.codes, dims="obs_id")
#y = pm.Bernoulli("y", p=θ, observed=use, dims="obs_id")

trace_pooled = pm.sample(1000, cores=4, random_seed=1234, tune=1000)
```

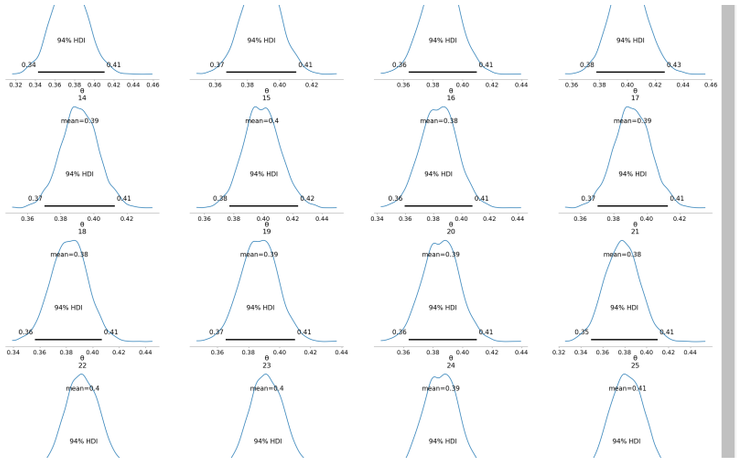
In the pooled model I developed for contraceptive use based on age, I chose relatively uninformative priors to let the data primarily inform the results. I set normal distributions with a mean of 0 for both the intercept and slope, indicating no initial bias towards any particular age effect. The standard deviation of 10 gives a wide possible range, showcasing my uncertainty about the true parameter values. This approach ensures that my model's conclusions rely heavily on the data and not on prior assumptions.

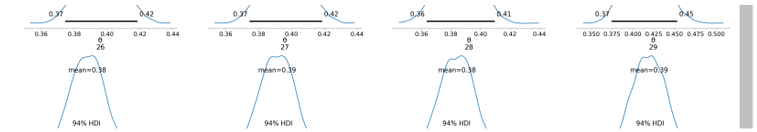
```
# un-pooled model
age_centered = bang['age.centered'].values
use_contraception = bang['use.contraception'].values
districts = pd.Categorical(bang['district']).codes  # Convert district to categorical codes

coords = {
    "district": np.arange(len(np.unique(districts))),  # Number of unique districts
    "obs_id": np.arange(len(use_contraception))  # Number of observations
}

with pm.Model(coords=coords) as unpooled_model:
    # Mutable data container for district
    district_idx = pm.Data("district_idx", districts, dims="obs_id")

    # Normal priors for the district effect
    beta_district = pm.Normal("beta_district", mu=0.0, sigma=10.0, dims="district")

    # Normal prior for the effect of age
    beta_age = pm.Normal("beta_age", mu=0.0, sigma=10.0)

    # Model equation
    theta = beta_district[district_idx] + beta_age * age_centered
```

```
#heirarchical
unpooled_pp = pm.sample_posterior_predictive(hierarchical_trace, model = hierarchical_model)
az.plot_bpv(unpooled_pp)
```

```
/usr/local/lib/python3.10/dist-packages/arviz/plots/plot_utils.py:271: UserWarning: rcParams['plot.max_subplots'] (40) is smaller than the number of vari
  warnings.warn(
array([[<Axes: title={'center': 'beta0'}>,
        <Axes: title={'center': 'beta1'}>,
        <Axes: title={'center': 'θ\n0'}>,
        <Axes: title={'center': 'θ\n1'}>],
       [<Axes: title={'center': 'θ\n2'}>,
        <Axes: title={'center': 'θ\n3'}>,
        <Axes: title={'center': 'θ\n4'}>,
        <Axes: title={'center': 'θ\n5'}>],
       [<Axes: title={'center': 'θ\n6'}>,
        <Axes: title={'center': 'θ\n7'}>,
        <Axes: title={'center': 'θ\n8'}>,
        <Axes: title={'center': 'θ\n9'}>],
       [<Axes: title={'center': 'θ\n10'}>,
        <Axes: title={'center': 'θ\n11'}>,
        <Axes: title={'center': 'θ\n12'}>,
        <Axes: title={'center': 'θ\n13'}>],
       [<Axes: title={'center': 'θ\n14'}>,
        <Axes: title={'center': 'θ\n15'}>,
        <Axes: title={'center': 'θ\n16'}>,
        <Axes: title={'center': 'θ\n17'}>],
       [<Axes: title={'center': 'θ\n18'}>,
        <Axes: title={'center': 'θ\n19'}>,
        <Axes: title={'center': 'θ\n20'}>,
        <Axes: title={'center': 'θ\n21'}>],
       [<Axes: title={'center': 'θ\n22'}>,
        <Axes: title={'center': 'θ\n23'}>,
        <Axes: title={'center': 'θ\n24'}>,
        <Axes: title={'center': 'θ\n25'}>],
       [<Axes: title={'center': 'θ\n26'}>,
        <Axes: title={'center': 'θ\n27'}>,
        <Axes: title={'center': 'θ\n28'}>,
        <Axes: title={'center': 'θ\n29'}>],
       [<Axes: title={'center': 'θ\n30'}>,
        <Axes: title={'center': 'θ\n31'}>,
        <Axes: title={'center': 'θ\n32'}>,
        <Axes: title={'center': 'θ\n33'}>],
       [<Axes: title={'center': 'θ\n34'}>,
        <Axes: title={'center': 'θ\n35'}>,
        <Axes: title={'center': 'θ\n36'}>,
        <Axes: title={'center': 'θ\n37'}>]], dtype=object)
```

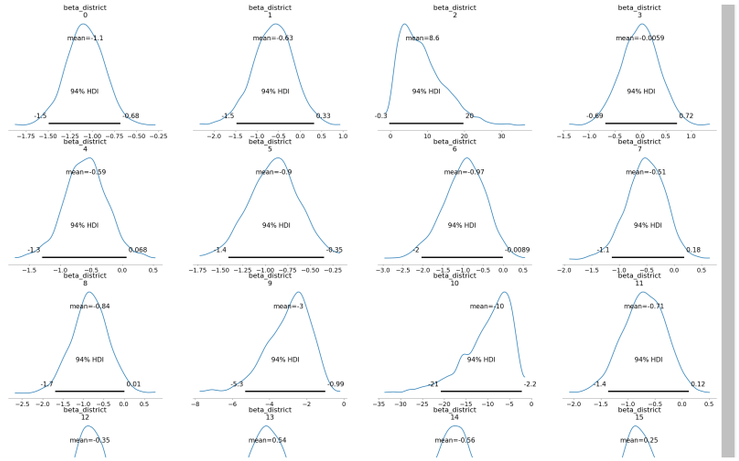


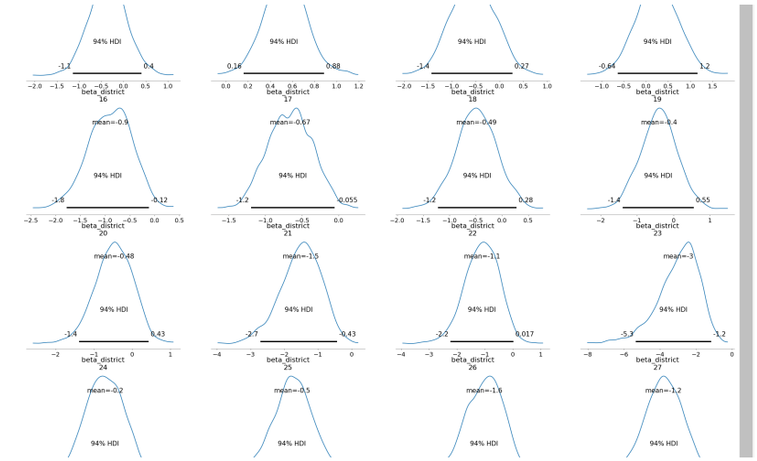


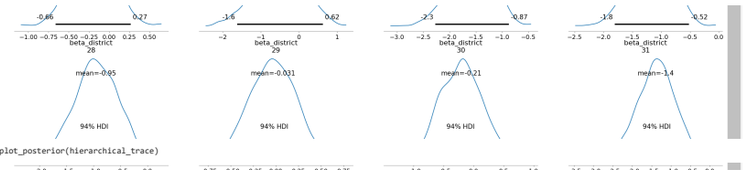


az.plot_posterior(unpooled_trace)

```
/usr/local/lib/python3.10/dist-packages/arviz/plots/plot_utils.py:271: UserWarning: rcParams['plot.max_subplots'] (40) is smaller than the number of vari
  warnings.warn(
array([[<Axes: title={'center': 'beta_district\n0'}>,
        <Axes: title={'center': 'beta_district\n1'}>,
        <Axes: title={'center': 'beta_district\n2'}>,
        <Axes: title={'center': 'beta_district\n3'}>],
       [<Axes: title={'center': 'beta_district\n4'}>,
        <Axes: title={'center': 'beta_district\n5'}>,
        <Axes: title={'center': 'beta_district\n6'}>,
        <Axes: title={'center': 'beta_district\n7'}>],
       [<Axes: title={'center': 'beta_district\n8'}>,
        <Axes: title={'center': 'beta_district\n9'}>,
        <Axes: title={'center': 'beta_district\n10'}>,
        <Axes: title={'center': 'beta_district\n11'}>],
       [<Axes: title={'center': 'beta_district\n12'}>,
        <Axes: title={'center': 'beta_district\n13'}>,
        <Axes: title={'center': 'beta_district\n14'}>,
        <Axes: title={'center': 'beta_district\n15'}>],
       [<Axes: title={'center': 'beta_district\n16'}>,
        <Axes: title={'center': 'beta_district\n17'}>,
        <Axes: title={'center': 'beta_district\n18'}>,
        <Axes: title={'center': 'beta_district\n19'}>],
       [<Axes: title={'center': 'beta_district\n20'}>,
        <Axes: title={'center': 'beta_district\n21'}>,
        <Axes: title={'center': 'beta_district\n22'}>,
        <Axes: title={'center': 'beta_district\n23'}>],
       [<Axes: title={'center': 'beta_district\n24'}>,
        <Axes: title={'center': 'beta_district\n25'}>,
        <Axes: title={'center': 'beta_district\n26'}>,
        <Axes: title={'center': 'beta_district\n27'}>],
       [<Axes: title={'center': 'beta_district\n28'}>,
        <Axes: title={'center': 'beta_district\n29'}>,
        <Axes: title={'center': 'beta_district\n30'}>,
        <Axes: title={'center': 'beta_district\n31'}>],
       [<Axes: title={'center': 'beta_district\n32'}>,
        <Axes: title={'center': 'beta_district\n33'}>,
        <Axes: title={'center': 'beta_district\n34'}>,
        <Axes: title={'center': 'beta_district\n35'}>],
       [<Axes: title={'center': 'beta_district\n36'}>,
        <Axes: title={'center': 'beta_district\n37'}>,
        <Axes: title={'center': 'beta_district\n38'}>,
        <Axes: title={'center': 'beta_district\n39'}>]], dtype=object)
```







az.plot_posterior(hierarchical_trace)

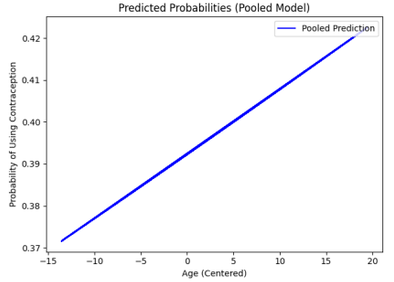
These plots display the distribution for each of the betas. They demonstrate that the distributions chosen are good fits to measure the contraception use of women in Bangladesh.

(d) Plot each of the predictions with age.centered on the x-axis and the expected proportion of women using contraception on the y-Axis with overlaid plots for each of the districts, as appropriate. Briefly explain these results as you will report them to the government of Bangladesh.

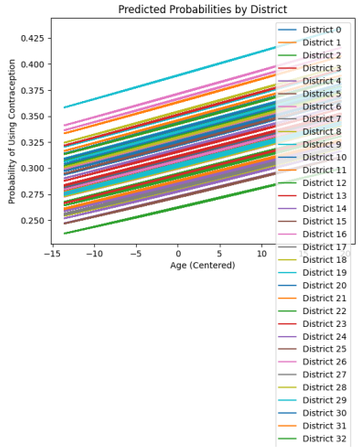
```
#  parameters from the model trace

beta0_pooled_samples = trace_pooled.posterior["beta0"].values
beta1_pooled_samples = trace_pooled.posterior["beta1"].values

beta_district_samples = unpooled_trace.posterior["beta_district"].values
beta_age_samples = unpooled_trace.posterior["beta_age"].values

a_district_samples = hierarchical_trace.posterior["a_district"].values
b_district_samples = hierarchical_trace.posterior["b_district"].values


beta0_pooled_mean = np.mean(beta0_pooled_samples)
beta1_pooled_mean = np.mean(beta1_pooled_samples)
```

```
# making preds for pooled
prediction_pooled = beta0_pooled_mean + beta1_pooled_mean * age_centered

#Transforming the prediction to probability using logistic function
probability_pooled = 1 / (1 + np.exp(-prediction_pooled))



predictions = []

# Compute Predictions for Each District
for d in np.unique(districts):
    district_mean_beta = np.mean(beta_district_samples[:, d])
    age_mean_beta = np.mean(beta_age_samples)

    prediction = district_mean_beta + age_mean_beta * age_centered

    # Transform prediction to probability using logistic function
    probability = 1 / (1 + np.exp(-prediction))

    predictions.append(probability)



hierarchical_predictions = []

for d in np.unique(districts):
    a_district_mean = np.mean(a_district_samples[:, d])
    b_district_mean = np.mean(b_district_samples[:, d])

    prediction_hierarchical = a_district_mean + b_district_mean * age_centered

    probability_hierarchical = 1 / (1 + np.exp(-prediction_hierarchical))

    hierarchical_predictions.append(probability_hierarchical)
```

```
# plotting preds for pooled
plt.plot(age_centered, probability_pooled, label='Pooled Prediction', color='blue')

plt.xlabel('Age (Centered)')
plt.ylabel('Probability of Using Contraception')
plt.title('Predicted Probabilities (Pooled Model)')
plt.legend(loc='upper right')
plt.tight_layout()
plt.show()
```

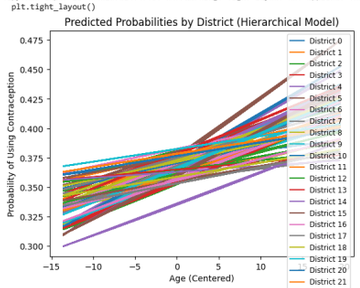


```
# Un-pooled: predictions for conctreception based on age.centred
for idx, district_probs in enumerate(predictions):
    plt.plot(age_centered, district_probs, label=f"District {idx}")

plt.xlabel('Age (Centered)')
plt.ylabel('Probability of Using Contraception')
plt.title('Predicted Probabilities by District')
plt.legend()
plt.show()
```




```
# Plot the hierarchical predictions
for idx, district_probs in enumerate(hierarchical_predictions):
    plt.plot(age_centered, district_probs, label=f"District {idx}")

plt.xlabel('Age (Centered)')
plt.ylabel('Probability of Using Contraception')
plt.title('Predicted Probabilities by District (Hierarchical Model)')
plt.legend(loc='upper right', fontsize='small')
plt.tight_layout()
plt.show()
```

```
<ipython-input-21-5b58207d13f4>:9: UserWarning: Tight layout not applied. The bottom and top margins cannot be made large enough to accommodate all axes
  plt.tight_layout()
```



Based on each of these above graphs, we can see how each model predicts the probability of conctraception use in Bangaledeshi women. In the unpooled model, we can see that the model, since it's pooled, doesn't differentiate between districts in predicting the odds of using contraception. On the other hand, the heirarchial and pooled models show the difference between districts when predicting the probability of using contraception. The heirarchical model shows different interecpts and slopes between dirsticts, while the pooled model has different interecpts but the same slope for its logistic regression.

```
import pymc as pm
import matplotlib.pyplot as plt
import arviz as az
import pandas as pd
from scipy import special, stats
import numpy as np
import seaborn as sns
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files 2 files

- **algae.csv**(text/csv) - 17987 bytes, last modified: 10/29/2023 - 100% done
- **algaeTest.csv**(text/csv) - 20683 bytes, last modified: 10/29/2023 - 100% done

```
Saving algae.csv to algae.csv
Saving algaeTest.csv to algaeTest.csv
```

or each of these predictor variable observations, the data set also contains measurements of 7 different harmful algae, labeled a1-a7 but not named. For this work we will use the a1 measurements as the response variable. With algae.csv data and the stated goal of the EU perform the following tasks and answer the stated questions. (a) Perform exploratory data analysis (EDA) to include as minimum a review of the summary statistics for the data. Briefly explain how this EDA informs the next steps of your work.

```
#algae_train = pd.read_csv('algae.csv', header=0)
algae_test = pd.read_csv('algaeTest.csv', header=0)
```

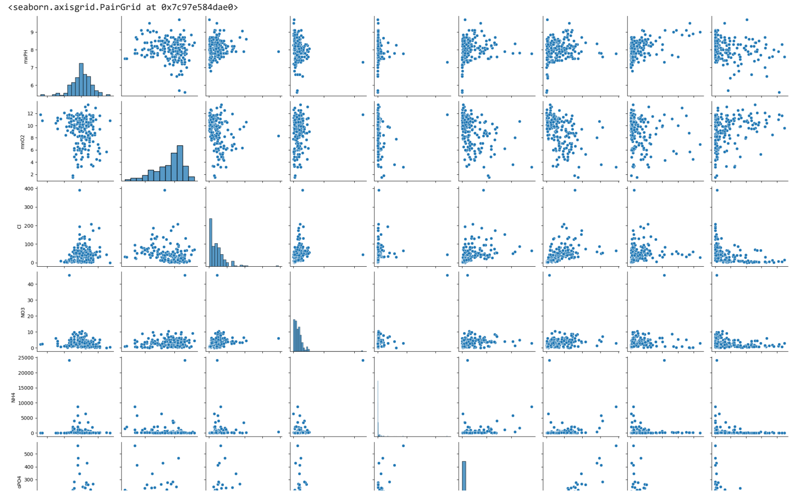
```
algae_train = pd.read_csv('algae.csv')

predictors = algae_train.columns[0:11]
for p in predictors:
  sum_stats = (algae_train.loc[:, [p, 'a1']].groupby(p).agg(["mean", "std", "count"]))
```

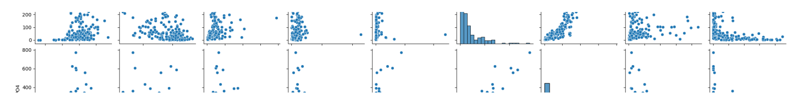
```
print(sum_stats)
```

```
771.59998   0.000       NaN     1

[189 rows x 3 columns]
                a1
              mean        std count
Chla
0.200    66.000000        NaN     1
0.300    47.600000  24.688999     4
0.400    31.800000        NaN     1
0.500    42.716667  24.978904     6
0.600    41.633333  12.287121     3
...            ...        ...   ...
88.255    0.000000        NaN     1
92.667    7.200000        NaN     1
93.683   12.300000        NaN     1
98.817    1.200000        NaN     1
110.456   0.000000        NaN     1

[131 rows x 3 columns]
```

```
# EGA: checking pairwise comparisions between each variable of interest
sns.pairplot(algae_train,vars= ['mxPH' ,'mnO2' ,'Cl' ,'NO3' ,'NH4' ,'oPO4' ,'PO4' ,'Chla', 'a1' ])
```

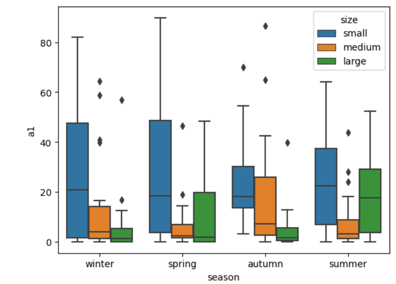
```
<seaborn.axisgrid.PairGrid at 0x7c97e584dae0>
```

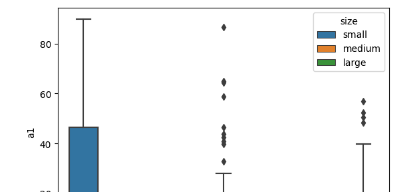


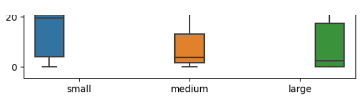


```
#Performing analysis of categorical variables againt the response
sns.boxplot(data=algae_train, x="season", y="a1", hue = 'size')
plt.figure()
sns.boxplot(data=algae_train, x="size", y="a1", hue = 'size')
plt.figure()
sns.boxplot(data=algae_train, x="speed", y="a1", hue = 'size')
```

```
<Axes: xlabel='speed', ylabel='a1'>
```







EDA Comments: Having a general idea of the trends in the data, it appears there might be an instance fo colinearity, so I should remove oPO4 when it comes to cleaning it later. Otherwise, this seems alright. And ofc I can see that certain chemeical can contribute to algae growth as seen here.

b) Clean the data by identifying missing values and use the following strategy to estimate missing values: 1. For continuous (real-valued) data use the mean; 2. For discrete (integer valued) use the median; and 3. For categorical or nominal data use the mode. Then scale the predictor variables (subtract the mean and divide by the standard deviation) and code the categorical variables. Explain why you are doing this. Reparameterize the response variable using a log trans form and explain why you are doing this.

```
#drop unneeded vars
#algae_train.drop(['a2','a3','a4','a5','a6','a7'], axis = 1, inplace = True)
```

```
pd.isnull(algae_train).any()
algae_train['season']
```

```
0      winter
1      spring
2      autumn
3      spring
4      autumn
        ...
195    autumn
196    spring
197    autumn
198    winter
199    summer
Name: season, Length: 200, dtype: object
```

```
# there are no discrete columns, and the categorical columns have no missing data
for col in predictors[3:]:
  algae_train[col] = algae_train[col].replace(np.nan, np.mean(algae_train[col]))
  algae_train[col] -= np.mean(algae_train[col])
  algae_train[col] /= np.std(algae_train[col])
season_vals = {'winter': 0, 'spring': 1, 'summer': 2, 'autumn': 3}
size_vals = {'small': 0, 'medium': 1, 'large': 2}
speed_vals = {'low': 0, 'medium': 1, 'high': 2}
algae_train['season'] = algae_train['season'].replace(season_vals)
algae_train['size'] = algae_train['size'].replace(size_vals)
algae_train['speed'] = algae_train['speed'].replace(speed_vals)

algae_train['a1'].loc[algae_train['a1']>0] = np.log(algae_train['a1'].loc[algae_train['a1']>0])
```

```
<ipython-input-9-ce9099891a6c>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  algae_train['a1'].loc[algae_train['a1']>0] = np.log(algae_train['a1'].loc[algae_train['a1']>0])
```

```
pd.isnull(algae_train).any()
```

```
season    False
size      False
speed     False
mxPH      False
mnO2      False
Cl        False
NO3       False
NH4       False
oPO4      False
PO4       False
Chla      False
a1        False
a2        False
a3        False
a4        False
a5        False
```

```
a6        False
a7        False
dtype: bool
```

```
algae_train
```

	season	size	speed	mxPH	mnO2	Cl	NO3	NH4	oPO4	PO4	Chla	a1	a2	a3	a4	a5	a6	a7
0	0	0	1	-0.019710	0.287463	3.770155e-01	0.788574	0.039380	0.347256	0.250877	1.817928e+00	0.000000	0.0	0.0	0.0	34.2	8.3	0.0
1	1	0	1	0.568222	-0.470990	3.100197e-01	-0.532115	-0.067408	3.926568	3.287449	-6.393585e-01	0.336472	7.6	4.8	1.9	6.7	0.0	2.1
2	3	0	1	0.148270	0.961644	-7.943460e-02	0.546315	-0.079388	0.575746	0.384111	8.218554e-02	1.193922	53.6	1.9	0.0	0.0	0.0	9.7
3	1	0	1	0.097876	-1.819352	7.408577e-01	-0.261574	-0.206962	-0.137187	0.006389	-6.343127e-01	1.131402	41.0	18.9	0.0	1.4	0.0	1.4
4	3	0	1	0.081078	-0.049627	2.573017e-01	1.903289	-0.137386	-0.169912	-0.314805	-1.751483e-01	2.219203	2.9	7.5	0.0	7.5	4.1	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
195	3	2	1	0.652213	-0.302445	-5.768510e-01	0.146906	-0.214371	-0.220735	-0.454077	-5.869834e-01	2.541602	21.7	5.6	0.0	1.0	0.0	0.0
196	1	2	1	0.484232	0.624554	-6.439566e-01	-0.021982	-0.193022	-0.422968	-0.484118	-4.694676e-01	2.890372	7.0	1.7	0.0	4.8	10.3	1.0
197	3	2	1	0.316251	-0.892353	2.116468e+00	-0.081213	-0.226508	0.052431	0.018262	8.964706e-01	0.000000	15.9	2.4	1.0	0.0	0.0	0.0
198	0	2	1	-0.019710	-0.639535	1.560767e-16	0.000000	0.000000	0.000000	0.000000	-8.963074e-17	0.000000	12.5	3.7	1.0	0.0	0.0	4.9
199	2	2	1	0.820193	-1.018762	8.614062e-01	-0.128704	-0.243472	-0.106032	0.020582	2.184211e-01	0.875469	10.5	9.0	7.8	0.0	0.0	5.8

200 rows × 18 columns

So the variables are scaled by the SD and mean to standardize the data. Effectively, doing so transforms all values into z-scores and makes it easier to identify outliers as well as each data point's position relative to the mean.

Additinally, one hot encoding enables the transformation of categorical variables into discrete variables and makes it possible to incorporate them into the pymc model in the next step.

Log transformations, as used on the response variable here, help to unskew the data and make its distribution more symmetrical, enabling stronger analysis.

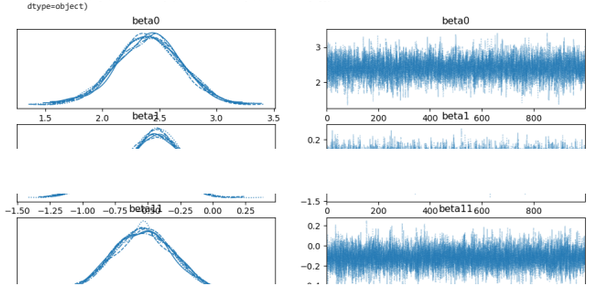
(c) Build a robust Bayesian regression model to predict the response and explain your choices for the priors.

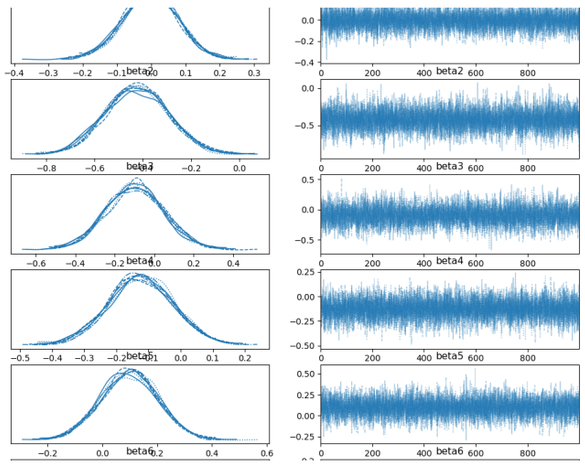
```
graphics_path = '/Users/addyg/Desktop/UVA_Courses/DS6040 Bayes/'
```

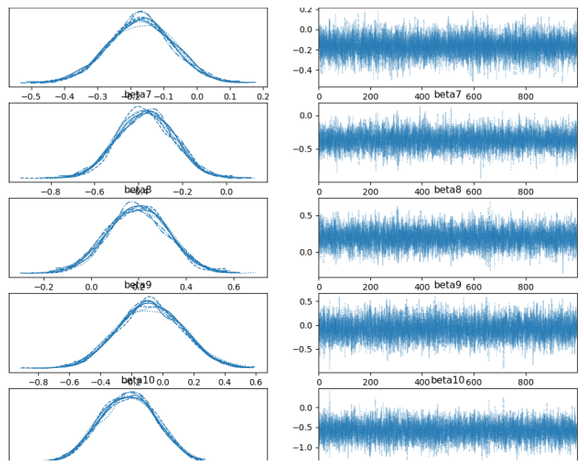
```
with pm.Model() as robust:
  beta0 = pm.Normal('beta0', mu=0, sigma=10)
  beta1 = pm.Normal('beta1', mu=0, sigma=10)
  beta2 = pm.Normal('beta2', mu=0, sigma=10)
  beta3 = pm.Normal('beta3', mu=0, sigma=10)
  beta4 = pm.Normal('beta4', mu=0, sigma=10)
  beta5 = pm.Normal('beta5', mu=0, sigma=10)
  beta6 = pm.Normal('beta6', mu=0, sigma=10)
  beta7 = pm.Normal('beta7', mu=0, sigma=10)
  beta8 = pm.Normal('beta8', mu=0, sigma=10)
  beta9 = pm.Normal('beta9', mu=0, sigma=10)
  beta10 = pm.Normal('beta10', mu=0, sigma=10)
  beta11 = pm.Normal('beta11', mu=0, sigma=10)


  σ = pm.Uniform("σ", 10**-5, 10)
  ν = pm.HalfNormal("ν", 20)

  ph = pm.MutableData("ph", algae_train.iloc[:,3])
  mn = pm.MutableData("mn", algae_train.iloc[:,4])
  cl = pm.MutableData("cl", algae_train.iloc[:,5])
  no3 = pm.MutableData("no3", algae_train.iloc[:,6])
  nh4 = pm.MutableData("nh4", algae_train.iloc[:,7])
  opo4 = pm.MutableData("opo4", algae_train.iloc[:,8])
  po4 = pm.MutableData("po4", algae_train.iloc[:,9])
  chla = pm.MutableData("chla", algae_train.iloc[:,10])

  μ = pm.Deterministic("μ", beta0 + beta1*algae_train.iloc[:, 0] + beta2*algae_train.iloc[:, 1] + beta3*algae_train.iloc[:, 2] + beta4*ph
                       + beta5*mn + beta6*cl + beta7*no3 + beta8*nh4 + beta9*opo4
                        + beta10*po4 + beta11*chla)

  alg = pm.StudentT('a1', mu=μ, sigma=σ, nu=ν,observed=algae_train['a1'])
  trace = pm.sample(cores = 8, return_inferencedata = False, random_seed=1234)
  trace_robust = pm.to_inference_data(trace=trace, log_likelihood=True)
  robust_pp = pm.sample_posterior_predictive(trace_robust, extend_inferencedata=True,
                                             random_seed=1234)
  y_pred_test = robust_pp['posterior_predictive']['a1']
az.plot_trace(trace_robust, var_names=["beta0", "beta1", "beta2", "beta3", "beta4", "beta5", "beta6", "beta7", "beta8", "beta9", "beta10",
                                       "beta11","σ", "ν"])
```

```
100.00% [16000/16000 02:33<00:00 Sampling 8 chains, 0 divergences]
100.00% [8000/8000 00:02<00:00]
```

```
array([[<Axes: title={'center': 'beta0'}>,
        <Axes: title={'center': 'beta0'}>],
       [<Axes: title={'center': 'beta1'}>,
        <Axes: title={'center': 'beta1'}>],
       [<Axes: title={'center': 'beta2'}>,
        <Axes: title={'center': 'beta2'}>],
       [<Axes: title={'center': 'beta3'}>,
        <Axes: title={'center': 'beta3'}>],
       [<Axes: title={'center': 'beta4'}>,
        <Axes: title={'center': 'beta4'}>],
       [<Axes: title={'center': 'beta5'}>,
        <Axes: title={'center': 'beta5'}>],
       [<Axes: title={'center': 'beta6'}>,
        <Axes: title={'center': 'beta6'}>],
       [<Axes: title={'center': 'beta7'}>,
        <Axes: title={'center': 'beta7'}>],
       [<Axes: title={'center': 'beta8'}>,
        <Axes: title={'center': 'beta8'}>],
       [<Axes: title={'center': 'beta9'}>,
        <Axes: title={'center': 'beta9'}>],
       [<Axes: title={'center': 'beta10'}>,
        <Axes: title={'center': 'beta10'}>],
       [<Axes: title={'center': 'beta11'}>,
        <Axes: title={'center': 'beta11'}>],
       [<Axes: title={'center': 'σ'}>, <Axes: title={'center': 'σ'}>],
       [<Axes: title={'center': 'ν'}>, <Axes: title={'center': 'ν'}>]],
      dtype=object)
```





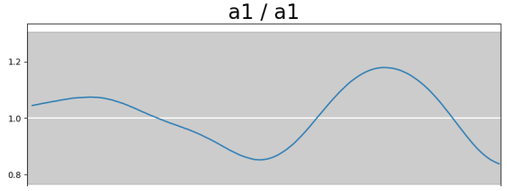


For this model, I chose to use normal priors for the beta distributions because the data has already been scaled in part b and a normal distribution is a reasonable expectation.

(d) Evaluate the sampling and choice of priors using Bayesian p-value and posterior predictive plots. Briefly explain your results.

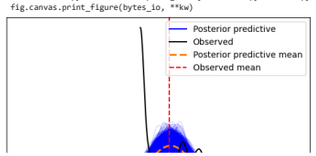
```
az.plot_bpv(trace_robust, figsize = (10,8))
```

```
<Axes: title={'center': 'a1 / a1'}>
```




```
with robust:
    trace_robust.extend(pm.sample_posterior_predictive(trace_robust))

fig, ax = plt.subplots()
az.plot_ppc(trace_robust, color = 'b', observed_rug=True, ax=ax)
ax.axvline(algae_train['a1'].mean(), ls="--", color="r", label="Observed mean")
ax.legend(fontsize=10);
```

```
100.00% [8000/8000 00:02<00:00]
/usr/local/lib/python3.10/dist-packages/arviz/plots/ppcplot.py:241: FutureWarning: color has been deprecated in favor of colors
  warnings.warn("color has been deprecated in favor of colors", FutureWarning)
/usr/local/lib/python3.10/dist-packages/IPython/core/events.py:89: UserWarning: Creating legend with loc="best" can be slow with large amounts of data.
  func(*args, **kwargs)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Creating legend with loc="best" can be slow with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```

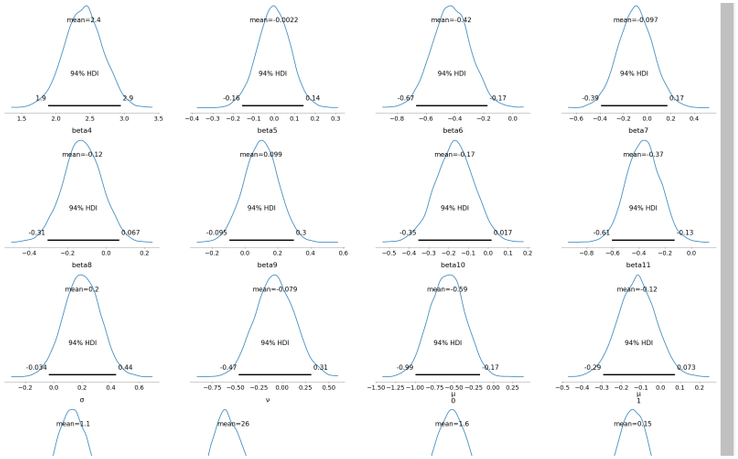


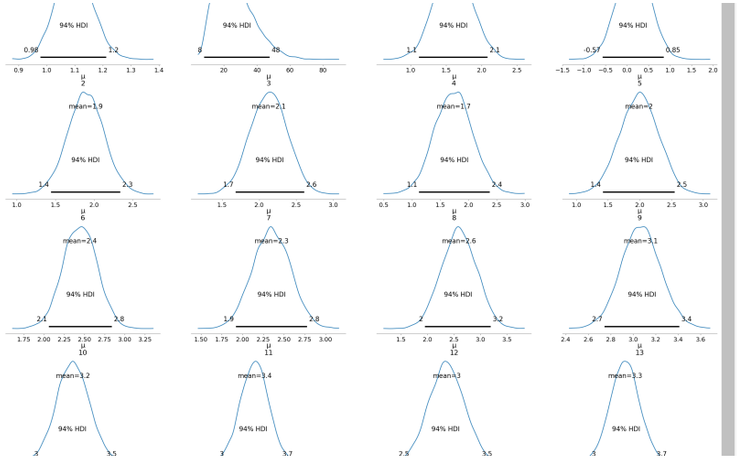
In the above plots, it's apparent that the Bayesian P-values that this robust model created distributions that run clearly to expected distribution of Bayes p-values. The posterior predictive plot demonstrates that the sampling worked well, since the observed and predicted values are similar.

(e) Plot the posterior distributions for the parameters and discuss your results. Plot the response variable vs. one of the predictors using counterfactuals and discuss this plot.

```
az.plot_posterior(trace_robust)
```

```
/usr/local/lib/python3.10/dist-packages/arviz/plots/plot_utils.py:271: UserWarning: rcParams['plot.max_subplots'] (40) is smaller than the number of vari
  warnings.warn(
array([[<Axes: title={'center': 'beta0'}>,
        <Axes: title={'center': 'beta1'}>,
        <Axes: title={'center': 'beta2'}>,
        <Axes: title={'center': 'beta3'}>],
       [<Axes: title={'center': 'beta4'}>,
        <Axes: title={'center': 'beta5'}>,
        <Axes: title={'center': 'beta6'}>,
        <Axes: title={'center': 'beta7'}>],
       [<Axes: title={'center': 'beta8'}>,
        <Axes: title={'center': 'beta9'}>,
        <Axes: title={'center': 'beta10'}>,
        <Axes: title={'center': 'beta11'}>],
       [<Axes: title={'center': 'σ'}>, <Axes: title={'center': 'ν'}>,
        <Axes: title={'center': 'μ\n0'}>,
        <Axes: title={'center': 'μ\n1'}>],
       [<Axes: title={'center': 'μ\n2'}>,
        <Axes: title={'center': 'μ\n3'}>,
        <Axes: title={'center': 'μ\n4'}>,
        <Axes: title={'center': 'μ\n5'}>],
       [<Axes: title={'center': 'μ\n6'}>,
        <Axes: title={'center': 'μ\n7'}>,
        <Axes: title={'center': 'μ\n8'}>,
        <Axes: title={'center': 'μ\n9'}>],
       [<Axes: title={'center': 'μ\n10'}>,
        <Axes: title={'center': 'μ\n11'}>,
        <Axes: title={'center': 'μ\n12'}>,
        <Axes: title={'center': 'μ\n13'}>],
       [<Axes: title={'center': 'μ\n14'}>,
        <Axes: title={'center': 'μ\n15'}>,
        <Axes: title={'center': 'μ\n16'}>,
        <Axes: title={'center': 'μ\n17'}>],
       [<Axes: title={'center': 'μ\n18'}>,
        <Axes: title={'center': 'μ\n19'}>,
        <Axes: title={'center': 'μ\n20'}>,
        <Axes: title={'center': 'μ\n21'}>],
       [<Axes: title={'center': 'μ\n22'}>,
        <Axes: title={'center': 'μ\n23'}>,
        <Axes: title={'center': 'μ\n24'}>,
        <Axes: title={'center': 'μ\n25'}>]], dtype=object)
```

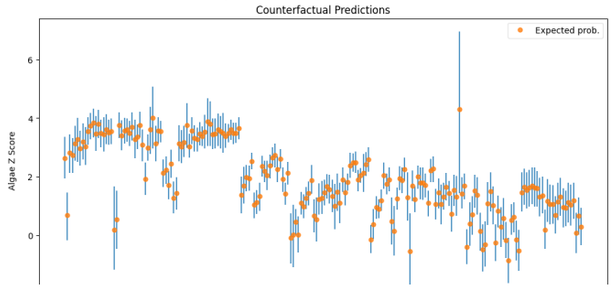




These posterior plots show the parameters follow a relativel normal distribution, and the distribution of the betas follows a similar trend, demonstrating satisfactory sampling. The only possible issue is a rightward skew of the mean distr. of oPo4

```
    trace_robust,
    var_names=["μ"],
    return_inferencedata=True,
    predictions=True,
    extend_inferencedata=True,
    random_seed=1234,
  )

_, ax = plt.subplots(figsize=(12, 6))

counterfactual = algae_pp_pred.predictions_constant_data['no3']
model_preds = algae_pp_pred.predictions

ax.vlines(
  counterfactual,
  *az.hdi(model_preds)["μ"].transpose("hdi", ...),
  alpha=0.8,
)

ax.plot(
  counterfactual,
  model_preds["μ"].mean(("chain", "draw")),
  "o",
  ms=5,
  color="C1",
  alpha=0.8,
  label="Expected prob.",
)
ax.set_xlabel("no3 Z Score")
ax.set_ylabel("Algae Z Score")
ax.set_title("Counterfactual Predictions")
ax.legend(fontsize=10, frameon=True, framealpha=0.5);
```

```
100.00% [8000/8000 00:01<00:00]
```



Though, this is not ideal, a general negative trend can be observed between the no3 Z-score and the algae z-score based on the counterfactual plot.

(f) The data set, algaetest.csv, contains a test set. Build a second model that may contain fewer predictors but should contain at least one nonlinear element. Evaluate both models using mean square error for the test set.

```
algae_test.columns = algae_train.columns

i = 0
to_drop = []
for i in range(len(algae_test['NO3'])):
  if (algae_test.iloc[i,6] == '1.0530011160.599611435.000001690.00000') or (algae_test.iloc[i,6] == '1.598006249.60010') or (algae_test.iloc[i,6] ==
    to_drop.append(i)
algae_test = algae_test.drop(to_drop)

predictors2 = list(algae_test.columns[0:11])

for col in predictors2[3:]:
  print(col)
  idx = predictors2.index(col)
  for i in range(len(algae_test[col])):
    algae_test.iloc[i, idx] = float(algae_test.iloc[i, idx])
  algae_test[col] = algae_test[col].replace(np.nan, np.mean(algae_test[col]))
  algae_test[col] -= np.mean(algae_test[col])
  algae_test[col] /= np.std(algae_test[col])

algae_test['a1'].loc[algae_test['a1']>0] = np.log(algae_test['a1'].loc[algae_test['a1']>0])
```

```
mxPH
mnO2
Cl
NO3
NH4
oPO4
PO4
Chla
<ipython-input-18-46ed7063d092>:25: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  algae_test['a1'].loc[algae_test['a1']>0] = np.log(algae_test['a1'].loc[algae_test['a1']>0])
```

```
algae_test
```

```
to_replace = {'high__': 'high', 'small_': 'small', 'large_': 'large', 'low___':
              'low', 'XXXXXXX': 0.00000}

algae_test = algae_test.replace(to_replace)
```

	season	size	speed	mxPH	mnO2	Cl	NO3	NH4	oPO4	PO4	Chla	a1	a2	a3	a4	a5	a6	a7
0	winter	small	medium	0.069543	-0.250135	0.500401	1.847195	0.575631	-0.314827	0.30931	-0.235486	0.182322	0.0	0.0	0.0	23.2	46.4	0.0
1	summer	small	medium	0.091987	-1.002142	0.996222	-0.389696	-0.000947	-0.101083	0.059933	-0.428343	1.945910	23.0	6.5	1.4	21.2	0.0	2.1
2	spring	small	high	0.484759	-0.438137	0.708331	0.050274	1.454666	-0.140185	0.040279	-0.524772	0.336472	38.2	2.4	0.0	4.8	1.0	1.2
3	spring	small	medium	0.204208	1.817884	-0.467225	-1.195141	-0.228159	-0.717077	-0.696988	-0.613783	1.360977	55.4	8.4	0.0	0.0	0.0	0.0
4	summer	small	medium	0.507204	1.300879	-0.614769	-0.849421	-0.818174	-0.731794	-0.999832	0.261493	3.346389	2.4	0.0	0.0	0.0	0.0	4.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
134	summer	medium	high	0.226652	0.407871	-0.74041	-0.907453	-0.722951	-0.392811	-0.597816	-0.598948	2.895912	1.7	2.0	0.0	1.7	5.9	0.0
135	winter	large	low	0.574536	0.689874	-0.067968	1.366891	-0.207637	0.717794	-1.107721	2.603597	0.095310	3.9	2.1	0.0	3.9	4.6	2.3
136	winter	large	low	0.877532	1.112878	-0.408073	0.354425	-0.425942	-0.246145	-0.255449	0.498856	0.000000	4.7	0.0	0.0	2.6	2.6	0.0
137	summer	large	low	0.204208	-0.532137	-0.286333	-0.040683	-0.675801	-0.091620	-0.202824	0.274844	0.000000	12.0	1.7	0.0	2.7	0.0	0.0
138	autumn	large	low	0.484759	0.830875	-0.381817	0.097193	-0.504183	-0.201995	-0.094596	0.53268	0.530628	7.0	1.2	0.0	4.8	3.1	0.0

132 rows × 18 columns

```
algae_test['mxPH'] = np.square(algae_test['mxPH'])
with pm.Model() as robust_test:
  beta0 = pm.Normal('beta0', mu=0, sigma=10)
  beta1 = pm.Normal('beta1', 0, 10)
  beta2 = pm.Normal('beta2', mu=0, sigma=10)
  beta3 = pm.Normal('beta3', mu=0, sigma=10)
  beta4 = pm.Normal('beta4', mu=0, sigma=10)
  beta5 = pm.Normal('beta5', mu=0, sigma=10)
  beta6 = pm.Normal('beta6', 0, 10)


  #σ = pm.HalfNormal("σ", 50)
  σ = pm.Uniform("σ", 10**-5, 10)
  ν = pm.HalfNormal("ν", 20)

  ph = pm.MutableData("ph", algae_test.iloc[:,3])
```

```
  mn = pm.MutableData("mn", algae_test.iloc[:,4])
  cl = pm.MutableData("cl", algae_test.iloc[:,5])
  nh4 = pm.MutableData("nh4", algae_test.iloc[:,7])
  po4 = pm.MutableData("po4", algae_test.iloc[:,9])
  Chla = pm.MutableData("Chla", algae_test.iloc[:,10])

  μ = pm.Deterministic("μ", beta0 + beta1*ph + beta2*mn
                       + beta3*cl + beta4*nh4 + beta5*po4 + beta6*Chla)

  alg_test = pm.StudentT('a1', mu=μ, sigma=σ, nu=ν,observed=algae_test['a1'])
  trace_test = pm.sample(cores = 8, return_inferencedata = False, random_seed=1234)
  trace_robust_test = pm.to_inference_data(trace=trace_test, log_likelihood=True)
  robust_pp_test = pm.sample_posterior_predictive(trace_robust_test, extend_inferencedata=True,
                                                  random_seed=1234)
  test_y_pred = robust_pp_test['posterior_predictive']['a1']
```

```
100.00% [16000/16000 01:14<00:00 Sampling 8 chains, 0 divergences]
100.00% [8000/8000 00:02<00:00]
```

```
#Evaluating MSE
from sklearn.metrics import mean_squared_error

pred = y_pred_test.mean(dim =['chain','draw'])
mse = mean_squared_error(pred, algae_train['a1'])
mse
```

```
1.2065609866571974
```

```
pred_test = test_y_pred.mean(dim =['chain','draw'])
mse_test = mean_squared_error(pred_test, algae_test['a1'])
mse_test
```

```
1.2711914607782464
```

(g) Summarize your results and give your answer to the EU's goal.

Bsed on everything that I've seen in these models, their p-values, predictive performance MSE, it seems that that these would do a decent job in predicting the likelyhood of algae growth, given each of the predictor variables and outcomes. In order to better deal with these algae blooms, the linear model appears to do a better job, and I would reccommend the EU use such a model when motioring water content.

```
import pymc as pm
import matplotlib.pyplot as plt
import arviz as az
import pandas as pd
from scipy import special, stats
import numpy as np
import seaborn as sns



from google.colab import files
uploaded = files.upload()
```

Choose Files CHDdata.csv

- **CHDdata.csv**(text/csv) - 21018 bytes, last modified: 10/24/2023 - 100% done

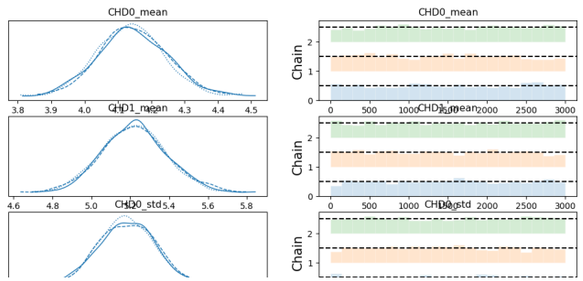
Saving CHDdata.csv to CHDdata.csv

```
chd = pd.read_csv('CHDdata.csv', header=0)
```

1. (10) Based on the analysis we did in class comparing systolic blood pres sure among men in the South African study [1], you are now asked to extend this analysis by comparing low density lipoprotein (ldl) levels be tween the same two groups in the study: those with and without coronary heart disease. Perform this analysis and comment on whether you nd a statistically signi cant di erence between the groups for this variable. To do this analysis you will use the CHD data set (CHDdata.csv).

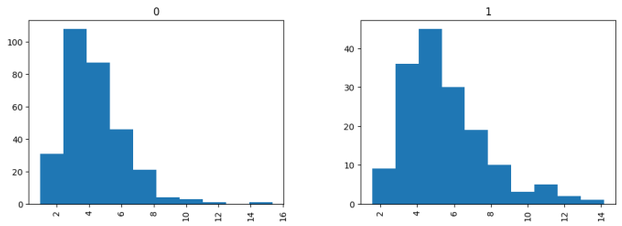
```
summary_stats = (chd.loc[:, ["chd", "ldl"]]
                    .groupby("chd")
                    .agg(["mean", "std", "count"]))
summary_stats

ldl_df = chd[["ldl", "chd"]]
ldl_df.hist("ldl", by="chd", figsize=(12, 4))
```

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
CHD0_mean	4.143	0.103	3.956	4.341	0.002	0.002	2362.0	2091.0	1.0
CHD1_mean	5.220	0.166	4.916	5.542	0.003	0.002	2701.0	2168.0	1.0
CHD0_std	1.492	0.092	1.328	1.675	0.002	0.001	2345.0	1868.0	1.0
CHD1_std	1.806	0.144	1.543	2.081	0.003	0.002	2408.0	2179.0	1.0
v	6.034	1.720	3.529	9.056	0.041	0.032	2060.0	2034.0	1.0
difference of means	-1.077	0.189	-1.443	-0.726	0.004	0.003	2846.0	2413.0	1.0
difference of stds	-0.314	0.146	-0.584	-0.043	0.002	0.002	3766.0	2542.0	1.0
effect size	-0.651	0.119	-0.874	-0.427	0.002	0.002	2736.0	2171.0	1.0




```
az.plot_forest(chd_trace, var_names=["difference of means", "difference of stds", "effect size"])
```

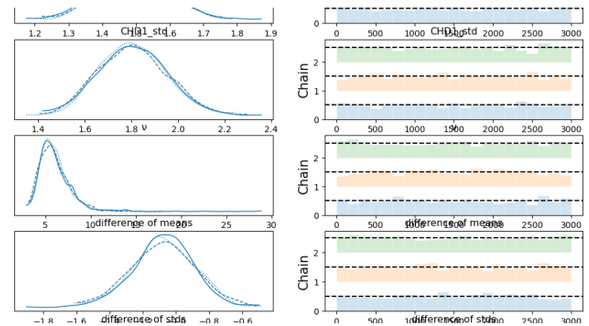
```
y0 = ldl_df.loc[ldl_df["chd"]==0]['ldl']
y1 = ldl_df.loc[ldl_df["chd"]==1]["ldl"]
```



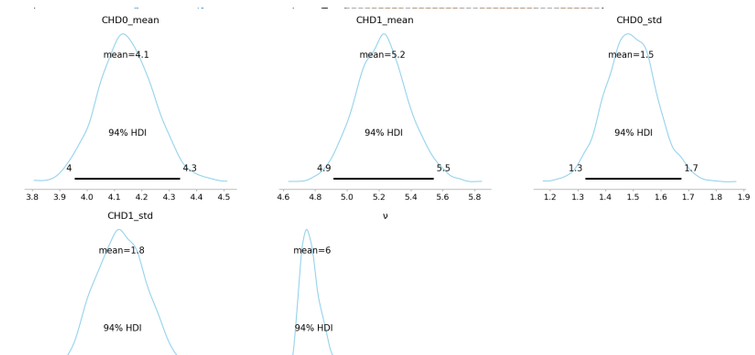

```
random_seed = 100
cores = 3

μ_prior = chd['ldl'].mean()
σ_prior = chd['ldl'].std() * 2

# prior for Std
σ_low = 1
σ_high = 100

with pm.Model() as model:
    CHD0_mean = pm.Normal("CHD0_mean", mu=μ_prior, sigma=σ_prior)
    CHD1_mean = pm.Normal("CHD1_mean", mu=μ_prior, sigma=σ_prior)
    CHD0_std = pm.Uniform("CHD0_std", lower=σ_low, upper=σ_high)
    CHD1_std = pm.Uniform("CHD1_std", lower=σ_low, upper=σ_high)
    v = pm.Exponential("v", 1/29)
    CHD0 = pm.StudentT("No_CHD", nu=v, mu=CHD0_mean, sigma=CHD0_std, observed=y0)
    CHD1 = pm.StudentT("CHD", nu=v, mu=CHD1_mean, sigma=CHD1_std, observed=y1)
    diff_of_means = pm.Deterministic("difference of means", CHD0_mean - CHD1_mean)
    diff_of_stds = pm.Deterministic("difference of stds", CHD0_std - CHD1_std)
    effect_size = pm.Deterministic(
        "effect size", diff_of_means / np.sqrt((CHD0_std ** 2 + CHD1_std ** 2) / 2)
    )

    chd_trace=pm.sample(random_seed = random_seed, cores = cores)
```

100.00% [6000/6000 00:12<00:00 Sampling 3 chains, 0 divergences]

```
az.plot_trace(chd_trace, kind = "rank_bars")
az.summary(chd_trace)
```

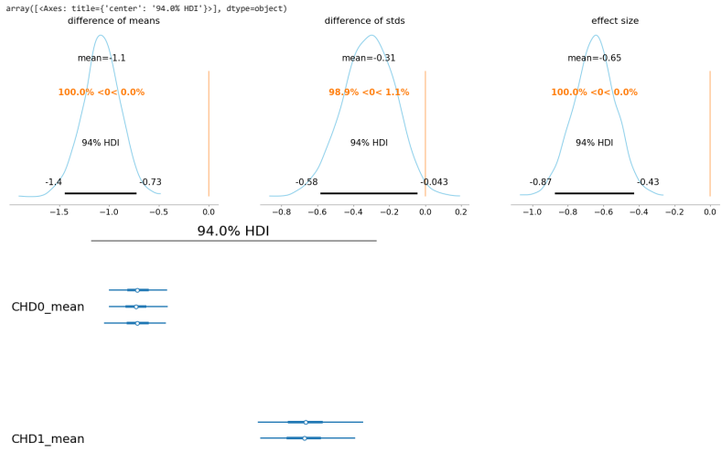



```
az.plot_posterior(
    chd_trace,
    var_names=["CHD0_mean", "CHD1_mean", "CHD0_std", "CHD1_std", "v"],
    color="#87ceeb",
);
```




```
az.plot_posterior(
    chd_trace,
    var_names=["difference of means", "difference of stds", "effect size"],
    ref_val=0,
    color="#87ceeb",
);
az.plot_forest(chd_trace, var_names = ['CHD0_mean','CHD1_mean'])
```

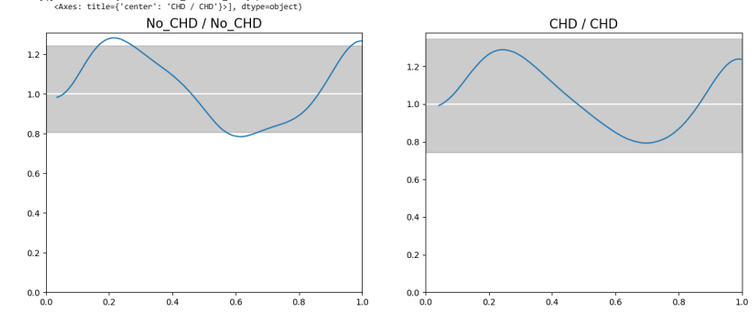
```
array([<Axes: title={'center': '94.0% HDI'}>], dtype=object)
```




```
#chd_trace
chd_pp = pm.sample_posterior_predictive(chd_trace, model = model)
az.plot_bpv(chd_pp)

# !!!!!!! Note: I ran very similar code as the prof and these plots were generated once, but it seems
# that pymc was updated or something and now I can't create the plots when I reran this and I havent beeen able to resolve the issue
```

100.00% [3000/3000 00:01<00:00]

```
array([<Axes: title={'center': 'No_CHD / No_CHD'}>,
       <Axes: title={'center': 'CHD / CHD'}>], dtype=object)
```

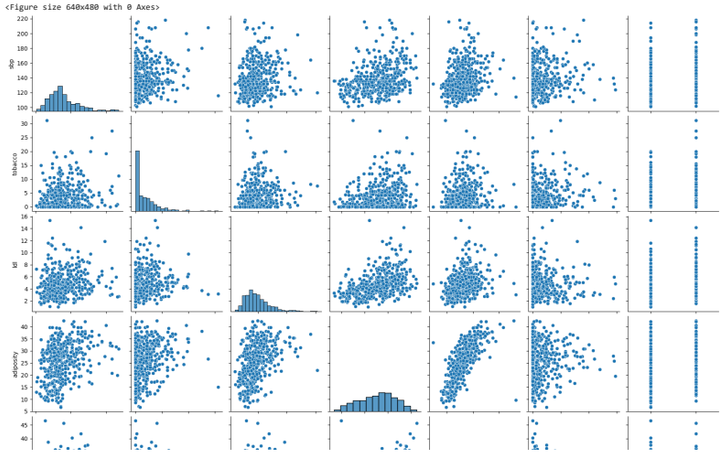


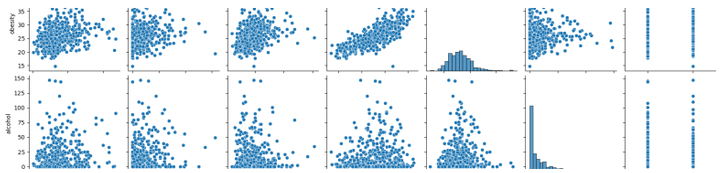
Based on the various plotes above, there is a clear difference in likelyhood to develop CHD between groups with lower and higher LDL contnet. In particular, the 94% HDI of the posterior difference of means shows that group with higher mean LDL content is more likely to have CHD. The same can be seen with the secondary 94%HDI set of graphs, and the subsequenmt difference of means graph. The effect size, in addition to the difference of means, seems to show the same general trends as well.

3. (30) The South African Health Ministry is now asking you to to extend your work from problem 1 and perform a full analysis of the data in the coronary heart disease study [1]. Their goal is to understand factors that may be associate with coronary heart disease to inform further more focused research on those factors. For this work you will again use the CHD data set (CHDdata.csv). Perform the following steps and answer the associated questions.

(a) Perform a simple exploratory analysis of the data by at least viewing summary statistics. Comment on what the EDA implies for your analysis.

```
#EDA: checking how each continious var is compared against each other continiuous var and also the response variable CHD
# even though chd is discrete I think its a good idea to see how each variable affects it
sns.pairplot(chd,vars= ['sbp' ,'tobacco' ,'ldl' , 'adiposity' ,'obesity' ,'alcohol' ,'chd'])
#Performing analysis of categorical variables againt the response
sns.boxplot(data=chd, x="famhist", y="chd")
plt.figure()
```





Based on the figures above, its clear that theres some correlations between high sbp levels, tobacco use, ldl, obesity and alcohol use. But at the same time there doesnt appear to be any colinearity so high that any predictor needs to be dropped.

(b) Preprocess the data by scaling the predictor variables using a z-transformation (subtract the mean divide by the standard deviation). Explain why this is appropriate. Code the categorical variables and explain your coding

```
preds_num = ['sbp', 'tobacco', 'ldl', 'adiposity', 'typea', 'obesity', 'alcohol', 'age']
for p in preds_num:
  chd[p] -= np.mean(chd[p])
  chd[p] /= np.std(chd[p])
#chd = pd.get_dummies(chd,columns=['famhist'])
fam_vals = {'Absent': 0, 'Present': 1}
chd['famhist'] = chd['famhist'].replace(fam_vals)


chd
```

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
0	1.058564	1.823073	0.478412	-0.295503	1	-0.418470	-0.176786	3.277738	0.629336	1
1	0.277089	-0.790237	-0.159680	0.412140	0	0.193344	0.671373	-0.612745	1.383115	1
2	-0.992806	-0.774980	-0.609245	0.884332	1	-0.112563	0.735519	-0.541183	0.218184	0
3	1.546985	0.842264	0.807126	1.624141	1	-0.214532	1.412621	0.295062	1.040488	1
4	-0.211332	2.171805	-0.599577	0.305351	1	0.703189	-0.012856	1.647775	0.423760	1
...	...	...	...	...	...	...	...	...	...	...
457	3.696039	-0.705234	0.599263	0.812281	0	1.111065	0.571590	-0.696983	1.040488	0
458	2.133091	0.123004	-0.159680	0.861173	0	-0.112563	0.609602	0.068519	0.629336	1
459	-1.481228	-0.138545	-1.522877	-1.309364	0	-1.336191	-1.414575	0.392385	0.834912	0

Here, scaling the data makes the distribution appear normal and thereby we can avoid the need to use the individual means for each column of the data. Encoding the categorical variables allows for numerical analysis to be run on those variables.

462 rows × 10 columns

(c) Develop and use Bayesian GLM model with all the predictor variables. Explain your choice of priors and their parameters. Show the graphical representation of your model.

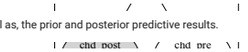
```
with pm.Model() as chd_lin_model:
  beta0 = pm.Normal('beta0', mu=0, sigma=10)
  beta1 = pm.Normal('beta1', mu=0, sigma=10)
  beta2 = pm.Normal('beta2', mu=0, sigma=10)
  beta3 = pm.Normal('beta3', mu=0, sigma=10)
  beta4 = pm.Normal('beta4', mu=0, sigma=10)
  beta5 = pm.Normal('beta5', mu=0, sigma=10)
  beta6 = pm.Normal('beta6', mu=0, sigma=10)
  beta7 = pm.Normal('beta7', mu=0, sigma=10)
  beta8 = pm.Normal('beta8', mu=0, sigma=10)
  beta9 = pm.Normal('beta9', mu=0, sigma=10)
```

```
μ = pm.Deterministic("μ", beta0 + beta1*chd.iloc[:, 0] + beta2*chd.iloc[:, 1] + beta3*chd.iloc[:, 2] + beta4*chd.iloc[:, 3]
                         + beta5*chd.iloc[:, 4] + beta6*chd.iloc[:, 5] + beta7*chd.iloc[:, 6] + beta8*chd.iloc[:, 7] + beta9*chd.iloc[:, 8])

  chd_data = chd.iloc[:, 9]

  chd_pre = pm.Bernoulli('chd_pre', p = pm.math.sigmoid(μ))

  chd_post = pm.Bernoulli('chd_post', p = pm.math.sigmoid(μ), observed=chd_data)

  trace_glm = pm.sample(1000, cores=8, random_seed=1234, return_inferencedata=False)
  glm_trace = pm.to_inference_data(trace=trace_glm, log_likelihood=True)
  glm_pp = pm.sample_posterior_predictive(glm_trace,
                                          var_names=["chd_pre","chd_post"],
                                          random_seed = 1234)

g3=pm.model_to_graphviz(chd_lin_model)
graphics_path = '/content/drive/MyDrive/'
g3.render(graphics_path+"chd_lin_model", format='png', cleanup=True)
g3
```

100.00% [16000/16000 18:17<00:00 Sampling 8 chains, 0 divergences]
100.00% [8000/8000 00:03<00:00]

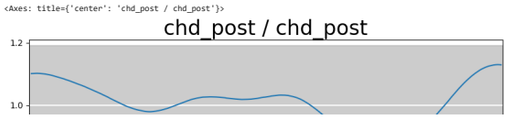
In this linear model, I model the correlation of 9 variables with the outcome chd_data. Each variable is associated with a coefficient (beta) that describes its relationship with the outcome. A deterministic function computes the expected value (μ) based on a linear combination of these variables and their respective coefficients. The outcome is binary, and hence a Bernoulli distribution is used for both the prior prediction (chd_pre) and the posterior likelihood (chd_post), with a logistic sigmoid function linking the linear combination to a probability. Sampling provides posterior estimates for all model parameters.

(d) Evaluate both the sampling used by your model, as well as, the prior and posterior predictive results.




```
with chd_lin_model:
  chd_prior = pm.sample_prior_predictive()
az.plot_dist(chd_prior.prior['chd_pre'])
```

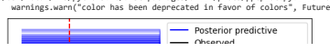

```
az.plot_bpv(glm_pp, figsize = (10,8))
```




```
with chd_lin_model:
  pm.sample_posterior_predictive(glm_trace, extend_inferencedata=True, random_seed=1234)
fig, ax = plt.subplots()
az.plot_ppc(glm_pp, color = 'b', observed_rug=True, ax=ax)
ax.axvline(chd['chd'].mean(), ls="--", color="r", label="Observed mean")
ax.legend(fontsize=10);
```

100.00% [8000/8000 00:02<00:00]

```
/usr/local/lib/python3.10/dist-packages/arviz/plots/ppcplot.py:241: FutureWarning: color has been deprecated in favor of colors
  warnings.warn("color has been deprecated in favor of colors", FutureWarning)
```

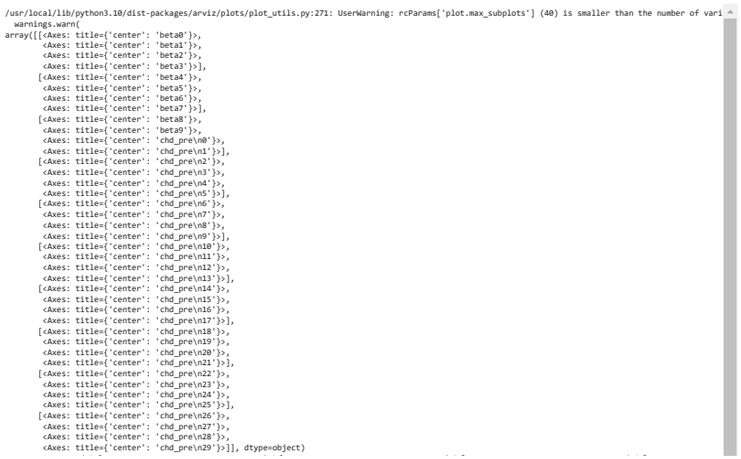


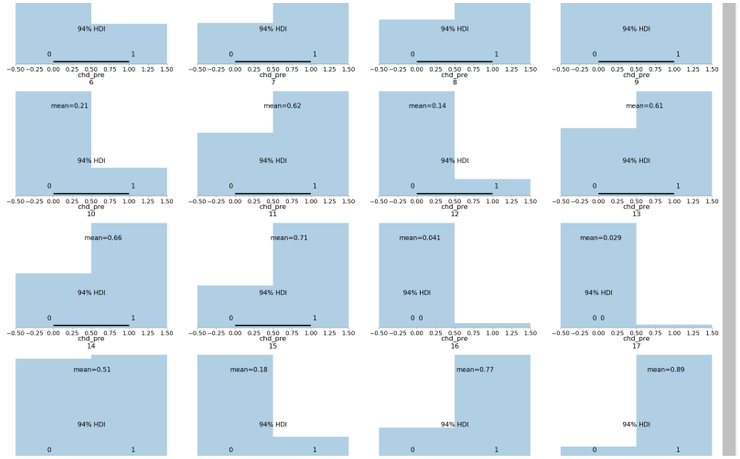
Basedon the above plots, we can see that the model has p-values that lie in the accepted range, demonstrating that the model is satisfactory. The posterior predictive sampling, with similaries between the observed and predicted parameters, supports the validity of the model.

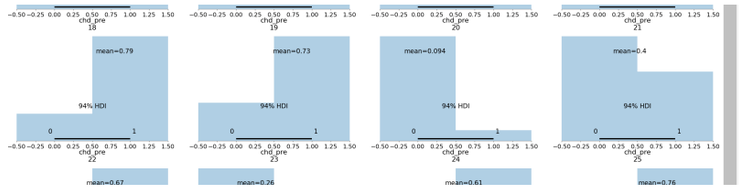
(e) Provide plots of the posterior distributions for the parameters and discuss what these results imply for the relevant predictor variables and the overall model. Show posterior analyses to include relevant odds ratio(s) and describe what this analysis means for the goal of your study.

```
az.plot_posterior(glm_trace)
```

```
/usr/local/lib/python3.10/dist-packages/arviz/plots/plot_utils.py:271: UserWarning: rcParams['plot.max_subplots'] (40) is smaller than the number of vari
  warnings.warn(
array([[<Axes: title={'center': 'beta0'}>,
        <Axes: title={'center': 'beta1'}>,
        <Axes: title={'center': 'beta2'}>,
        <Axes: title={'center': 'beta3'}>],
       [<Axes: title={'center': 'beta4'}>,
        <Axes: title={'center': 'beta5'}>,
        <Axes: title={'center': 'beta6'}>,
        <Axes: title={'center': 'beta7'}>],
       [<Axes: title={'center': 'beta8'}>,
        <Axes: title={'center': 'beta9'}>,
        <Axes: title={'center': 'chd_pre\n0'}>,
        <Axes: title={'center': 'chd_pre\n1'}>],
       [<Axes: title={'center': 'chd_pre\n2'}>,
        <Axes: title={'center': 'chd_pre\n3'}>,
        <Axes: title={'center': 'chd_pre\n4'}>,
        <Axes: title={'center': 'chd_pre\n5'}>],
       [<Axes: title={'center': 'chd_pre\n6'}>,
        <Axes: title={'center': 'chd_pre\n7'}>,
        <Axes: title={'center': 'chd_pre\n8'}>,
        <Axes: title={'center': 'chd_pre\n9'}>],
       [<Axes: title={'center': 'chd_pre\n10'}>,
        <Axes: title={'center': 'chd_pre\n11'}>,
        <Axes: title={'center': 'chd_pre\n12'}>,
        <Axes: title={'center': 'chd_pre\n13'}>],
       [<Axes: title={'center': 'chd_pre\n14'}>,
        <Axes: title={'center': 'chd_pre\n15'}>,
        <Axes: title={'center': 'chd_pre\n16'}>,
        <Axes: title={'center': 'chd_pre\n17'}>],
       [<Axes: title={'center': 'chd_pre\n18'}>,
        <Axes: title={'center': 'chd_pre\n19'}>,
        <Axes: title={'center': 'chd_pre\n20'}>,
        <Axes: title={'center': 'chd_pre\n21'}>],
       [<Axes: title={'center': 'chd_pre\n22'}>,
        <Axes: title={'center': 'chd_pre\n23'}>,
        <Axes: title={'center': 'chd_pre\n24'}>,
        <Axes: title={'center': 'chd_pre\n25'}>],
       [<Axes: title={'center': 'chd_pre\n26'}>,
        <Axes: title={'center': 'chd_pre\n27'}>,
        <Axes: title={'center': 'chd_pre\n28'}>,
        <Axes: title={'center': 'chd_pre\n29'}>]], dtype=object)
```







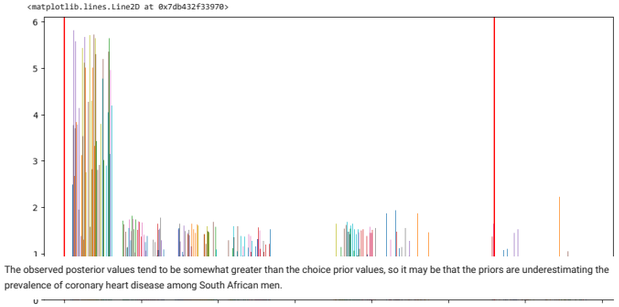
Based on the above plots, its clear that the explanatroy variables show a normal distribtution, so Bernouli distribution for the prior was a good choice. The HDI charts produced seem to indicate that the odds ratios for the parameters are acceptable. Beta are also normally distributed, showing that there shouldnt have been in issue in that area either.

In terms of the goals of the study, this helps us to see that the model being produced is accurate and should help our stakeholders achieve their goals.

```
_, ax = plt.subplots(figsize=(12, 6))
b = glm_trace.posterior["chd_pre"]
OR = np.mean(b, axis =0)
lb, ub = np.percentile(OR, 2.5), np.percentile(OR, 97.5)
plt.hist(np.exp(OR), bins=10, density=True)
plt.axvline(np.exp(lb), color = 'r')
plt.axvline(np.exp(ub), color = 'r')
```



```
<matplotlib.lines.Line2D at 0x7db432f33970>
```



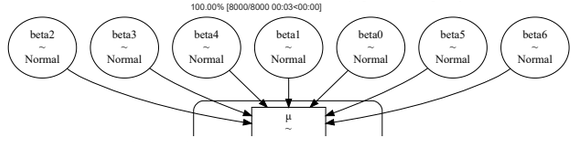
The observed posterior values tend to be somewhat greater than the choice prior values, so it may be that the priors are underestimating the prevalence of coronary heart disease among South African men.

(f) Develop a second model to help in your study of this problem. This new model can use fewer predictor variables but must contain at least one nonlinear component. Explain your choice of priors and their parameters, and show the graphical representation of your new model.

```
chd2 = chd
chd2['age'] = np.square(chd2['age'])

with pm.Model() as glm2:
  beta0 = pm.Normal('beta0', mu=0, sigma=10)
```

```
  beta1 = pm.Normal('beta1', mu=0, sigma=10)
  beta2 = pm.Normal('beta2', mu=0, sigma=10)
  beta3 = pm.Normal('beta3', mu=0, sigma=10)
  beta4 = pm.Normal('beta4', mu=0, sigma=10)
  beta5 = pm.Normal('beta5', mu=0, sigma=10)
  beta6 = pm.Normal('beta6', 0, 10)

  μ = pm.Deterministic("μ", beta0 + beta1*chd2.iloc[:, 0] + beta2*chd2.iloc[:, 1] + beta3*chd2.iloc[:, 2]
                       + beta4*chd2.iloc[:, 5] + beta5*chd2.iloc[:, 6] + beta6*chd2.iloc[:, 8])

  chd2_data = chd2.iloc[:, 9]

  chd2_pre = pm.Bernoulli('chd2_pre', p=pm.math.sigmoid(μ))

  chd2_post = pm.Bernoulli('chd2_post', p = pm.math.sigmoid(μ), observed=chd2_data)

  trace_glm2 = pm.sample(1000, cores=8, random_seed=1234, target_accept=.99, return_inferencedata=False)
  glm_trace2 = pm.to_inference_data(trace=trace_glm2, log_likelihood=True)
  glm_pp2 = pm.sample_posterior_predictive(glm_trace2,
                                           var_names=["chd2_pre","chd2_post"],
                                           random_seed = 1234)
g3f=pm.model_to_graphviz(glm2)
graphics_path = '/content/drive/MyDrive/'
g3f.render(graphics_path+"GLM_Model", format='png', cleanup=True)
g3f
```

```
100.00% [16000/16000 16:08<00:00 Sampling 8 chains, 0 divergences]
100.00% [8000/8000 00:03<00:00]
```

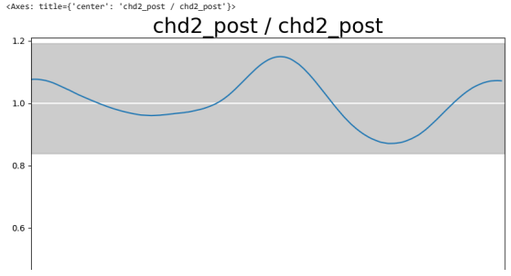


I kept the betas as normal, except for the use of a half normal on age, which was squared to become the nonlinear element. All other parameters remained the same for consistency.

(g) For the second model evaluate the sampling and choice of priors using Bayesian p-value and posterior predictive plots. Briefly explain your results.

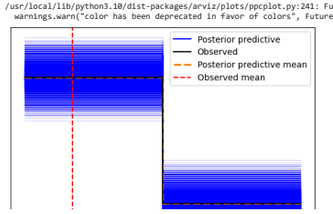
```
az.plot_bpv(glm_pp2, figsize = (10,8))
```

```
<Axes: title={'center': 'chd2_post / chd2_post'}>
```




```
with glm2:
  pm.sample_posterior_predictive(glm_trace2, extend_inferencedata=True, random_seed=1234)
fig, ax = plt.subplots()
az.plot_ppc(glm_pp2, color = 'b', observed_rug=True, ax=ax)
ax.axvline(chd2['chd'].mean(), ls="--", color="r", label="Observed mean")
ax.legend(fontsize=10);
```

```
100.00% [8000/8000 00:02<00:00]
/usr/local/lib/python3.10/dist-packages/arviz/plots/ppcplot.py:241: FutureWarning: color has been deprecated in favor of colors
  warnings.warn("color has been deprecated in favor of colors", FutureWarning)
```

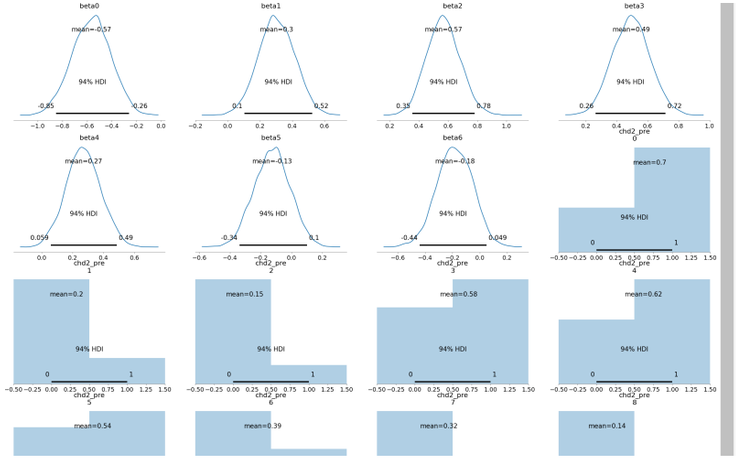


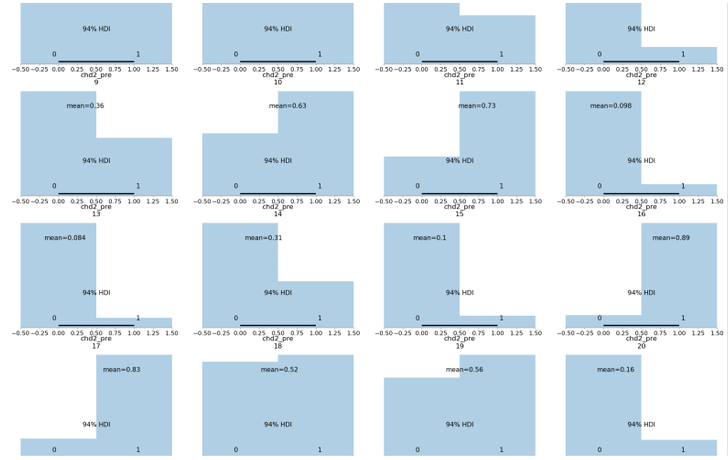
This model appears to be pretty decent. The produced Bayesian p-value is very close to 1, which appears satisfactory. Moreover, there are two clusters present within the posterior predictive plot, so it seems that the priors and parameters were well chosen.

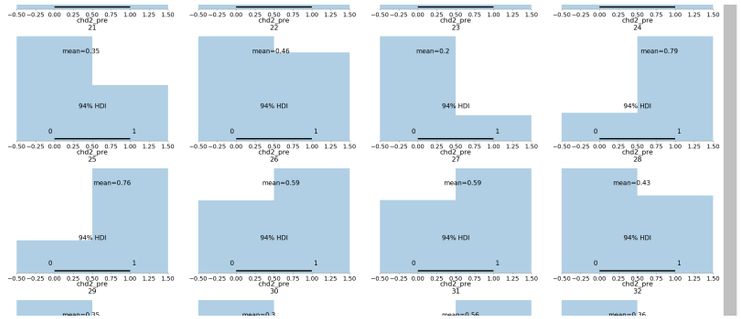
(h) For the second model provide plots of the posterior distributions for the parameters and discuss what these results imply for the relevant predictor variables and the overall model. Show posterior analyses to include relevant odds ratio(s) and describe what this analysis means for the goal of your study.

```
az.plot_posterior(glm_trace2)
```

```
/usr/local/lib/python3.10/dist-packages/arviz/plots/plot_utils.py:271: UserWarning: rcParams['plot.max_subplots'] (40) is smaller than the number of vari
  warnings.warn(
array([[<Axes: title={'center': 'beta0'}>,
        <Axes: title={'center': 'beta1'}>,
        <Axes: title={'center': 'beta2'}>,
        <Axes: title={'center': 'beta3'}>],
       [<Axes: title={'center': 'beta4'}>,
        <Axes: title={'center': 'beta5'}>,
        <Axes: title={'center': 'beta6'}>,
        <Axes: title={'center': 'chd2_pre\n0'}>],
       [<Axes: title={'center': 'chd2_pre\n1'}>,
        <Axes: title={'center': 'chd2_pre\n2'}>,
        <Axes: title={'center': 'chd2_pre\n3'}>,
        <Axes: title={'center': 'chd2_pre\n4'}>],
       [<Axes: title={'center': 'chd2_pre\n5'}>,
        <Axes: title={'center': 'chd2_pre\n6'}>,
        <Axes: title={'center': 'chd2_pre\n7'}>,
        <Axes: title={'center': 'chd2_pre\n8'}>],
       [<Axes: title={'center': 'chd2_pre\n9'}>,
        <Axes: title={'center': 'chd2_pre\n10'}>,
        <Axes: title={'center': 'chd2_pre\n11'}>,
        <Axes: title={'center': 'chd2_pre\n12'}>],
       [<Axes: title={'center': 'chd2_pre\n13'}>,
        <Axes: title={'center': 'chd2_pre\n14'}>,
        <Axes: title={'center': 'chd2_pre\n15'}>,
        <Axes: title={'center': 'chd2_pre\n16'}>],
       [<Axes: title={'center': 'chd2_pre\n17'}>,
        <Axes: title={'center': 'chd2_pre\n18'}>,
        <Axes: title={'center': 'chd2_pre\n19'}>,
        <Axes: title={'center': 'chd2_pre\n20'}>],
       [<Axes: title={'center': 'chd2_pre\n21'}>,
        <Axes: title={'center': 'chd2_pre\n22'}>,
        <Axes: title={'center': 'chd2_pre\n23'}>,
        <Axes: title={'center': 'chd2_pre\n24'}>],
       [<Axes: title={'center': 'chd2_pre\n25'}>,
        <Axes: title={'center': 'chd2_pre\n26'}>,
        <Axes: title={'center': 'chd2_pre\n27'}>,
        <Axes: title={'center': 'chd2_pre\n28'}>],
       [<Axes: title={'center': 'chd2_pre\n29'}>,
        <Axes: title={'center': 'chd2_pre\n30'}>,
        <Axes: title={'center': 'chd2_pre\n31'}>,
        <Axes: title={'center': 'chd2_pre\n32'}>]], dtype=object)
```

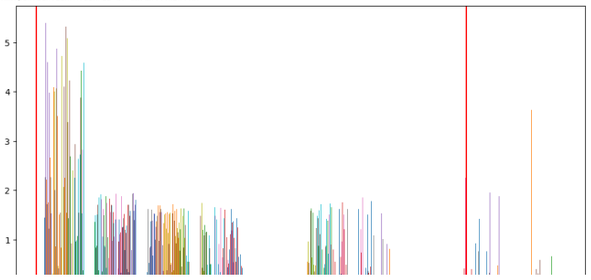







```
_, ax = plt.subplots(figsize=(12, 6))
b = glm_trace2.posterior["chd2_pre"]
OR = np.mean(b, axis =0)
lb, ub = np.percentile(OR, 2.5), np.percentile(OR, 97.5)
plt.hist(np.exp(OR), bins=10, density=True)
plt.axvline(np.exp(lb), color = 'r')
plt.axvline(np.exp(ub), color = 'r')
```

```
<matplotlib.lines.Line2D at 0x7db43482c340>
```



The ratio between the posterior and the prior are again consistently greater than one, demonstrarting that the prescence of CHD may very well be underestimated. In terms of the stakeholders, I would recommend using the other model as it is likely to be more accurate.

(i) Compare the two models using PSIS - leave-one-out cross validation and WAIC.

```
az.compare({'Linear': glm_trace, 'Nonlinear':glm_trace2})
```

	rank	elpd_loo	p_loo	elpd_diff	weight	se	dse	warning	scale

Based on this comparison between the two models, we can see that the linear model appears to havea better overall fit. This is supported by the elpd_diff score, which shows the liinear model to be better.

(j) Briefly summarize your analysis and provide an answer to the South African Health ministry to the question implied by their goal statement.

In conclusion, I have produced a couple models that predict the likelyhood of developing CHD based on a variety of factors. The most prominent risk factors of developing CHD based on these models seem to be: tobacco, LDL, age, family history, and systolic blood pressure. Dealing with these health factros through a public-health based approach may prove to lower the odds of developing chd overall.