```
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()
```

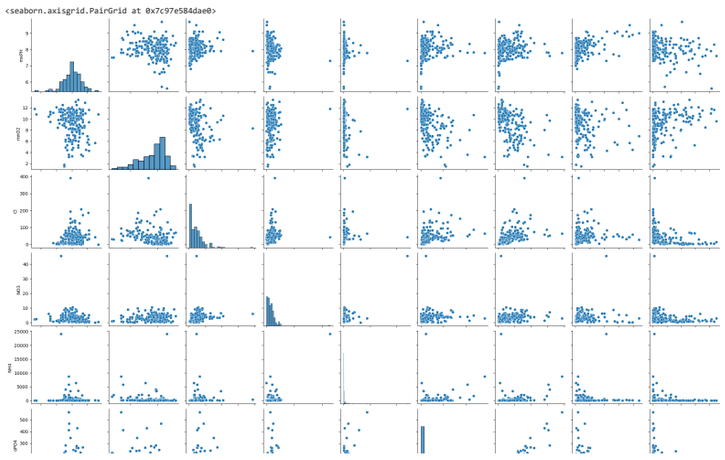
```
Choose Files  2 files
```
- **algae.csv**(text/csv) - 17987 bytes, last modified: 10/29/2023 - 100% done
- **algaeTest.csv**(text/csv) - 20683 bytes, last modified: 10/29/2023 - 100% done

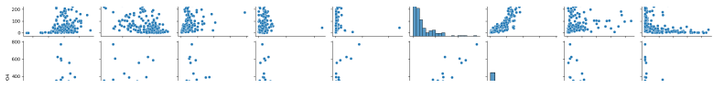
```
Saving algae.csv to algae.csv
Saving algaeTest.csv to algaeTest.csv
```

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.

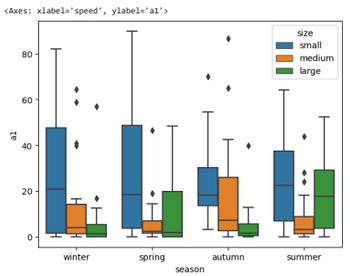
```
#algae_train = pd.read_csv('algae.csv', header=0)
algae_test = pd.read_csv('algaeTest.csv', header=0)


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)
```

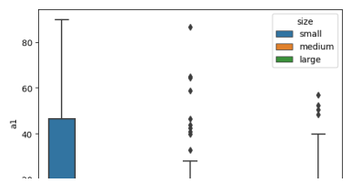
```
          a1
        mean       std count
mxPH
6.000  66.000       NaN     1
6.100  47.600  24.688999     4
6.200  31.800       NaN     1
6.250  42.716667  24.978904     6
6.300  41.633333  12.287121     3
...          ...       ...   ...
8.000   0.000000       NaN     1
8.050   7.200000       NaN     1
8.100  12.300000       NaN     1
8.150   1.200000       NaN     1
8.250   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' ])
```

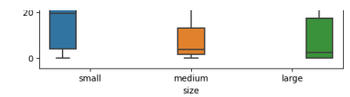




```
#Performing analysis of categorical variables againt the response
sns.boxplot(data=algae_train, x="season", y="a1", hue = 'size')
plt.figure()
sns.boxplot(data=algae_train, x="size", y="a1", hue = 'size')
plt.figure()
sns.boxplot(data=algae_train, x="speed", y="a1", hue = 'size')
```







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.

```
# 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])
```

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

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

```
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
```

```
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
```

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.

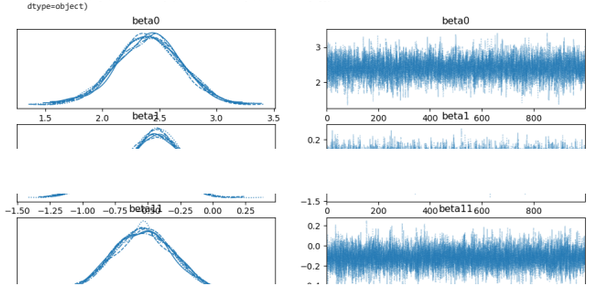
(c) Build a robust Bayesian regression model to predict the response and explain your choices for the priors.

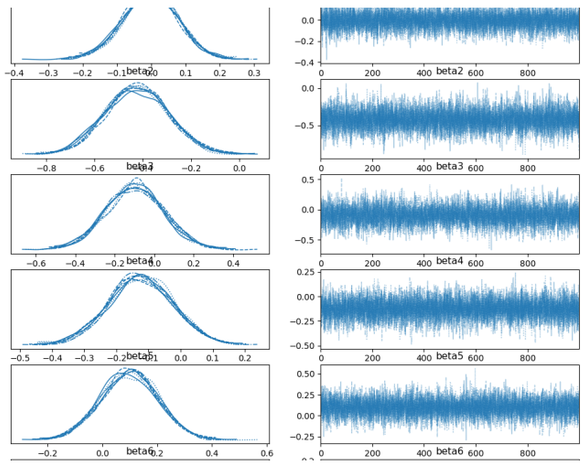
```
graphics_path = '/Users/addyg/Desktop/UVA_Courses/DS6040 Bayes/'
```

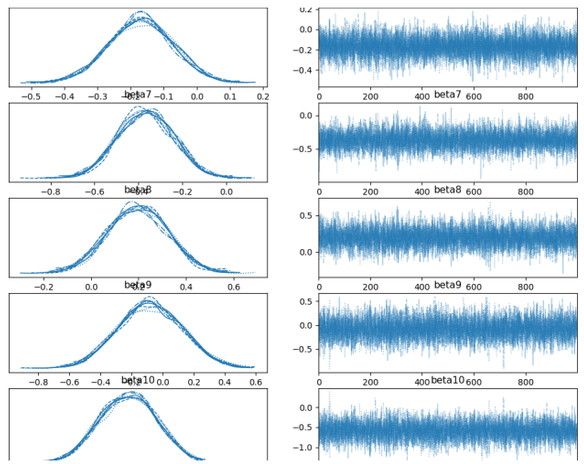
```
with pm.Model() as robust:
  beta0 = pm.Normal('beta0', mu=0, sigma=10)
  beta1 = pm.Normal('beta1', mu=0, sigma=10)
  beta2 = pm.Normal('beta2', mu=0, sigma=10)
  beta3 = pm.Normal('beta3', mu=0, sigma=10)
  beta4 = pm.Normal('beta4', mu=0, sigma=10)
  beta5 = pm.Normal('beta5', mu=0, sigma=10)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





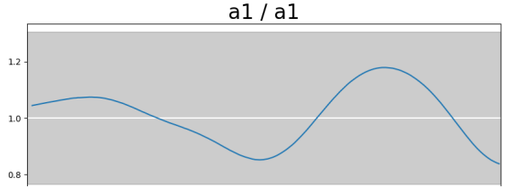


For this model, I chose to use normal priors for the beta distributions because the data has already been scaled in part b and a normal distribution is a reasonable expectation.

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

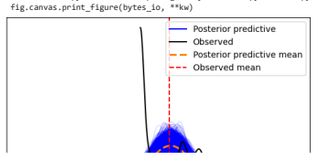
```
az.plot_bpv(trace_robust, figsize = (10,8))
```

```
<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)
```

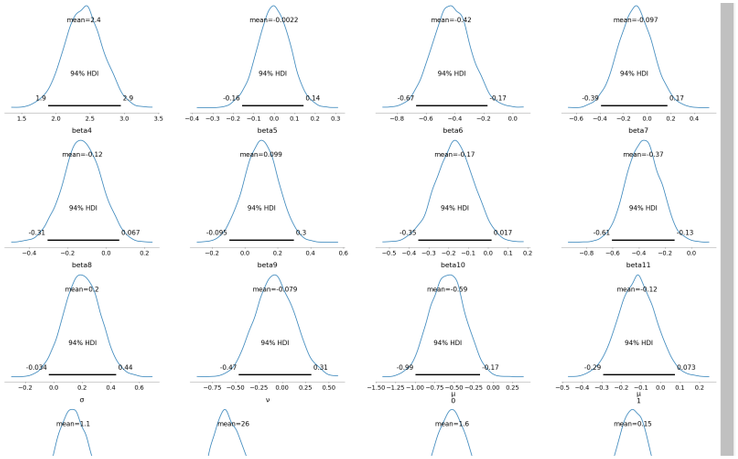


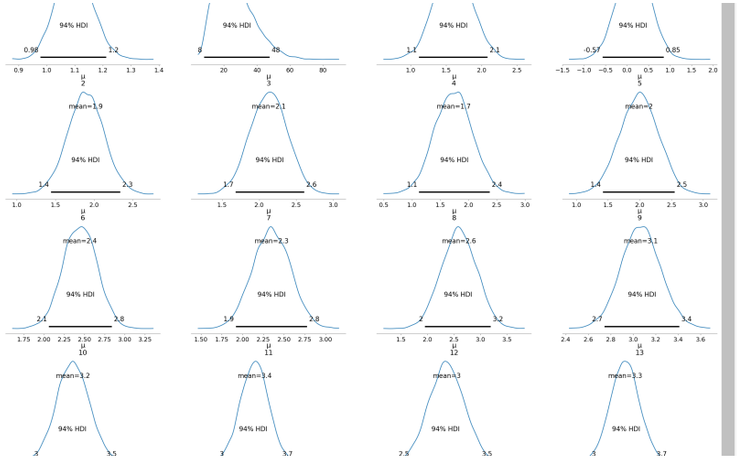
In the above plots, it's apparent that the Bayesian P-values that this robust model created distributions that run clearly to expected distribution of Bayes p-values. The posterior predictive plot demonstrates that the sampling worked well, since the observed and predicted values are similar.

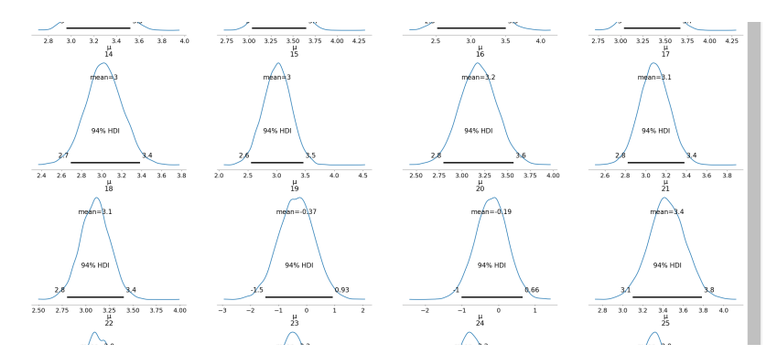
(e) Plot the posterior distributions for the parameters and discuss your results. Plot the response variable vs. one of the predictors using counterfactuals and discuss this plot.

```
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)
```





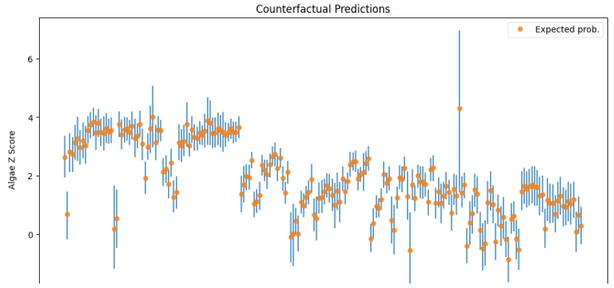


These posterior plots show the parameters follow a relativel normal distribution, and the distribution of the betas follows a similar trend, demonstrating satisfactory sampling. The only possible issue is a rightward skew of the mean distr. of oPo4

```
    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);
```

```
new_no3 = np.linspace(-2, 2, num=200)
#new_ph = np.ma.zeros(200,)
#here, setting all others constant
with robust:
  pm.set_data({'no3': new_no3})
  algae_pp_pred = pm.sample_posterior_predictive(
```

```
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.