DS 6040: Bayes Final Project Code

Stephanie Fissel, Jackie Fraley, Quinn Glovier

Loading Packages

In [1]:
```
import pandas as pd
import numpy as np
import pymc as pm
import arviz as az
import matplotlib.pyplot as plt
from cycler import cycler
import pytensor
import pytensor.tensor as pt
import scipy as sp
import scipy.stats as st
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_predict
from sklearn import linear_model
from sklearn.model_selection import cross_val_score

az.style.use('arviz-darkgrid')
plt.rcParams['figure.dpi'] = 300

random_seed = 444
cores = 4
```

Loading in the data

In [2]:
```
week = pd.read_csv("week_approach_maskedID_timeseries.csv")
week
```

Out[2]:

	nr. sessions	nr. rest days	total kms	max km one day	total km Z3-Z4-Z5-T1-T2	nr. tough sessions (effort in Z5, T1 or T2)	nr. days with interval session	total km Z3-4	max km Z3-4 one day	total km Z5-T1-T2	...	su
0	5.0	2.0	22.2	16.4	11.8	1.0	2.0	10.0	10.0	0.6	...	
1	5.0	2.0	21.6	16.4	11.7	1.0	2.0	10.0	10.0	0.5	...	
2	5.0	2.0	21.6	16.4	11.7	1.0	2.0	10.0	10.0	0.5	...	
3	5.0	2.0	21.6	16.4	11.7	1.0	2.0	10.0	10.0	0.5	...	
4	6.0	1.0	39.2	17.6	18.9	1.0	3.0	17.2	10.0	0.5	...	
...	...	...	...	...	...	...	...	...	...	...	...	
42793	4.0	3.0	59.5	19.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
42794	1.0	6.0	5.8	5.8	0.0	0.0	0.0	0.0	0.0	0.0	...	
42795	3.0	4.0	38.3	16.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
42796	5.0	2.0	67.0	15.1	0.0	0.0	0.0	0.0	0.0	0.0	...	
42797	4.0	3.0	45.0	12.2	0.0	0.0	0.0	0.0	0.0	0.0	...	

42798 rows × 72 columns

Exploratory Data Analysis (EDA)

Descriptive Statistics

In [3]:
```
week.describe()
```

Out[3]:

	nr. sessions	nr. rest days	total kms	max km one day	total km Z3-Z4-Z5-T1-T2	(e
count	42798.000000	42798.000000	42798.000000	42798.000000	42798.000000	427
mean	5.809337	1.874667	49.543911	14.009255	9.433621	
std	2.484234	1.853287	36.715017	9.071678	8.887120	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	5.000000	1.000000	22.800000	9.000000	1.000000	
50%	6.000000	1.000000	44.800000	13.400000	8.000000	
75%	7.000000	3.000000	70.100000	18.300000	14.600000	
max	14.000000	7.000000	242.000000	131.000000	100.000000	

8 rows × 72 columns

Data Information

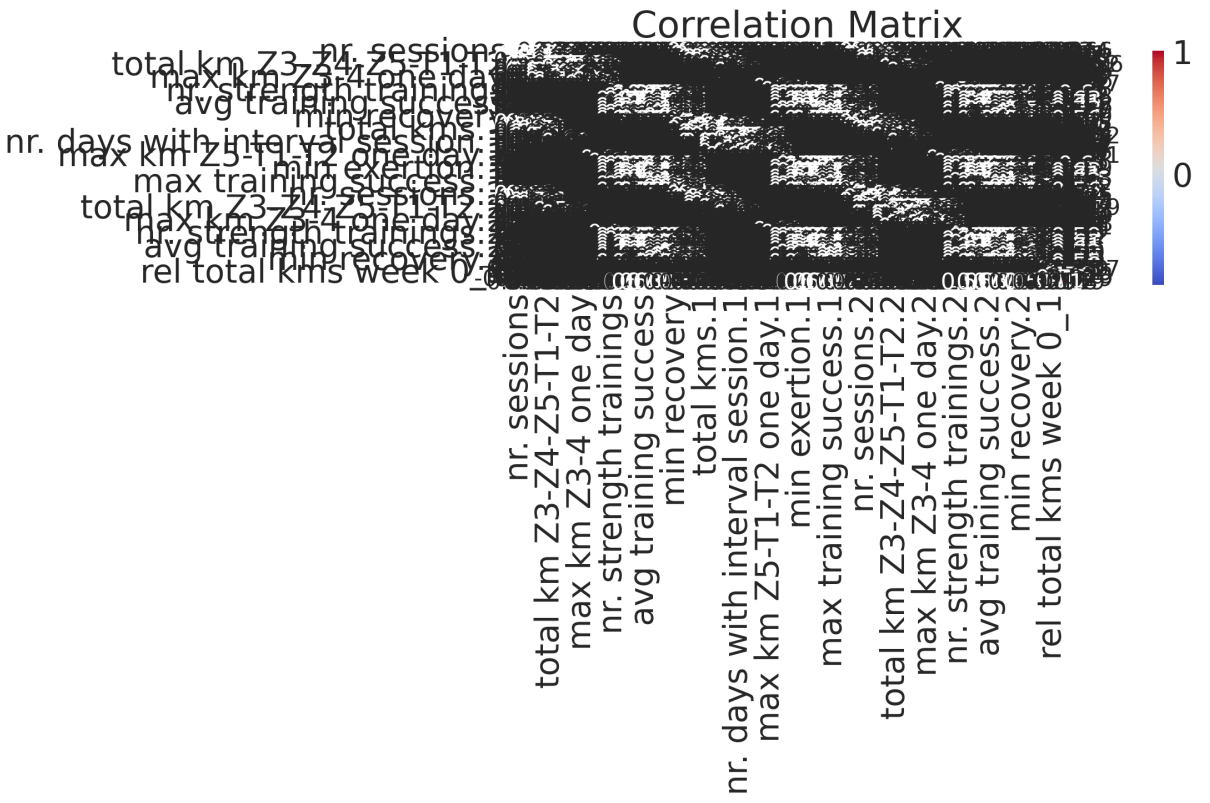
In [4]:
```
week.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42798 entries, 0 to 42797
Data columns (total 72 columns):
 #   Column                                        Non-Null Count  Dtype  
---  ------                                        --------------  -----  
 0   nr. sessions                                  42798 non-null  float64
 1   nr. rest days                                 42798 non-null  float64
 2   total kms                                     42798 non-null  float64
 3   max km one day                                42798 non-null  float64
 4   total km Z3-Z4-Z5-T1-T2                       42798 non-null  float64
 5   nr. tough sessions (effort in Z5, T1 or T2)   42798 non-null  float64
 6   nr. days with interval session                42798 non-null  float64
 7   total km Z3-4                                 42798 non-null  float64
 8   max km Z3-4 one day                           42798 non-null  float64
 9   total km Z5-T1-T2                             42798 non-null  float64
 10  max km Z5-T1-T2 one day                       42798 non-null  float64
 11  total hours alternative training              42798 non-null  float64
 12  nr. strength trainings                        42798 non-null  float64
 13  avg exertion                                  42798 non-null  float64
 14  min exertion                                  42798 non-null  float64
 15  max exertion                                  42798 non-null  float64
 16  avg training success                          42798 non-null  float64
 17  min training success                          42798 non-null  float64
 18  max training success                          42798 non-null  float64
 19  avg recovery                                  42798 non-null  float64
 20  min recovery                                  42798 non-null  float64
 21  max recovery                                  42798 non-null  float64
 22  nr. sessions.1                                42798 non-null  float64
 23  nr. rest days.1                               42798 non-null  float64
 24  total kms.1                                   42798 non-null  float64
 25  max km one day.1                              42798 non-null  float64
 26  total km Z3-Z4-Z5-T1-T2.1                     42798 non-null  float64
 27  nr. tough sessions (effort in Z5, T1 or T2).1 42798 non-null  float64
 28  nr. days with interval session.1              42798 non-null  float64
 29  total km Z3-4.1                               42798 non-null  float64
 30  max km Z3-4 one day.1                         42798 non-null  float64
 31  total km Z5-T1-T2.1                           42798 non-null  float64
 32  max km Z5-T1-T2 one day.1                     42798 non-null  float64
 33  total hours alternative training.1            42798 non-null  float64
 34  nr. strength trainings.1                      42798 non-null  float64
 35  avg exertion.1                                42798 non-null  float64
 36  min exertion.1                                42798 non-null  float64
 37  max exertion.1                                42798 non-null  float64
 38  avg training success.1                        42798 non-null  float64
 39  min training success.1                        42798 non-null  float64
 40  max training success.1                        42798 non-null  float64
 41  avg recovery.1                                42798 non-null  float64
 42  min recovery.1                                42798 non-null  float64
 43  max recovery.1                                42798 non-null  float64
 44  nr. sessions.2                                42798 non-null  float64
 45  nr. rest days.2                               42798 non-null  float64
 46  total kms.2                                   42798 non-null  float64
 47  max km one day.2                              42798 non-null  float64
 48  total km Z3-Z4-Z5-T1-T2.2                     42798 non-null  float64
 49  nr. tough sessions (effort in Z5, T1 or T2).2 42798 non-null  float64
 50  nr. days with interval session.2              42798 non-null  float64
```

```
 51  total km Z3-4.2                          42798 non-null  float64
 52  max km Z3-4 one day.2                    42798 non-null  float64
 53  total km Z5-T1-T2.2                      42798 non-null  float64
 54  max km Z5-T1-T2 one day.2                42798 non-null  float64
 55  total hours alternative training.2       42798 non-null  float64
 56  nr. strength trainings.2                 42798 non-null  float64
 57  avg exertion.2                           42798 non-null  float64
 58  min exertion.2                           42798 non-null  float64
 59  max exertion.2                           42798 non-null  float64
 60  avg training success.2                   42798 non-null  float64
 61  min training success.2                   42798 non-null  float64
 62  max training success.2                   42798 non-null  float64
 63  avg recovery.2                           42798 non-null  float64
 64  min recovery.2                           42798 non-null  float64
 65  max recovery.2                           42798 non-null  float64
 66  Athlete ID                               42798 non-null  int64
 67  injury                                   42798 non-null  int64
 68  rel total kms week 0_1                   42798 non-null  float64
 69  rel total kms week 0_2                   42798 non-null  float64
 70  rel total kms week 1_2                   42798 non-null  float64
 71  Date                                     42798 non-null  int64
dtypes: float64(69), int64(3)
memory usage: 23.5 MB
```

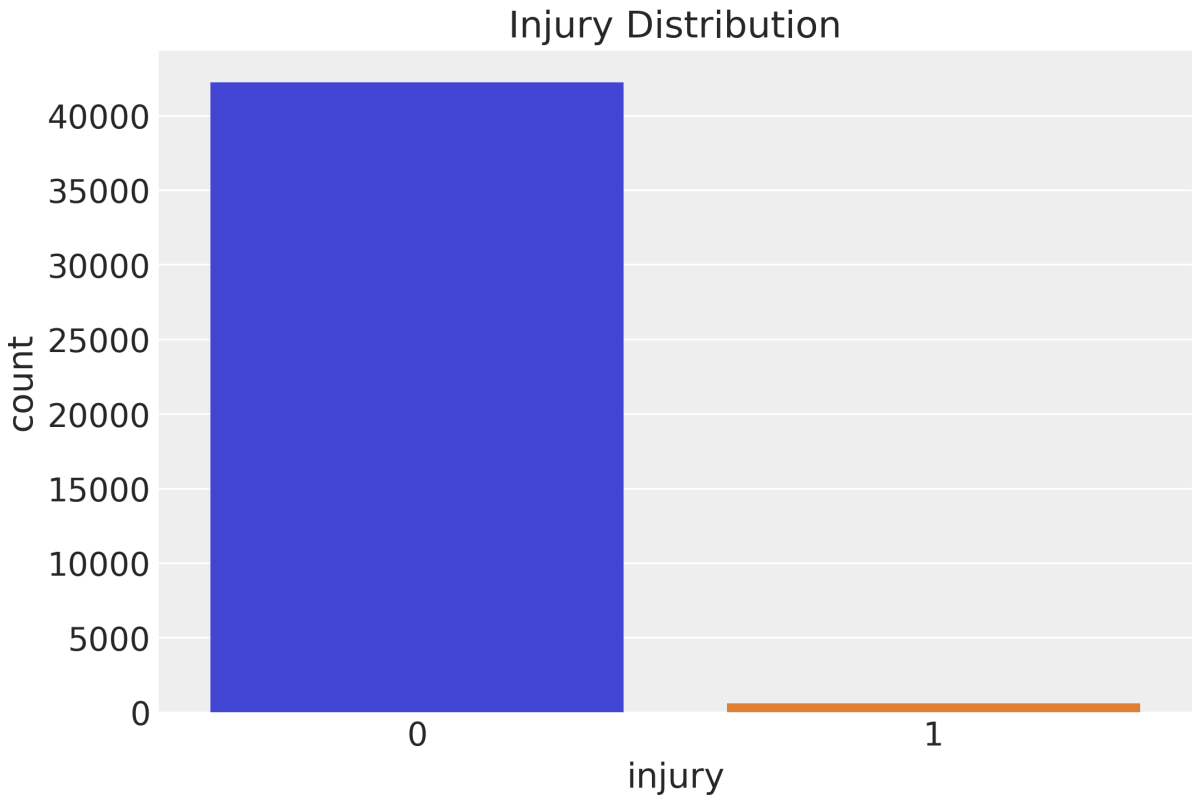
Correlation Analysis

In [5]:
```
correlation_matrix = week.corr()
sns.heatmap(correlation_matrix, cmap='coolwarm', annot=True)
plt.title('Correlation Matrix')
plt.show()
```



Injury Distribution

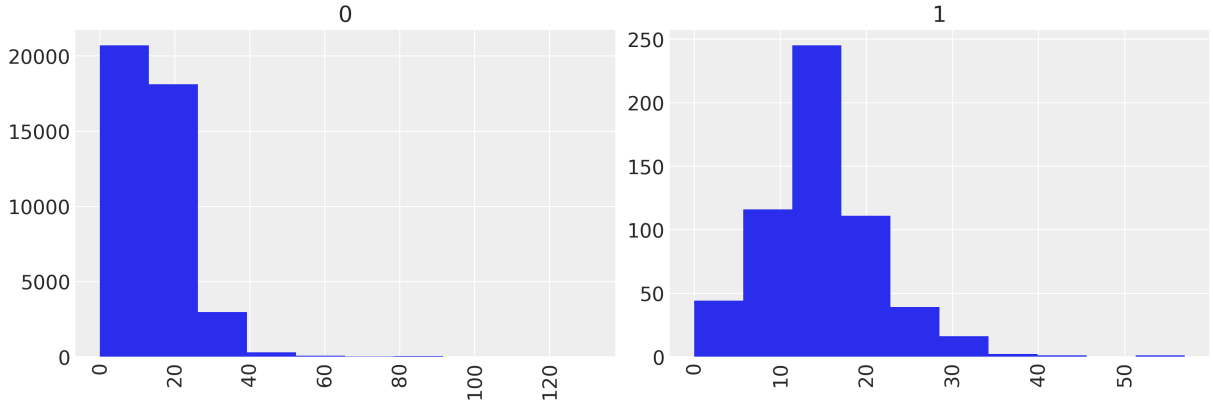
In [6]:
```
sns.countplot(x='injury', data = week)
plt.title("Injury Distribution")
plt.show()
```



Histogram of maximum kilometers ran per day by injury

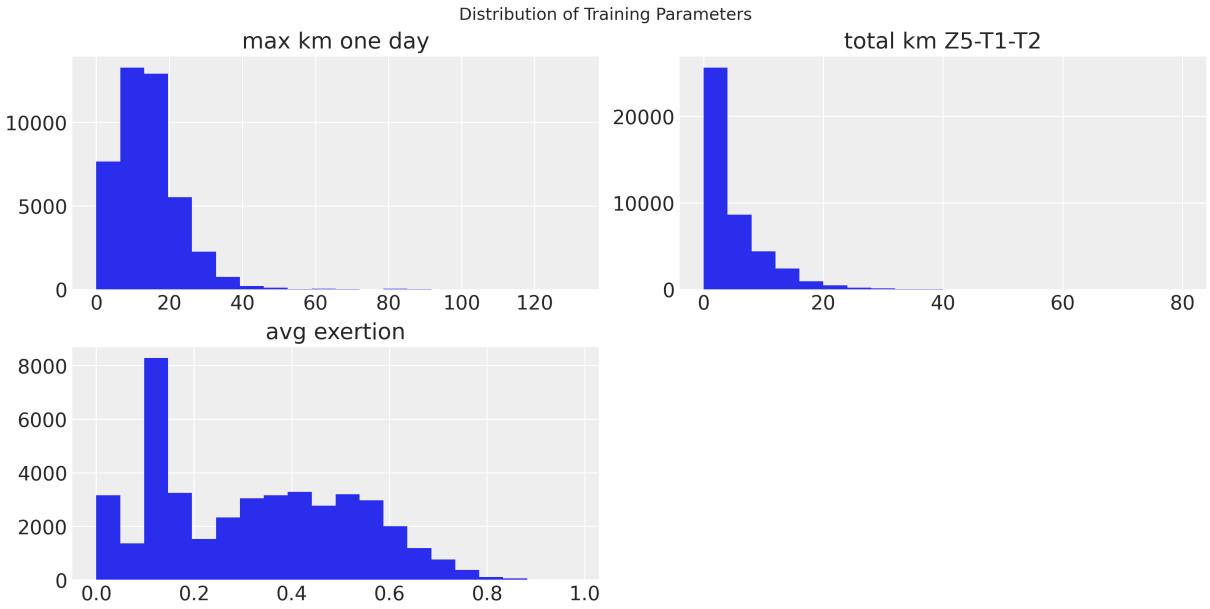
In [7]:
```
week.hist("max km one day", by="injury", figsize=(12, 4))
```

Out[7]:
```
array([<Axes: title={'center': '0'}>, <Axes: title={'center': '1'}>],
      dtype=object)
```



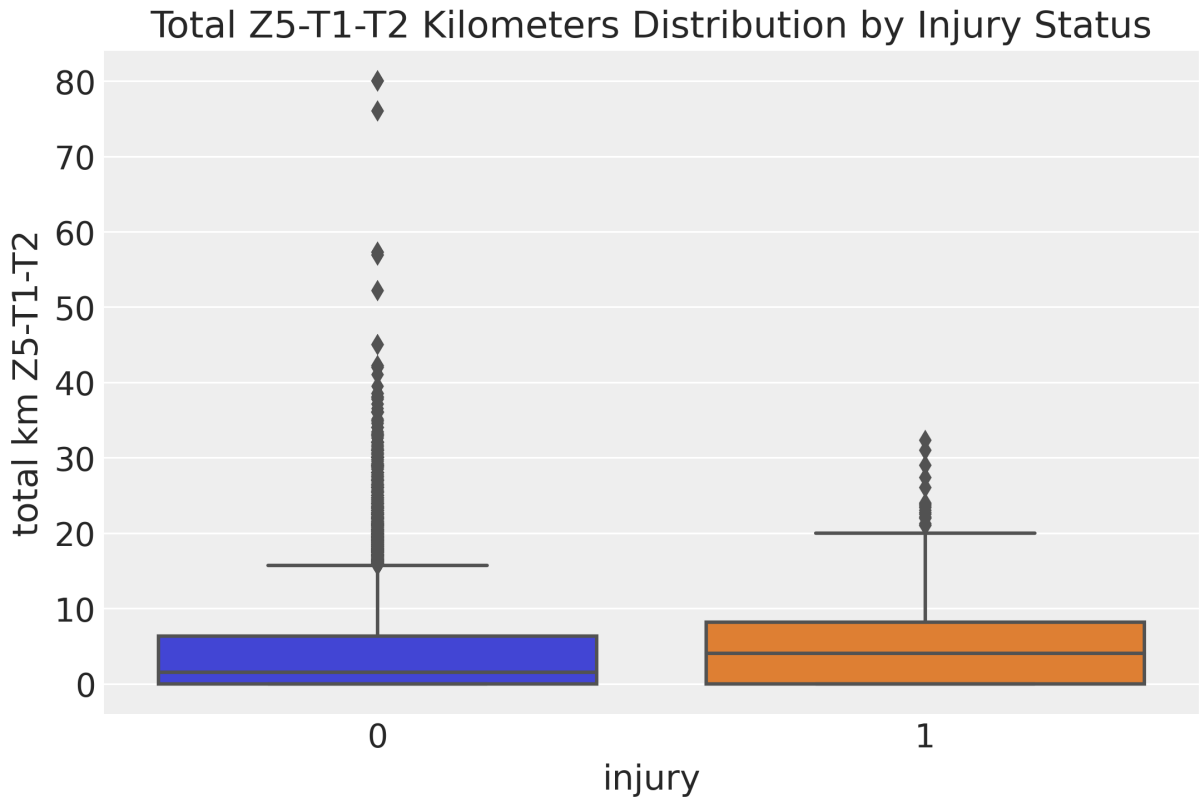
Distribution of Training Parameters

In [8]:
```
week[['max km one day', 'total km Z5-T1-T2', 'avg exertion']].hist(bins=20,
plt.suptitle('Distribution of Training Parameters')
plt.show()
```



Box Plot

In [9]:
```
sns.boxplot(x='injury', y='total km Z5-T1-T2', data=week)
plt.title('Total Z5-T1-T2 Kilometers Distribution by Injury Status')
plt.show()
```



Data Processing

Selecting relevant variables

In [10]:
```
week = week[["injury",
             "Athlete ID",
             "max km one day",
             "total km Z5-T1-T2",
             "avg exertion"
             ]]
```

Renaming columns

In [11]:
```
week = week.copy()
week.rename(columns = {"max km one day":"max_km_oneday",
                       "total km Z5-T1-T2":"total_kmZ5",
                       "avg exertion":"avg_exertion",
                       "Athlete ID":"id"}, inplace = True)
```

Missing Values

In [12]:
```
week.isnull().values.any()
```

Out[12]:
```
False
```

Outliers

Calculating the IQR for each variable

In [13]:
```
Q1 = week[['max_km_oneday', 'total_kmZ5', 'avg_exertion']].quantile(0.25)
Q3 = week[['max_km_oneday', 'total_kmZ5','avg_exertion']].quantile(0.75)
IQR = Q3 - Q1
```

Identify outliers using IQR

In [14]:
```
outliers_iqr = ((week[[ 'max_km_oneday', 'total_kmZ5', 'avg_exertion']] < (Q
outliers_iqr.describe()
```

Out[14]:

	max_km_oneday	total_kmZ5	avg_exertion
count	42798	42798	42798
unique	2	2	1
top	False	False	False
freq	41543	40964	42798

Drop rows containing outliers

In [15]:
```
week_no_outliers = week.drop(week.index[outliers_iqr.any(axis=1)])
```

Verify that outliers are removed

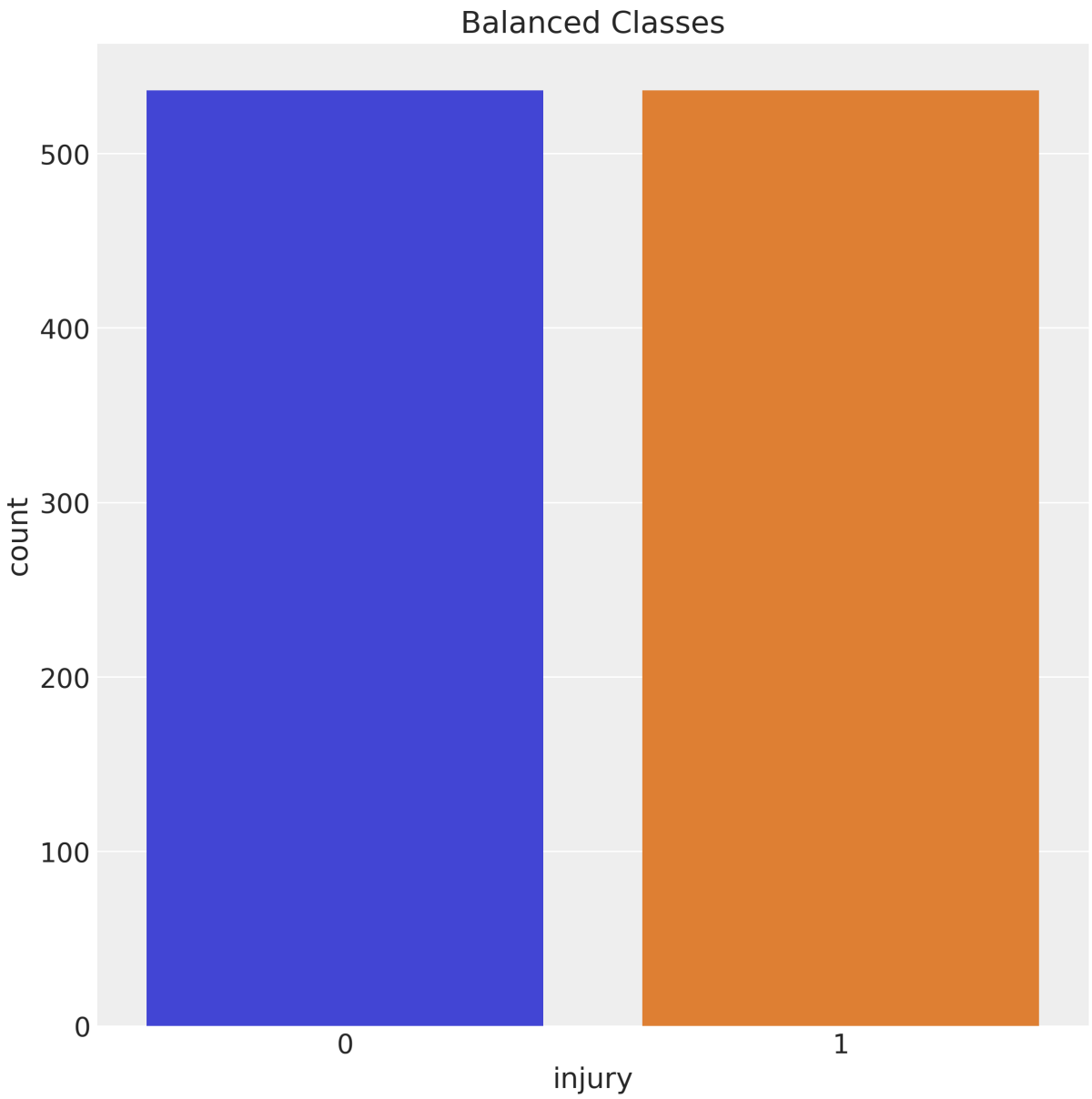
In [16]:
```
print("Original DataFrame shape:", week.shape)
print("DataFrame shape after removing outliers:", week_no_outliers.shape)
```

```
Original DataFrame shape: (42798, 5)
DataFrame shape after removing outliers: (39912, 5)
```

Resampling

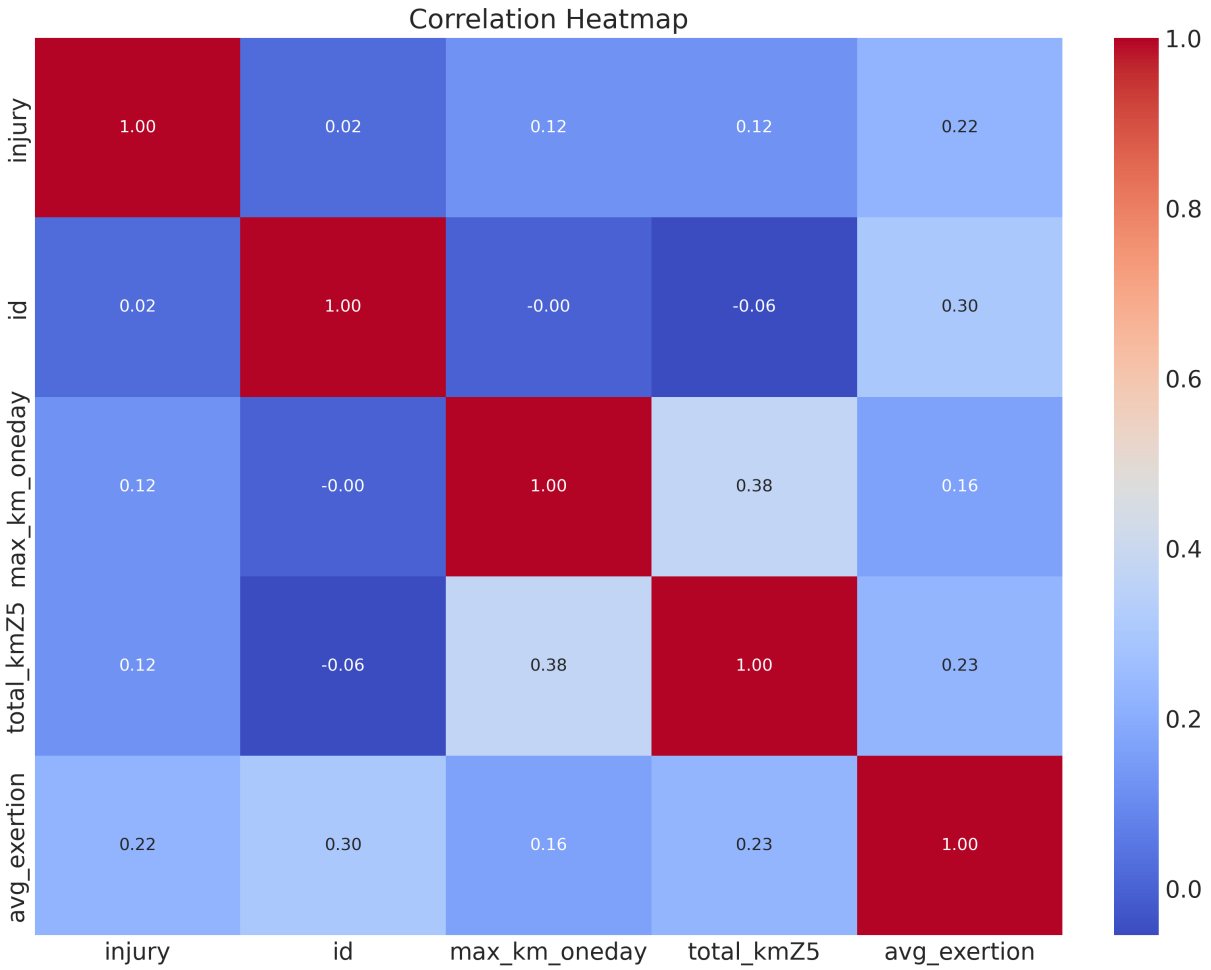
We chose to resample from the original data to account for the small proportion of injury observations compared to non-injured. This way, the response variable is more balanced.

In [17]:
```
week1 = week_no_outliers.sort_values(by='id')
cts = week1["injury"].value_counts().get(1, 0)
shuff_week1 = week1.sample(frac=1, random_state=4)
inj_week1 = shuff_week1.loc[shuff_week1['injury'] == 1]
non_inj_week1 = shuff_week1.loc[shuff_week1['injury'] == 0].sample(n=cts)
norm_week = pd.concat([inj_week1, non_inj_week1])

import warnings
# Ignore FutureWarning related to is_categorical_dtype
warnings.filterwarnings("ignore", category=FutureWarning)

plt.figure(figsize=(8, 8))
sns.countplot(x='injury', data=norm_week)
plt.title("Balanced Classes")
plt.show()
```



Correlation Heatmap with the cleaned data

In [18]:
```
# Calculate correlation matrix
correlation_matrix = norm_week.corr()

# Plot heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```



Categorizing the response variable

In [20]:
```
norm_week['injury'] = norm_week['injury'].astype('category')
norm_week['injury']
```

Out[20]:
```
42714    1
42535    1
42726    1
42227    1
42718    1
        ..
13551    0
12073    0
1551     0
34939    0
35194    0
Name: injury, Length: 1072, dtype: category
Categories (2, int64): [0, 1]
```

Scaling predictors

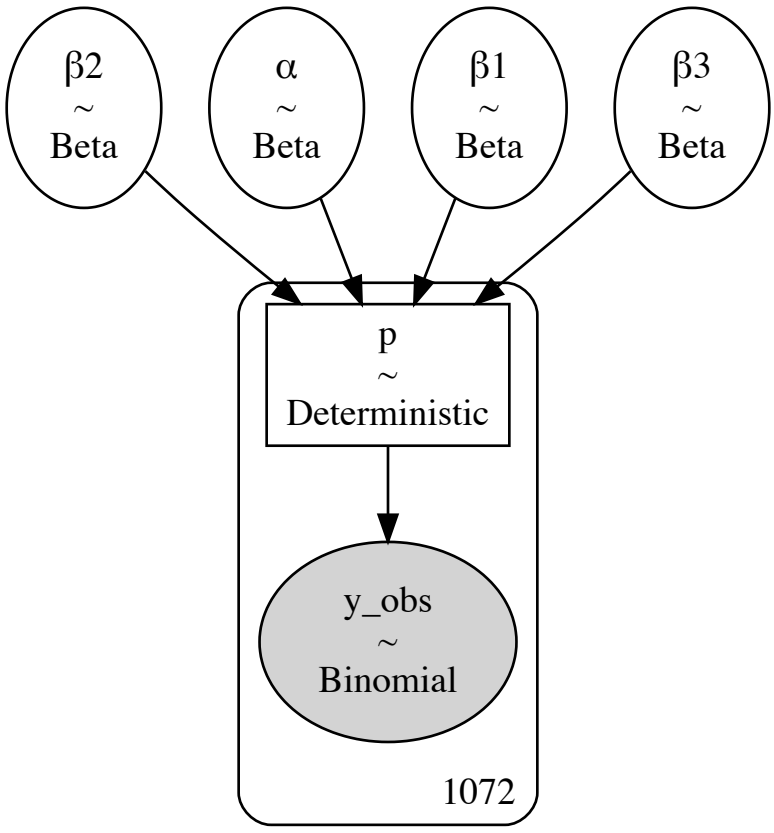
In [21]:
```
#Predictors
predictors = ["max_km_oneday", "total_kmZ5", "avg_exertion"]
n = norm_week.shape[0]
```

```
#Subsetting Data
x_num = norm_week[predictors]
y = norm_week['injury']

# Standardize numeric columns, to mean 0 variance 1
mean = x_num.mean()
std = x_num.std()
x_num = np.array((x_num - mean) / std)

X = x_num
k = X.shape[1]
```

Model Creation

In [27]:
```
with pm.Model() as glm:
    α = pm.Beta('α', alpha = 1, beta = 1) #intercept
    β1 = pm.Beta('β1', alpha = 1, beta = 1) #max kilometers in one day
    β2 = pm.Beta('β2', alpha = 1, beta = 1) #total kilometers ran with anaer
    β3 = pm.Beta('β3', alpha = 1, beta = 1) #average exertion
    μ =  α + β1 * X[:,0] + β2 * X[:,1] + β3 * X[:,2]
    p = pm.Deterministic("p", pm.invlogit(μ))
    y_obs = pm.Binomial('y_obs', n = 1, p = p, observed = y)

pm.model_to_graphviz(glm)
```

Out[27]:



HMC Sampling

In [28]:
```
#HMC Trace
with glm:
    trace = pm.sample(1000, tune = 2000, random_seed = random_seed, cores =
    glm_trace = pm.to_inference_data(trace=trace, log_likelihood=True)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [α, β1, β2, β3]
```

100.00% [12000/12000 00:06<00:00 Sampling 4 chains, 0 divergences]

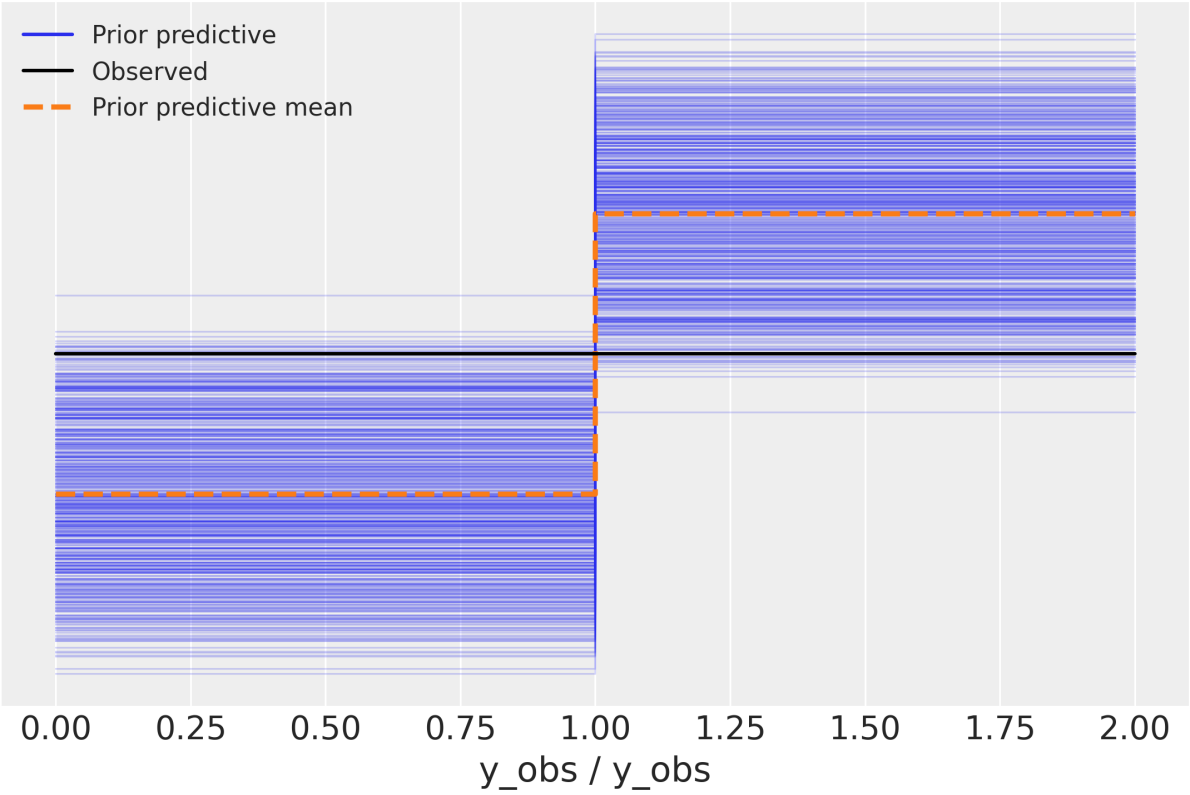
```
Sampling 4 chains for 2_000 tune and 1_000 draw iterations (8_000 + 4_000 dr
aws total) took 6 seconds.
```

Prior Predictive Check

In [29]:
```
with glm:
    prior_p = pm.sample_prior_predictive(random_seed=random_seed)
```

```
Sampling: [y_obs, α, β1, β2, β3]
```

In [30]:
```
az.plot_ppc(prior_p, group="prior");
```



ADVI Approximation

In [31]:
```
with glm:
    advi_fit = pm.fit(10000, method='advi', random_seed = random_seed)
```

100.00% [10000/10000 00:01<00:00
Average Loss = 719.27]

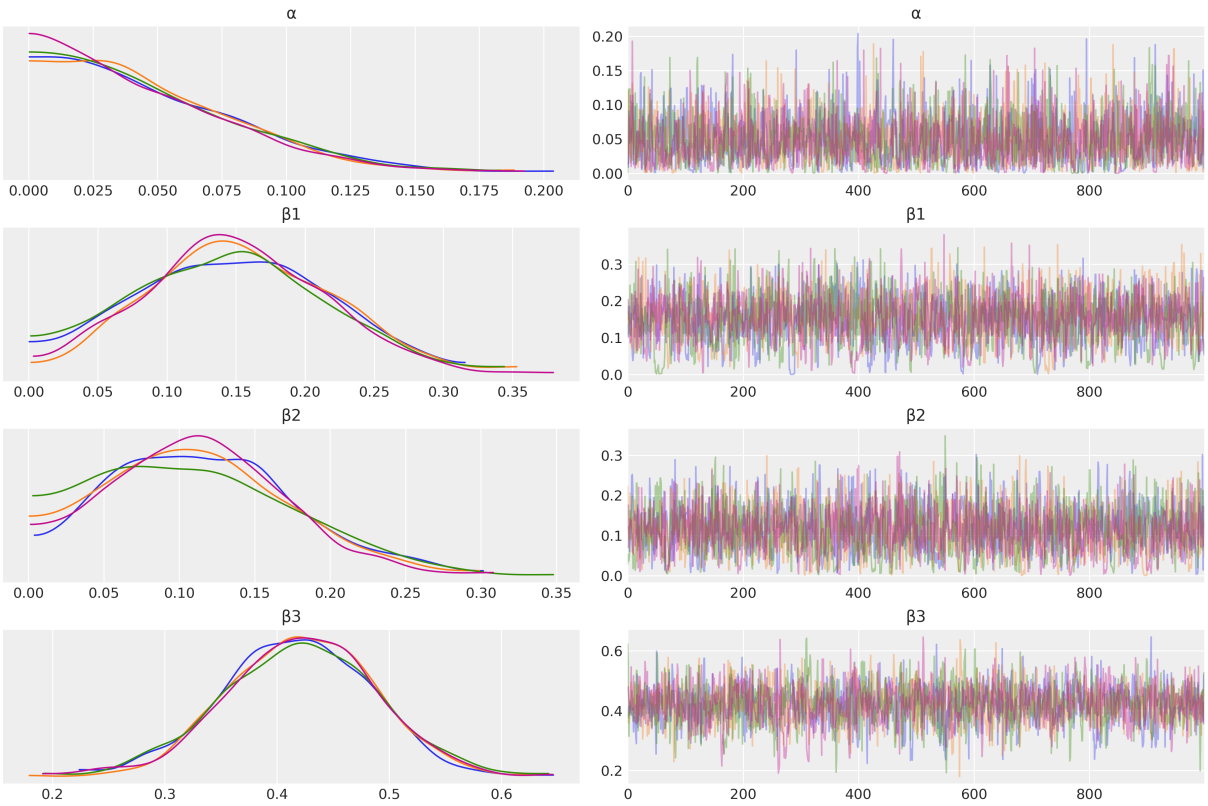
```
Finished [100%]: Average Loss = 719.27
```

In [32]:
```
advi_samples = 1000
advi_trace = advi_fit.sample(advi_samples)
```

HMC Evaluation

Trace Plot

In [33]:
```
az.plot_trace(glm_trace, compact = False, var_names = ['α','β1','β2', 'β3'])
```



WAIC and LOO

In [34]:
```
#WAIC
az.waic(glm_trace)
```

Out[34]:
```
Computed from 4000 posterior samples and 1072 observations log-likelihood m
atrix.

          Estimate       SE
elpd_waic  -713.78     7.90
p_waic        3.07        -
```

In [35]:
```
az.loo(glm_trace)
```

Out[35]:
```
Computed from 4000 posterior samples and 1072 observations log-likelihood m
atrix.

         Estimate       SE
elpd_loo  -713.79     7.90
p_loo        3.07        -
------

Pareto k diagnostic values:
                         Count   Pct.
(-Inf, 0.5]   (good)     1072  100.0%
 (0.5, 0.7]   (ok)          0    0.0%
   (0.7, 1]   (bad)         0    0.0%
   (1, Inf)   (very bad)    0    0.0%
```

Posterior Predictive Check

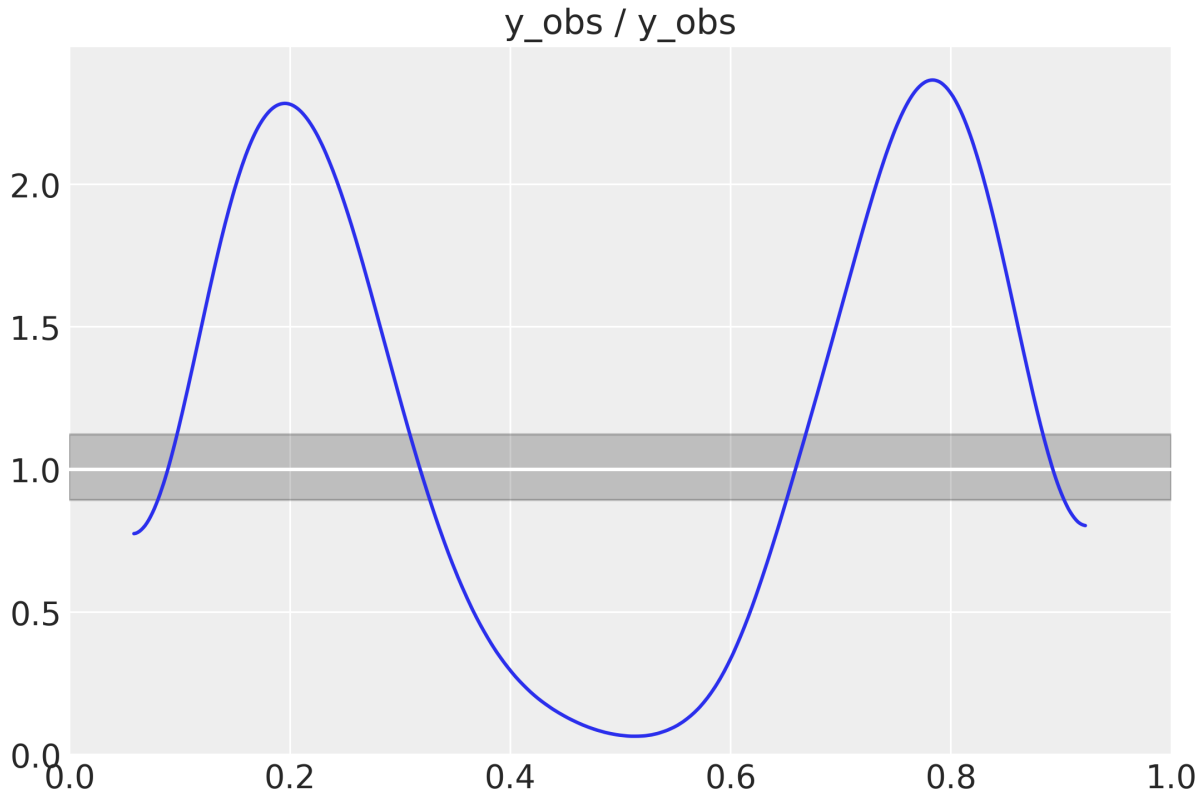
In [36]:
```
with glm:
    ppc = pm.sample_posterior_predictive(trace = glm_trace, random_seed = ra
```

```
Sampling: [y_obs]
```

100.00% [4000/4000 00:00<00:00]

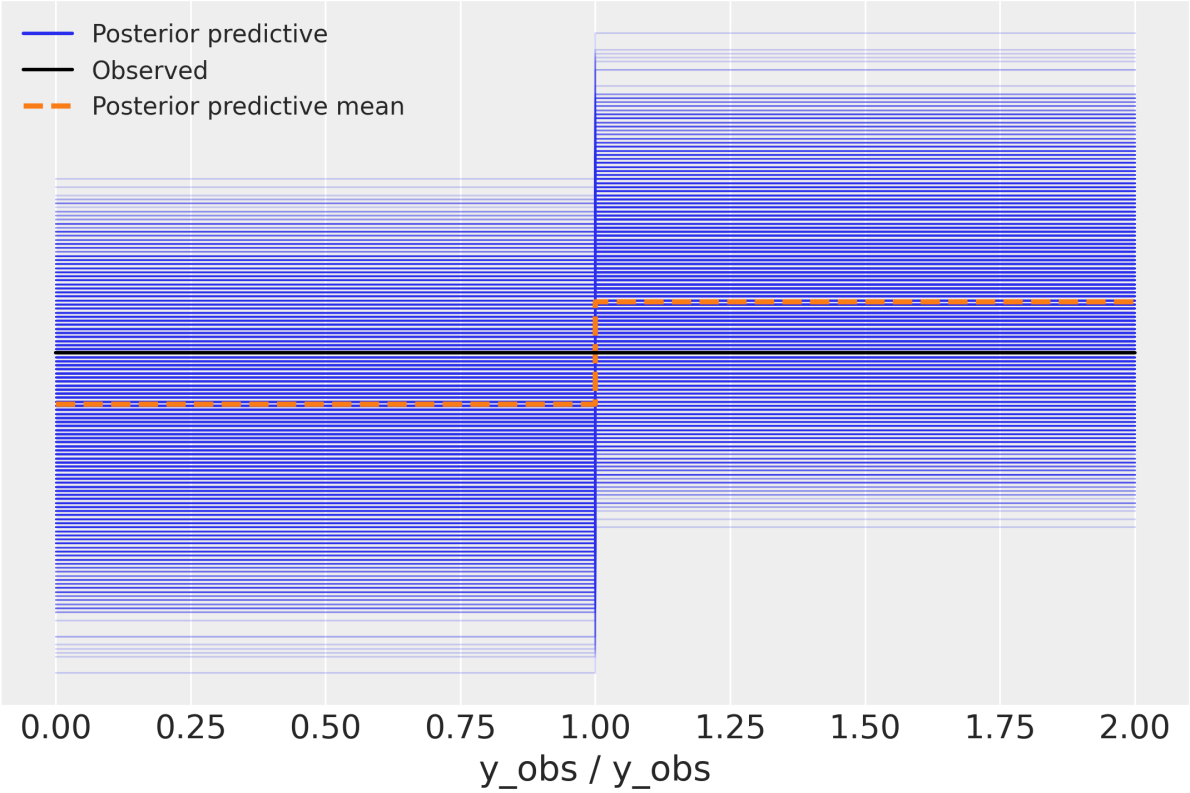
In [37]:
```
az.plot_bpv(ppc)
```

Out[37]:
```
<Axes: title={'center': 'y_obs / y_obs'}>
```



In [38]:
```
az.plot_ppc(ppc)
```

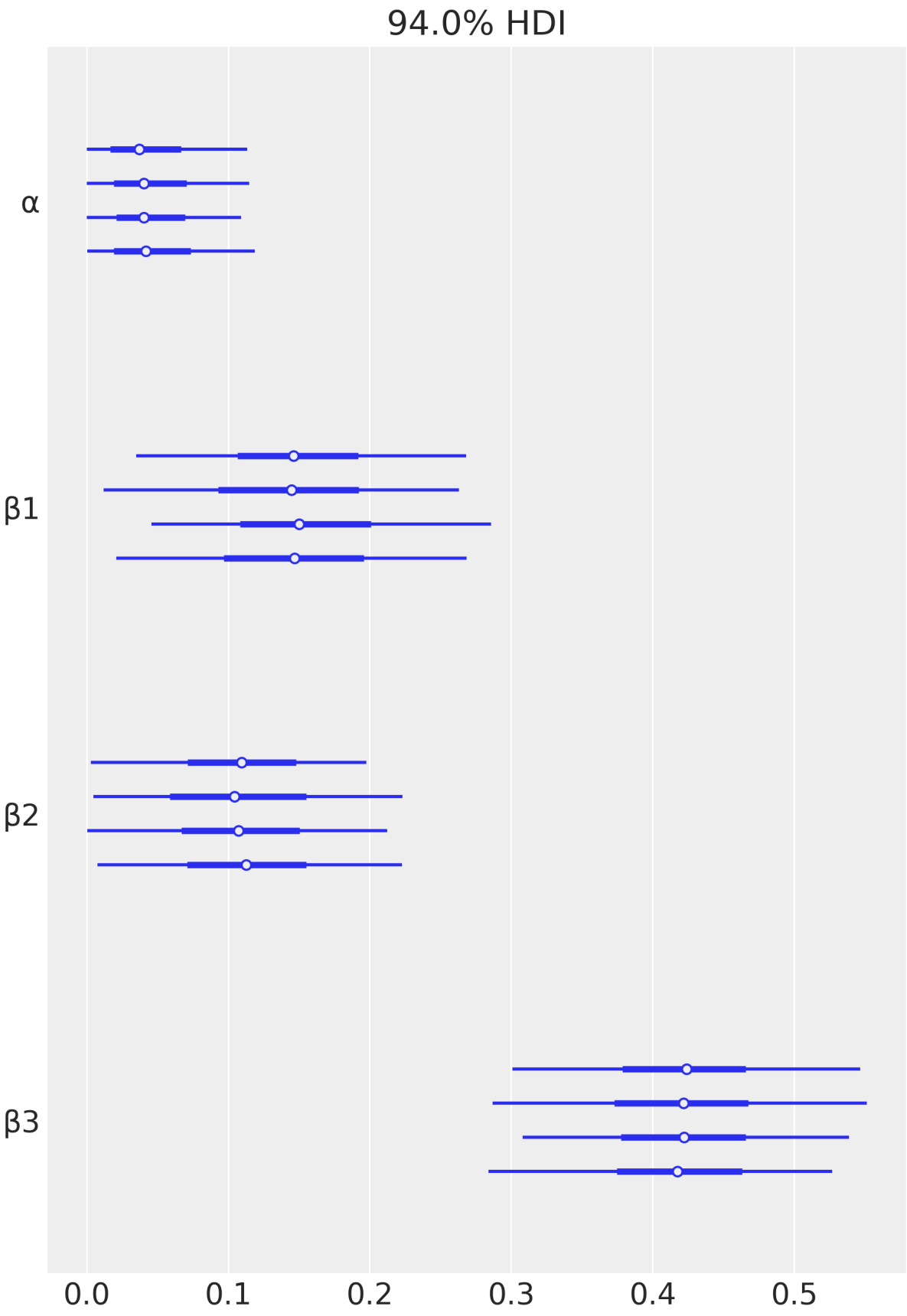
Out[38]:
```
<Axes: xlabel='y_obs / y_obs'>
```



Forest Plot

In [39]:
```
az.plot_forest(glm_trace, var_names = ['α',"β1",'β2', 'β3'])
```

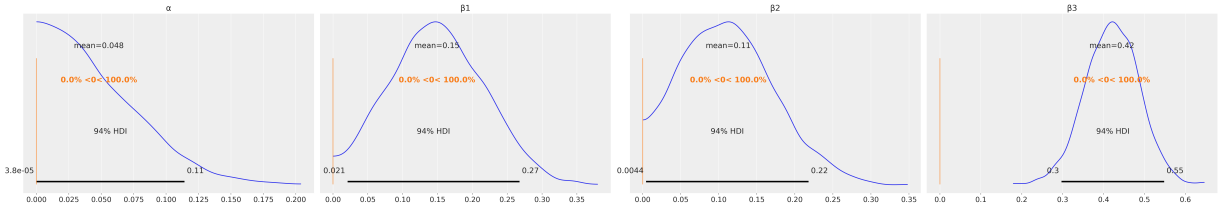
Out[39]:
```
array([<Axes: title={'center': '94.0% HDI'}>], dtype=object)
```



Posterior Parameters

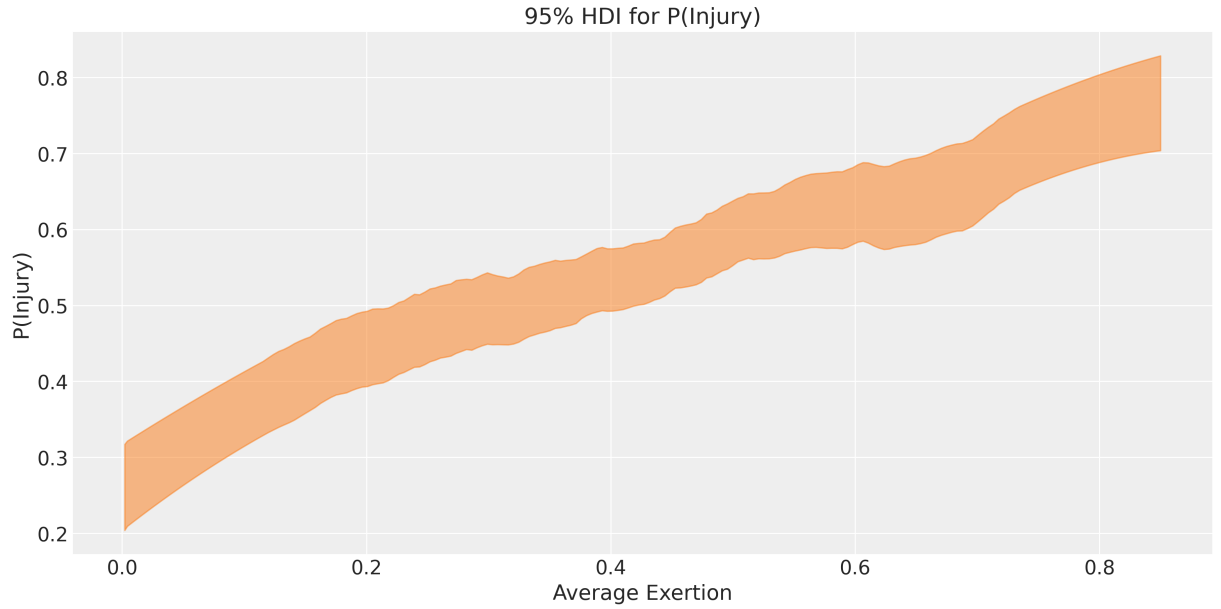
In [40]:
```
az.plot_posterior(glm_trace, var_names=['α',"β1",'β2', 'β3'], ref_val = 0)
```

Out[40]:
```
array([<Axes: title={'center': 'α'}>, <Axes: title={'center': 'β1'}>,
       <Axes: title={'center': 'β2'}>, <Axes: title={'center': 'β3'}>],
      dtype=object)
```



95% HDI plot of P(Injury) by Average Exertion

In [41]:
```
_, ax = plt.subplots(figsize=(12, 6))
rates = glm_trace.posterior["p"]
az.plot_hdi(norm_week["avg_exertion"], rates, smooth=True, hdi_prob=0.95)
rate_mean = glm_trace.posterior["p"].mean(dim=["draw", "chain"])
ax.set_title("95% HDI for P(Injury)")
ax.set_xlabel("Average Exertion")
24
ax.set_ylabel("P(Injury)");
```



Summary

In [42]:
```
az.summary(glm_trace, var_names = ['α',"β1",'β2', 'β3'])
```

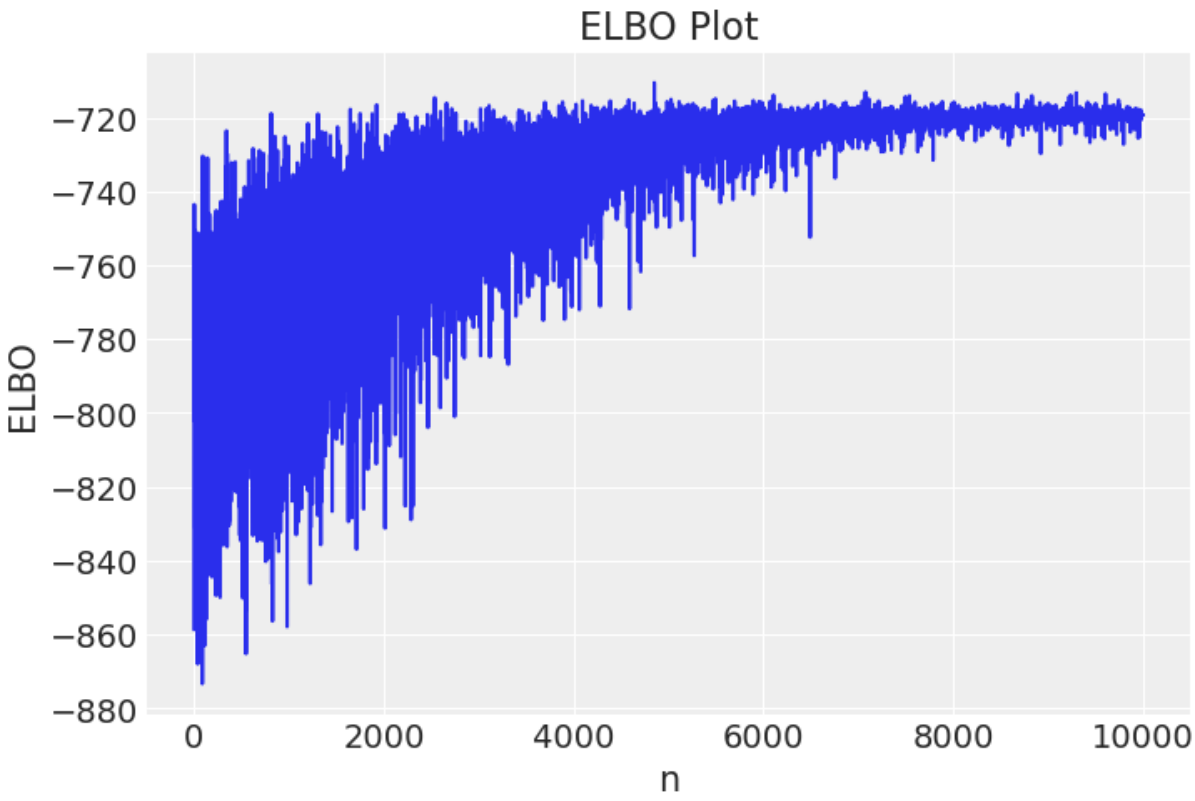
Out[42]:

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
α	0.048	0.037	0.000	0.114	0.001	0.000	1919.0	1078.0	1.0
β1	0.149	0.067	0.021	0.268	0.002	0.001	1560.0	814.0	1.0
β2	0.112	0.060	0.004	0.218	0.001	0.001	2097.0	1331.0	1.0
β3	0.420	0.066	0.297	0.548	0.001	0.001	2250.0	1553.0	1.0

ADVI Evaluation

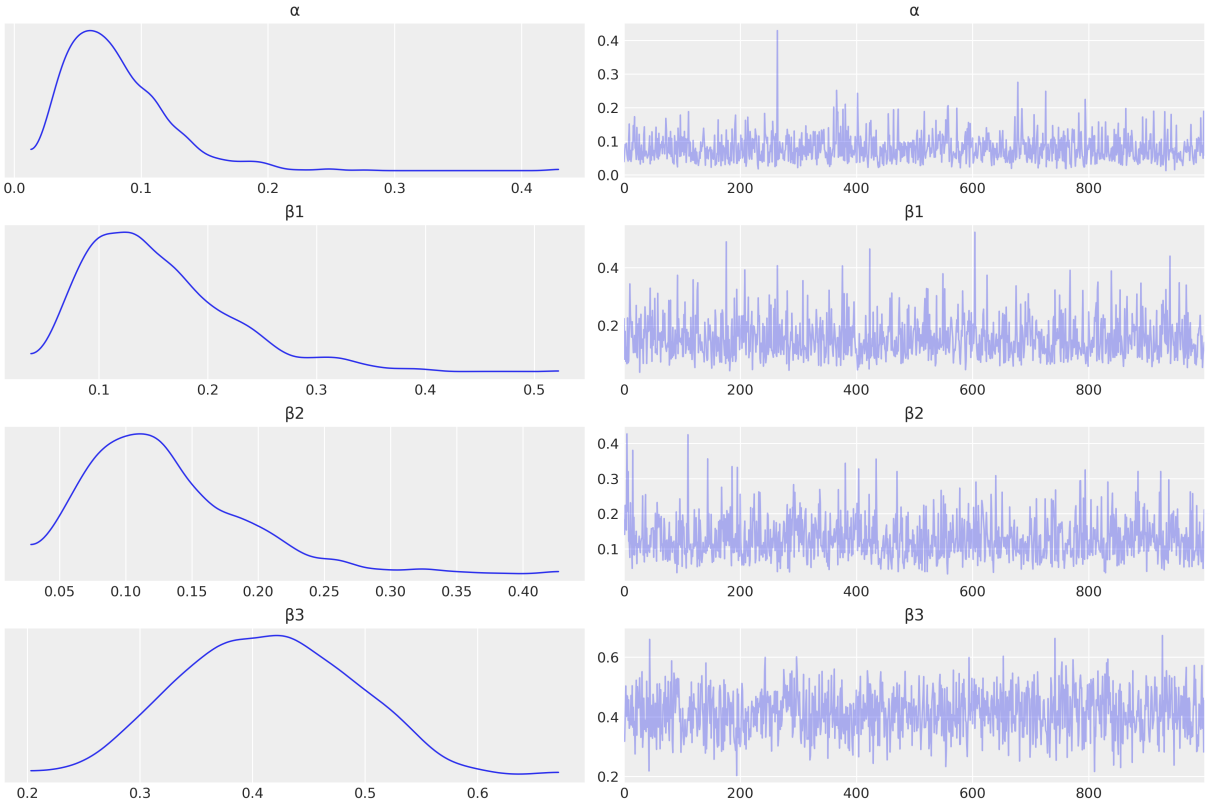
ELBO Plot

In [43]:
```
advi_elbo = pd.DataFrame(
    {'ELBO': -advi_fit.hist,
     'n': np.arange(advi_fit.hist.shape[0])})

plt.figure(dpi=100)
ax = sns.lineplot(y='ELBO', x='n', data=advi_elbo)
ax.set_title("ELBO Plot")
plt.show()
```



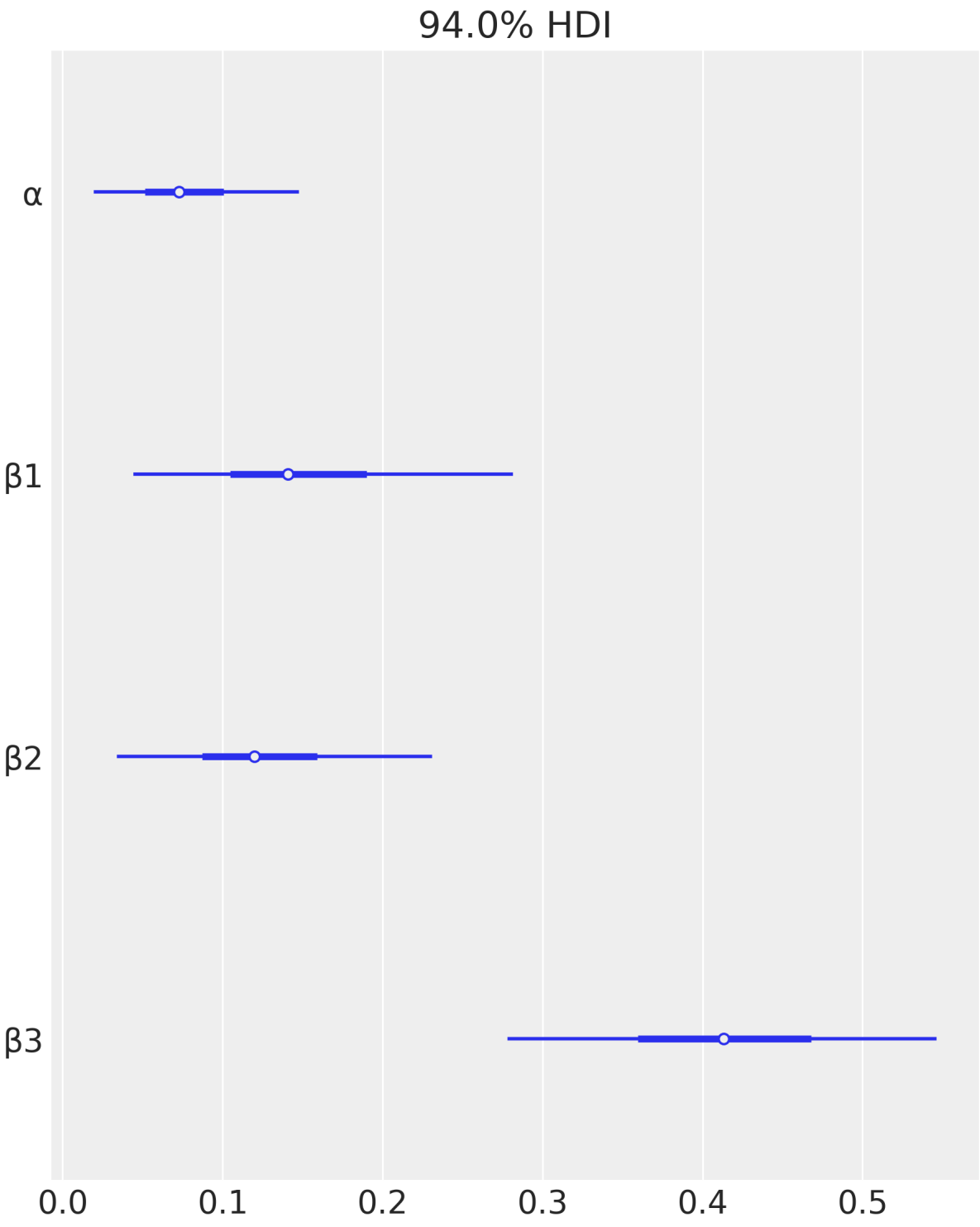
Trace Plot

In [44]:
```
az.plot_trace(advi_trace, compact = False, var_names = ['α','β1','β2', 'β3']
```



Forest Plot

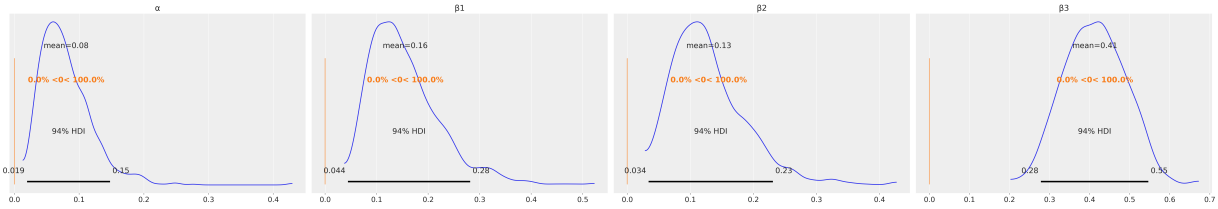
In [45]:
```
with glm:
    az.plot_forest(advi_trace, var_names=['α',"β1",'β2', 'β3'])
```



Posterior Parameters

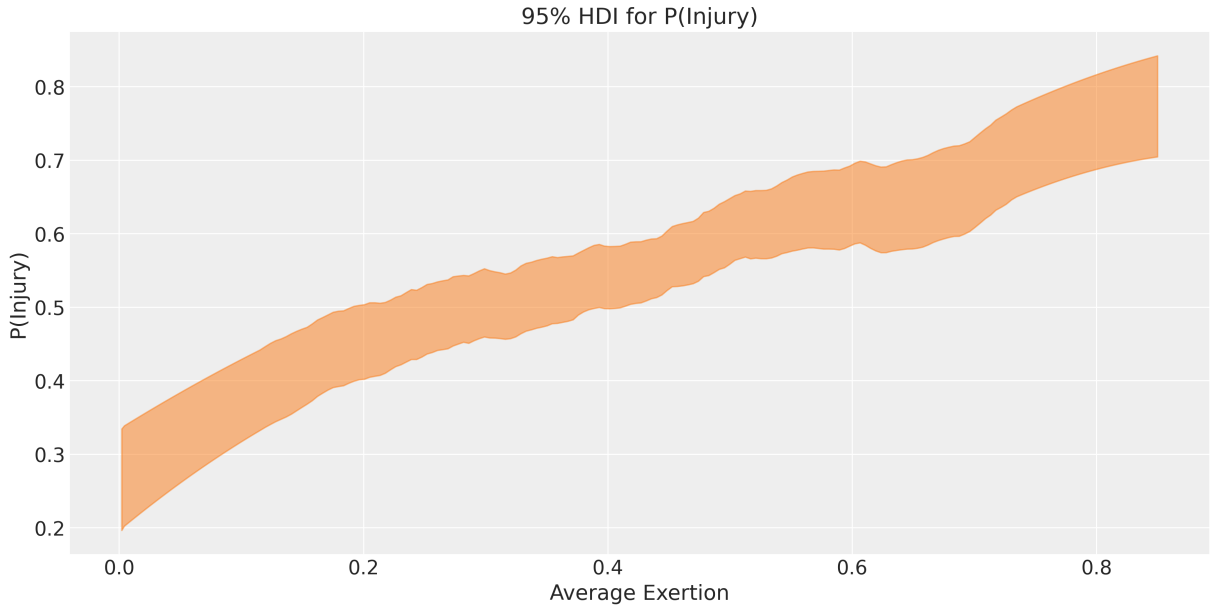
In [46]:
```
az.plot_posterior(
    advi_trace,
    var_names=['α',"β1",'β2', 'β3'], ref_val = 0
)
```

Out[46]:
```
array([<Axes: title={'center': 'α'}>, <Axes: title={'center': 'β1'}>,
       <Axes: title={'center': 'β2'}>, <Axes: title={'center': 'β3'}>],
      dtype=object)
```



95% HDI plot of P(Injury) by Average Exertion

In [47]:
```
_, ax = plt.subplots(figsize=(12, 6))
rates = advi_trace.posterior["p"]
az.plot_hdi(norm_week["avg_exertion"], rates, smooth=True, hdi_prob=0.95)
rate_mean = advi_trace.posterior["p"].mean(dim=["draw", "chain"])
ax.set_title("95% HDI for P(Injury)")
ax.set_xlabel("Average Exertion")
24
ax.set_ylabel("P(Injury)");
```



Summary

In [48]:
```
az.summary(advi_trace, var_names = ['α',"β1",'β2', 'β3'])
```

```
arviz - WARNING - Shape validation failed: input_shape: (1, 1000), minimum_s
hape: (chains=2, draws=4)
```

Out[48]:

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
α	0.080	0.040	0.019	0.148	0.001	0.001	901.0	853.0	NaN
β1	0.155	0.069	0.044	0.281	0.002	0.002	1002.0	1026.0	NaN
β2	0.129	0.059	0.034	0.231	0.002	0.001	784.0	954.0	NaN
β3	0.413	0.076	0.278	0.546	0.002	0.002	997.0	981.0	NaN

In [ ]: