lab04

June 12, 2021

1 Exercise 1

The aim of the exercise is to estimate the LD50 - the dose level at which the probability of death is 50%. We assuming that logistic model describes the data.

[1]:
```
import scipy.stats as stats
import numpy as np
import arviz as az
import matplotlib.pyplot as plt
import matplotlib as mpl
from cmdstanpy import CmdStanModel
import pandas as pd
from scipy.special import expit  # aka logistic
```

Below is an example of real data from an experiment in which twenty animals were tested, five at each of four dose levels.

[2]:
```
# data
x = np.array([-0.86, -0.30, -0.05, 0.73])
n = np.array([5, 5, 5, 5])
y = np.array([0, 1, 3, 5])
pd.DataFrame({'Dose xi':x,'Dose of animals ni':n,'Number of deaths yi':y})
```

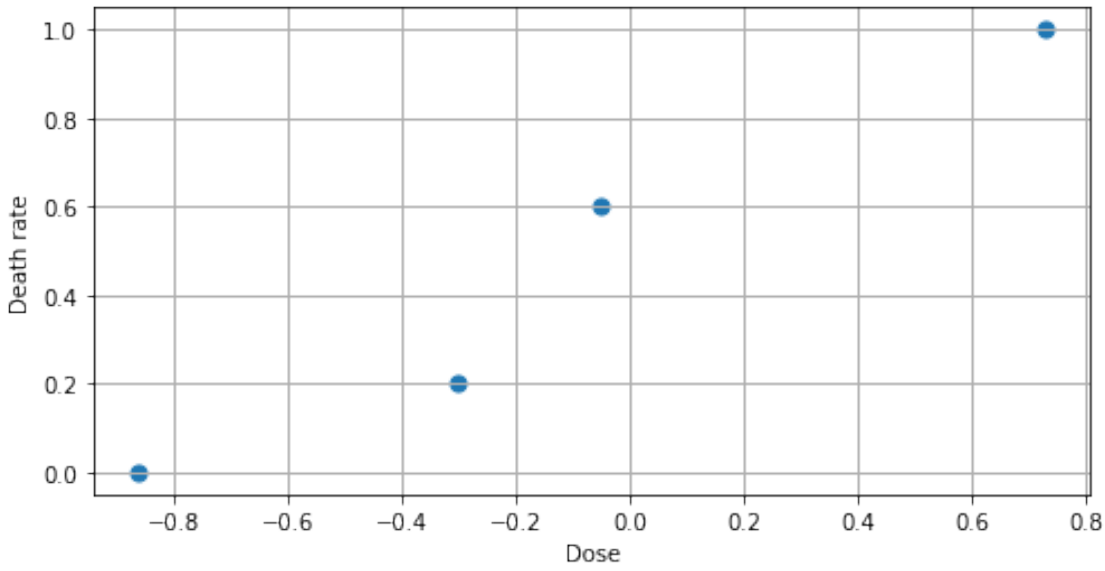
[2]:

	Dose xi	Dose of animals ni	Number of deaths yi
0	-0.86	5	0
1	-0.30	5	1
2	-0.05	5	3
3	0.73	5	5

Using Stan create the logistic model for presented data, estimate the LD50 and interpret the result.

Plot dose against death rate

[3]:
```
# plot the data
fig, ax = plt.subplots(figsize=(8, 4))
ax.grid(linestyle='-', linewidth=1)
ax.scatter(x, y/5, 50)
ax.set_xlabel('Dose')
ax.set_ylabel('Death rate');
```



1.1 Model

Knowing the outcome of dosage tests on animals in each group is exchangeable and using model of the dose-response relation we can set present our model as:

$$y_i|\theta_i \sim \text{Bin}(n_i, \theta_i),$$

$$\text{logit}(\theta_i) = \alpha + \beta x_i,$$

where logit is a logistic regression model. Because we don't have any information about the type of drug used or kind of animals it was tested on, we have to assume none of weakly informative priors. For logit it's suggested to use student-t distribution (Stan Wiki) we will keep it in mind as we continue our exercise. So our model looks like:

$$y_i|\theta_i \sim \text{Bin}(n_i, \theta_i)$$

$$\text{logit}(\theta_i) = \alpha + \beta x_i$$

Though it's not suggested to use uniform prior, we can still do it to perform simple analysis and find better priors for our data. For our first analysis we can use uniform priors given as:

$$\alpha \sim \text{Uniform}(-10, 10)$$

$$\beta \sim \text{Uniform}(-10, 10)$$

[4]:
```
with open('model_prior_uni1.stan', 'r') as file:
    print(file.read())
```

```
data {
  int N;
  int A[N];
```

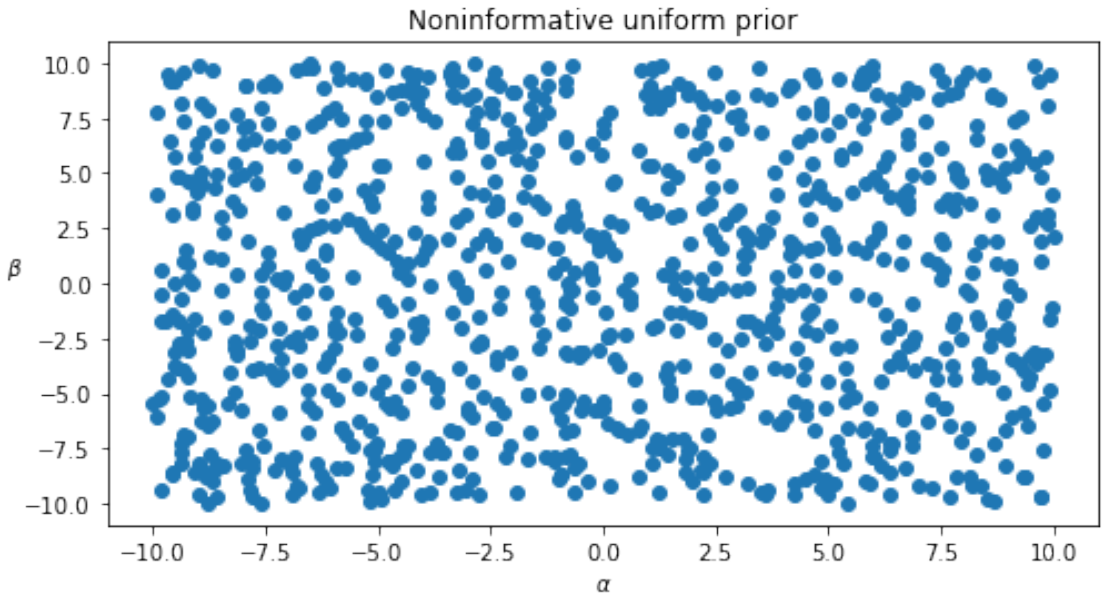
```
  real X[N];
}

generated quantities {
  real alpha=uniform_rng(-10,10);
  real beta=uniform_rng(-10,10);

  int y_sim[N];
  for (k in 1:N) {
    y_sim[k] = binomial_rng(A[k],inv_logit(alpha+beta*X[k]));
  }
}
```

[5]:
```
model_prior_uni=CmdStanModel(stan_file='model_prior_uni1.stan')
sim=model_prior_uni.sample(data=dict(N=len(x),A=n,X=x),
                      fixed_param=True,
                      iter_warmup=0,
                      chains=1)
```

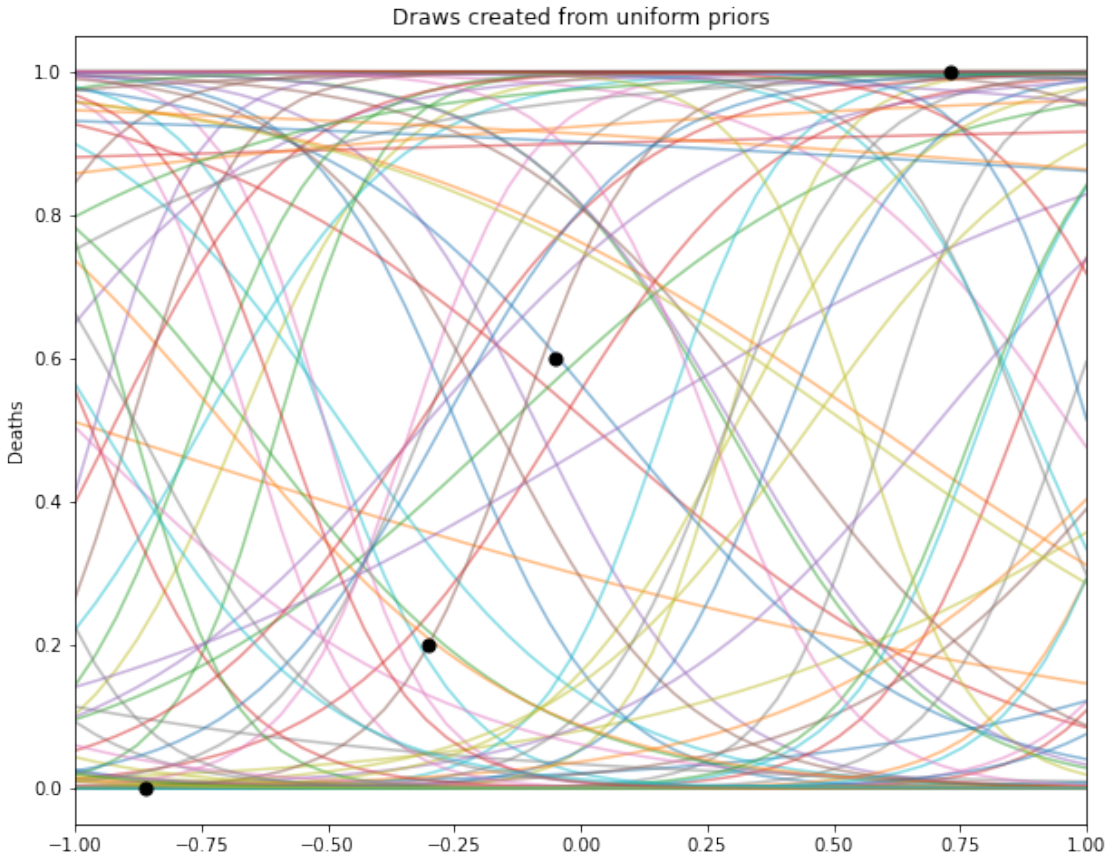
```
INFO:cmdstanpy:compiling stan program, exe file:
/mnt/c/ola/DataAnalytics/lab04/model_prior_uni1
INFO:cmdstanpy:compiler options: stanc_options=None, cpp_options=None
INFO:cmdstanpy:compiled model file:
/mnt/c/ola/DataAnalytics/lab04/model_prior_uni1
INFO:cmdstanpy:start chain 1
INFO:cmdstanpy:finish chain 1
```

[6]:
```
alpha_sim=sim.stan_variable('alpha')
beta_sim=sim.stan_variable('beta')
y_sim=sim.stan_variable('y_sim')
```

[7]:
```
fig, ax = plt.subplots(figsize=(8, 4))
ax.scatter(alpha_sim, beta_sim)
ax.set_xlabel(r'$\alpha$')
ax.set_ylabel(r'$\beta$',rotation=0)
ax.set_title('Noninformative uniform prior');
```



[8]:
```
fig, ax = plt.subplots(figsize=(10, 8))
xt = np.linspace(-1, 1)
fs = expit(alpha_sim[:, None] + beta_sim[:, None]*xt)
# plot 10 first samples
ax.plot(xt, fs[:100].T, alpha=0.5,zorder=0)
ax.scatter(x, y/n, 50, color='black')
ax.set_xlim((-1, 1))
ax.set_ylabel('Deaths')
ax.set_title('Draws created from uniform priors');
```



As we see on the plot above our uniform priors are quite messy and truly noninformative, we can try to upgrade it by learning more about the sigmoid shape we are looking for. In sigmoid function:

$$S(\alpha + \beta x) = \frac{1}{1 + e^{-(\alpha+\beta x)}}$$

We can notice that parameter β must be higher than 0 for sigmoid function to be rising chance of death with the rise of dosage. After doing some observation on the function we can also notice that alpha value will shift the function "rising edge" to right or to left. We can notice that it should be in range of (-2,2) for function to rise in accordance to rise in dosage.

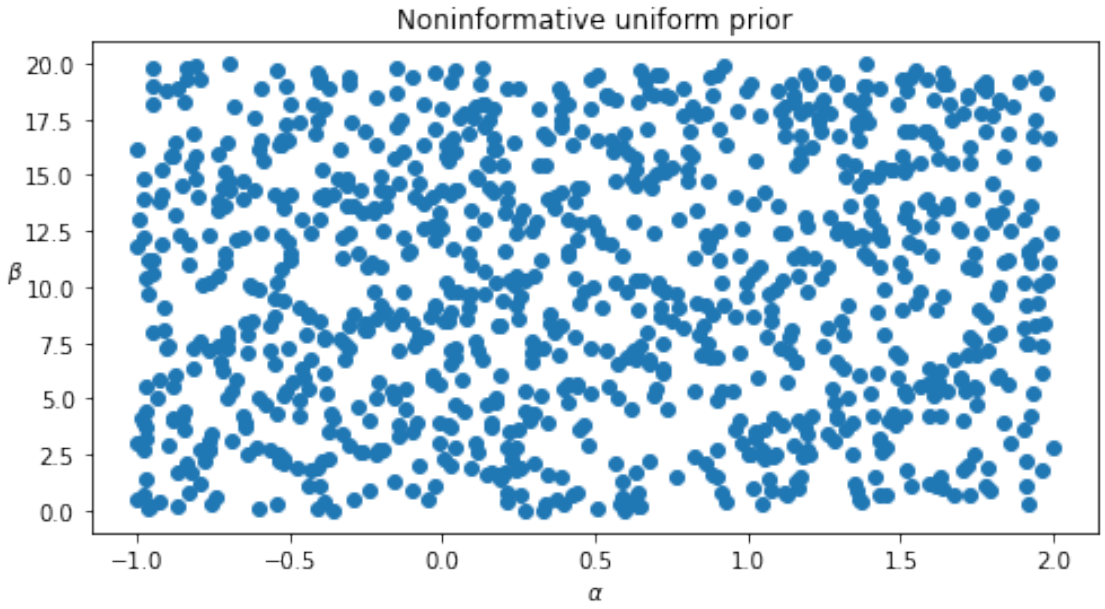
[11]:
```
model_prior_uni2=CmdStanModel(stan_file='model_prior_uni2.stan')
sim=model_prior_uni2.sample(data=dict(N=len(x),A=n,X=x),
                      fixed_param=True,
                      iter_warmup=0,
                      chains=1)
alpha_sim=sim.stan_variable('alpha')
beta_sim=sim.stan_variable('beta')
y_sim=sim.stan_variable('y_sim')
```

```
INFO:cmdstanpy:compiling stan program, exe file:
/mnt/c/ola/DataAnalytics/lab04/model_prior_uni2
```

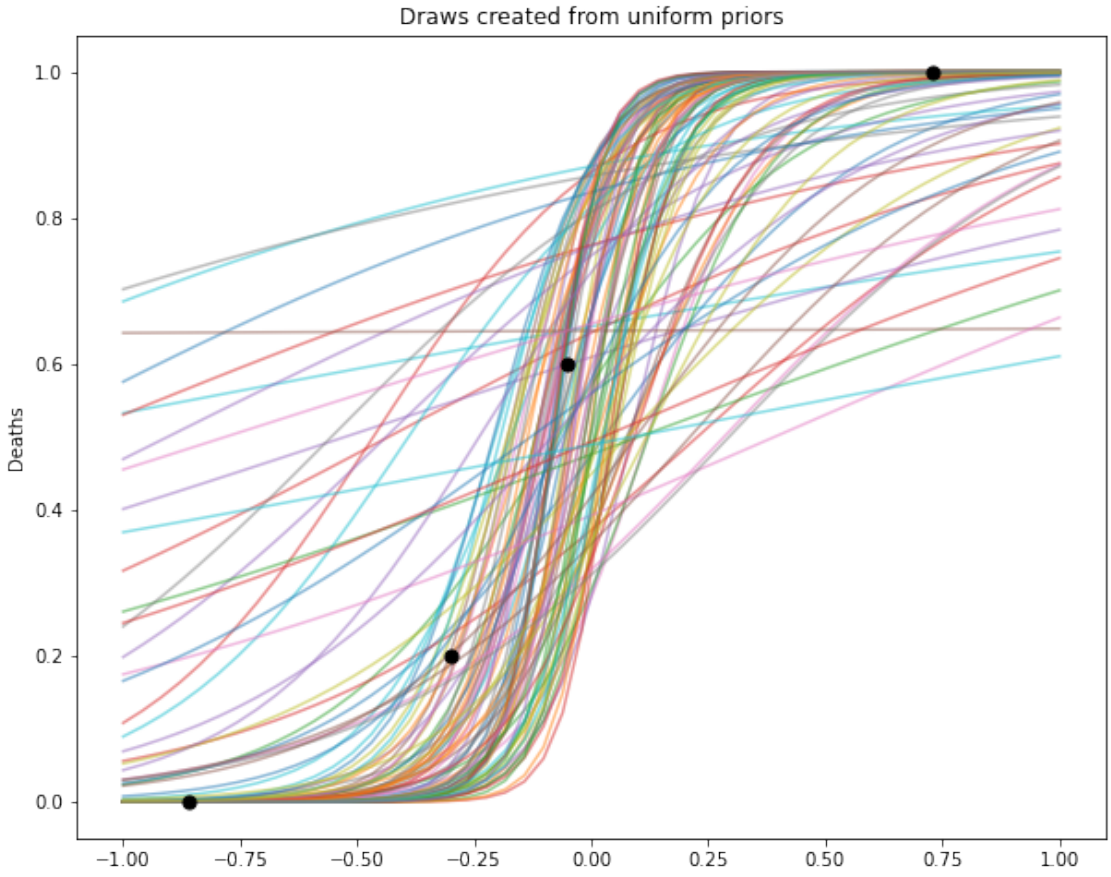
```
INFO:cmdstanpy:compiler options: stanc_options=None, cpp_options=None
INFO:cmdstanpy:compiled model file:
/mnt/c/ola/DataAnalytics/lab04/model_prior_uni2
INFO:cmdstanpy:start chain 1
INFO:cmdstanpy:finish chain 1
```

[12]:
```
fig, ax = plt.subplots(figsize=(8, 4))
ax.scatter(alpha_sim, beta_sim)
ax.set_xlabel(r'$\alpha$')
ax.set_ylabel(r'$\beta$',rotation=0)
ax.set_title('Noninformative uniform prior')
```

[12]:
```
Text(0.5, 1.0, 'Noninformative uniform prior')
```



[13]:
```
fig, ax = plt.subplots(figsize=(10, 8))
xt = np.linspace(-1, 1)
fs = expit(alpha_sim[:, None] + beta_sim[:, None]*xt)
ax.plot(xt, fs[:100].T, alpha=0.5,zorder=0)
ax.scatter(x, y/n, 50, color='black')
ax.set_ylabel('Deaths')
ax.set_title('Draws created from uniform priors');
```



We can see it's a lot better than previous version. But as we know from Stan Wiki for logit function noninformative priors we should use Students-t distribution. Bearing our previous attempts in priors in mind, we can write our model as:

$$y_i|\theta_i \sim \text{Bin}(n_i, \theta_i)$$

$$\text{logit}(\theta_i) = \alpha + \beta x_i$$

$$\alpha(\theta_i) = t_v(-2, 2)$$

$$\beta(\theta_i) = t_v(0, 20)$$

For $3 < v < 7$, we set v as 4.

[14]:
```
with open('model_prior_students.stan', 'r') as file:
    print(file.read())
```

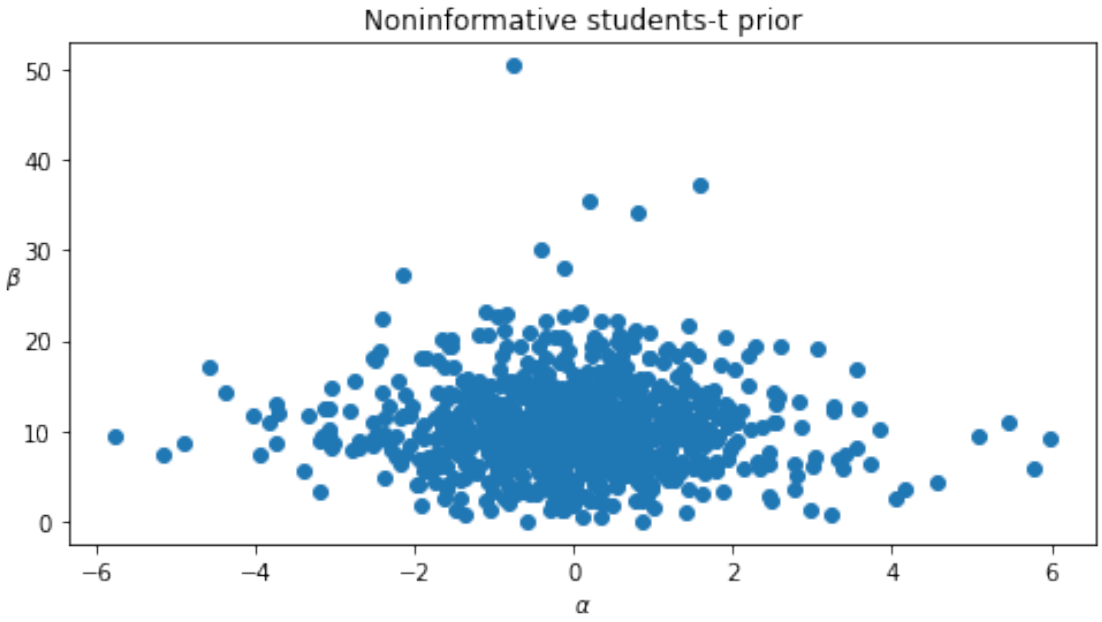
```
data {
  int N;
  int A[N];
  real X[N];
}
```

```
generated quantities {
  real alpha=student_t_rng(4,0,1);
  real<lower = 0> beta=student_t_rng(4,10,4);

  int y_sim[N];
  for (k in 1:N) {
    y_sim[k] = binomial_rng(A[k],inv_logit(alpha+beta*X[k]));
  }
}
```

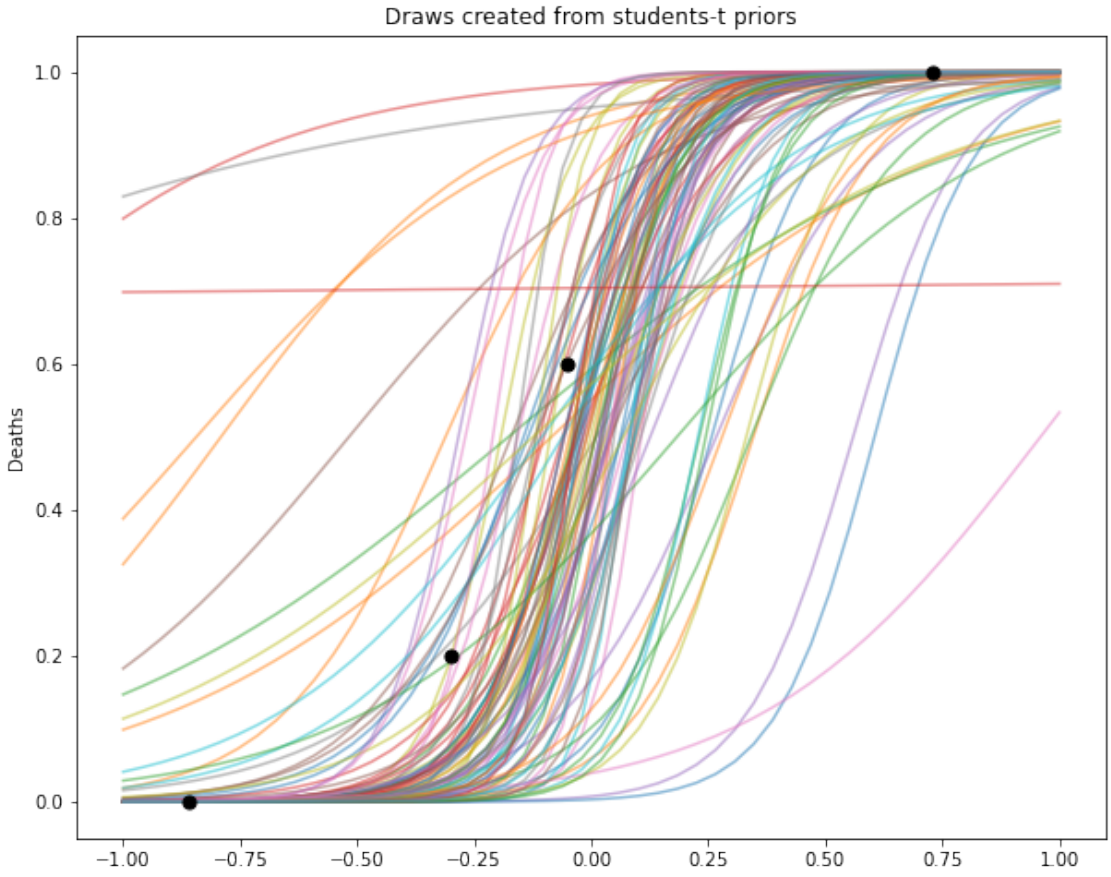
[15]:
```
model_prior_students=CmdStanModel(stan_file='model_prior_students.stan')
sim_prio=model_prior_students.sample(data=dict(N=len(x),A=n,X=x),
                     fixed_param=True,
                     iter_warmup=0,
                     chains=1)
alpha_prior=sim_prio.stan_variable('alpha')
beta_prior=sim_prio.stan_variable('beta')
y_prior=sim_prio.stan_variable('y_sim')
```

```
INFO:cmdstanpy:compiling stan program, exe file:
/mnt/c/ola/DataAnalytics/lab04/model_prior_students
INFO:cmdstanpy:compiler options: stanc_options=None, cpp_options=None
INFO:cmdstanpy:compiled model file:
/mnt/c/ola/DataAnalytics/lab04/model_prior_students
INFO:cmdstanpy:start chain 1
INFO:cmdstanpy:finish chain 1
```

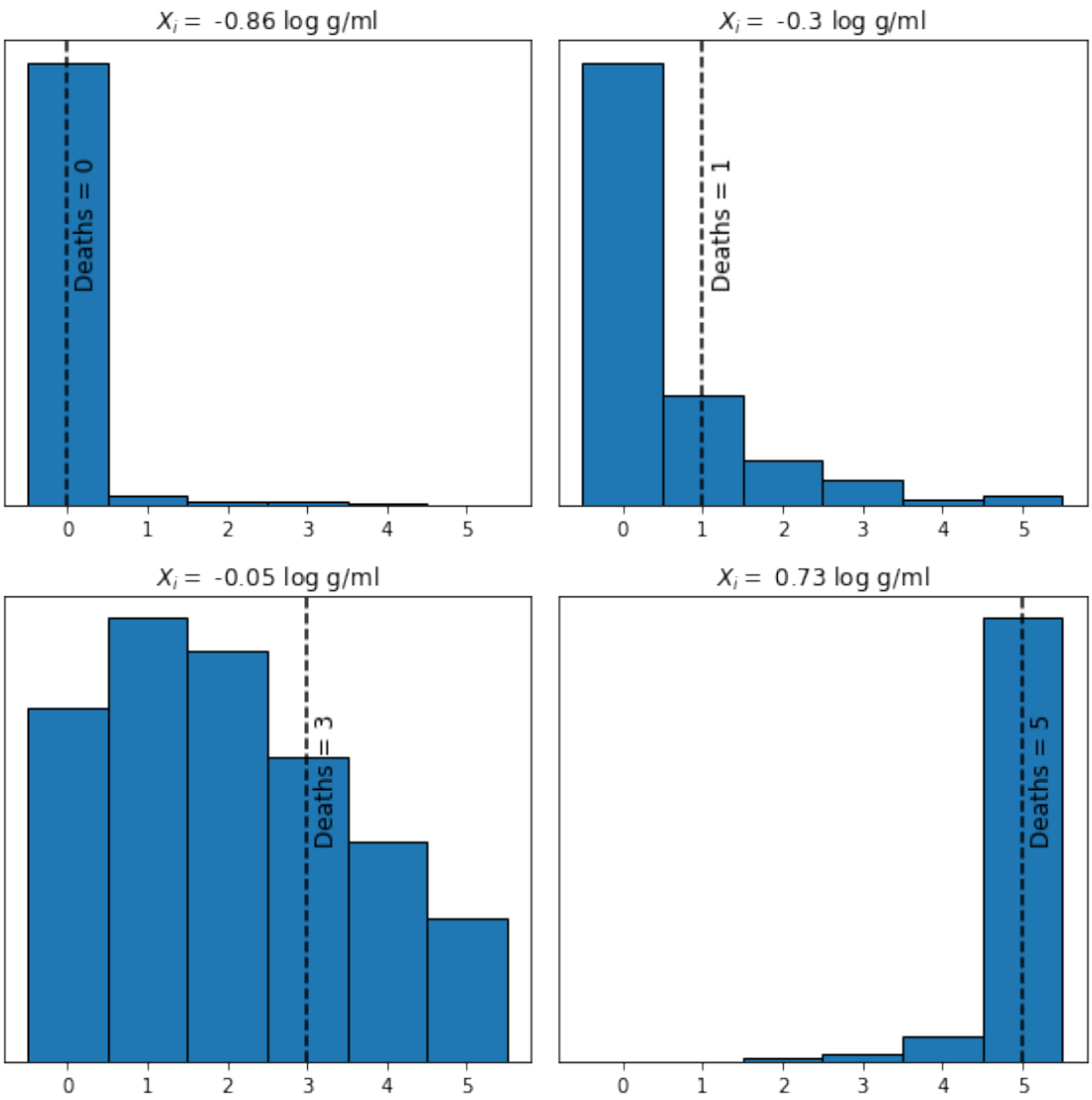
[16]:
```
fig, ax = plt.subplots(figsize=(8, 4))
ax.scatter(alpha_prior, beta_prior)
ax.set_xlabel(r'$\alpha$')
ax.set_ylabel(r'$\beta$',rotation=0)
ax.set_title('Noninformative students-t prior');
```



[17]:
```
fig, ax = plt.subplots(figsize=(10, 8))
xt = np.linspace(-1, 1)
fs = expit(alpha_prior[:, None] + beta_prior[:, None]*xt)
ax.plot(xt, fs[:100].T, alpha=0.5,zorder=0)
ax.scatter(x, y/n, 50, color='black')
ax.set_ylabel('Deaths')
ax.set_title('Draws created from students-t priors');
```



[18]:
```
fig, axes = plt.subplots(2, 2, figsize=(8, 8))
ax=axes.ravel()
for k in range(len(ax)):
    ax[k].hist(y_prior[:,k], bins=np.linspace(-0.5,5.5,7), density=True,␣
 ↪edgecolor ='black')
    ax[k].set_title('$X_i =$ ' + str(x[k]) + ' log g/ml')
    ax[k].axvline(y[k],linestyle='--',color='black')
    ax[k].text(y[k]+0.1,(ax[k].get_ylim()[1])/2.1, 'Deaths = '+str(y[k])␣
 ↪,rotation=90, fontsize=12)
    ax[k].set_yticks([])

fig.tight_layout()
```



After defining our priors we can create model in Stan language and calculate our posterior distribution.

[19]:
```
with open('model_post.stan', 'r') as file:
    print(file.read())
```

```
data {
  int N;
  int A[N];
  real X[N];
  int y[N];

  int N_C;
  real Xc[N_C];
```

```
}
parameters {
  real alpha;
  real<lower = 0> beta;
}
transformed parameters {
  vector[N] theta;
  for (k in 1:N) {
    theta[k] = inv_logit(alpha+beta*X[k]);
  }
}
model {
  alpha ~ student_t(4,0,1);
  beta ~ student_t(4,10,4);
  for (k in 1:N) {
    y[k] ~ binomial(A[k],theta[k]);
  }
}
generated quantities {
  int y_sim[N];
  int y_gen[N_C];
  for (k in 1:N) {
    y_sim[k] = binomial_rng(A[k],theta[k]);
  }
  for (k in 1:N_C) {
    y_gen[k] = bernoulli_rng(inv_logit(alpha+beta*Xc[k]));
  }
}
```

[20]:
```
model_post = CmdStanModel(stan_file='model_post.stan')
```

```
INFO:cmdstanpy:compiling stan program, exe file:
/mnt/c/ola/DataAnalytics/lab04/model_post
INFO:cmdstanpy:compiler options: stanc_options=None, cpp_options=None
INFO:cmdstanpy:compiled model file: /mnt/c/ola/DataAnalytics/lab04/model_post
```

[21]:
```
toget = [x/100.0 for x in range(-100, 100, 1)]
fit_post = model_post.sample(data=dict(N = len(x),A = n, X=x, y=y, N_C =␣
 ↪len(toget), Xc = toget))
fit_post.diagnose()
```

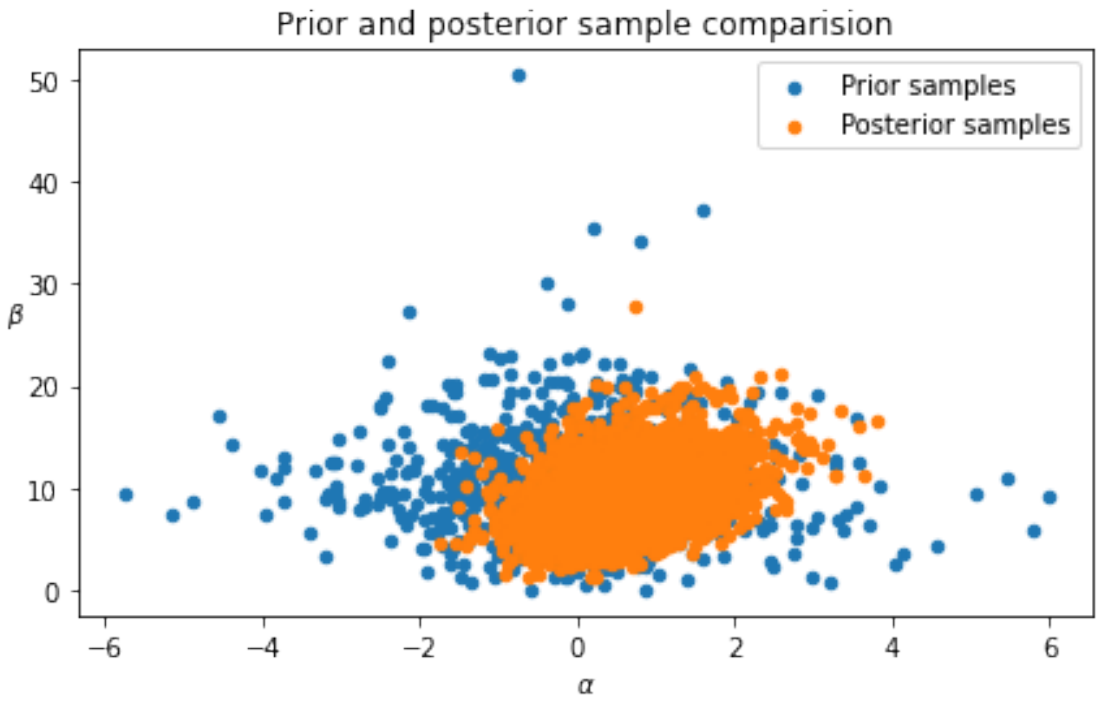
```
INFO:cmdstanpy:start chain 1
INFO:cmdstanpy:start chain 2
INFO:cmdstanpy:start chain 3
INFO:cmdstanpy:start chain 4
INFO:cmdstanpy:finish chain 4
INFO:cmdstanpy:finish chain 2
INFO:cmdstanpy:finish chain 3
```

```
INFO:cmdstanpy:finish chain 1
INFO:cmdstanpy:Processing csv files:
/tmp/tmphjn_h332/model_post-202106122207-1-afidhoef.csv,
/tmp/tmphjn_h332/model_post-202106122207-2-gcun5ncm.csv,
/tmp/tmphjn_h332/model_post-202106122207-3-_dwww0cz.csv,
/tmp/tmphjn_h332/model_post-202106122207-4-wpay7wn6.csv

Checking sampler transitions treedepth.
Treedepth satisfactory for all transitions.

Checking sampler transitions for divergences.
No divergent transitions found.

Checking E-BFMI - sampler transitions HMC potential energy.
E-BFMI satisfactory.

Effective sample size satisfactory.

Split R-hat values satisfactory all parameters.

Processing complete, no problems detected.
```

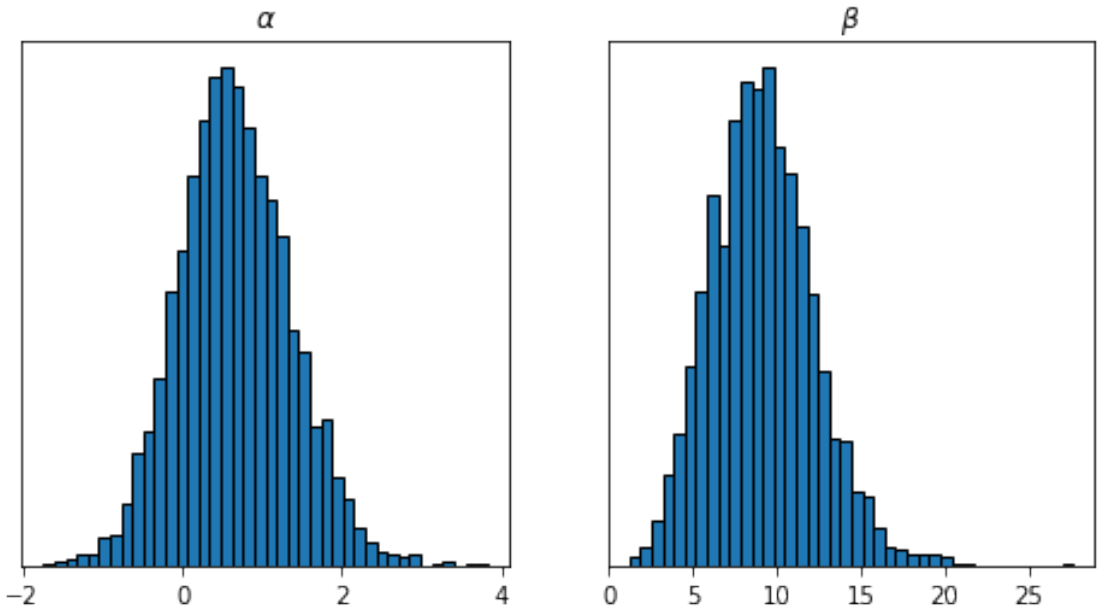
[21]:
```
'Processing csv files: /tmp/tmphjn_h332/model_post-202106122207-1-afidhoef.csv,
/tmp/tmphjn_h332/model_post-202106122207-2-gcun5ncm.csv,
/tmp/tmphjn_h332/model_post-202106122207-3-_dwww0cz.csv,
/tmp/tmphjn_h332/model_post-202106122207-4-wpay7wn6.csv\n\nChecking sampler
transitions treedepth.\nTreedepth satisfactory for all transitions.\n\nChecking
sampler transitions for divergences.\nNo divergent transitions
found.\n\nChecking E-BFMI - sampler transitions HMC potential energy.\nE-BFMI
satisfactory.\n\nEffective sample size satisfactory.\n\nSplit R-hat values
satisfactory all parameters.\n\nProcessing complete, no problems detected.'
```

[22]:
```
alpha=fit_post.stan_variable('alpha')
beta=fit_post.stan_variable('beta')
```

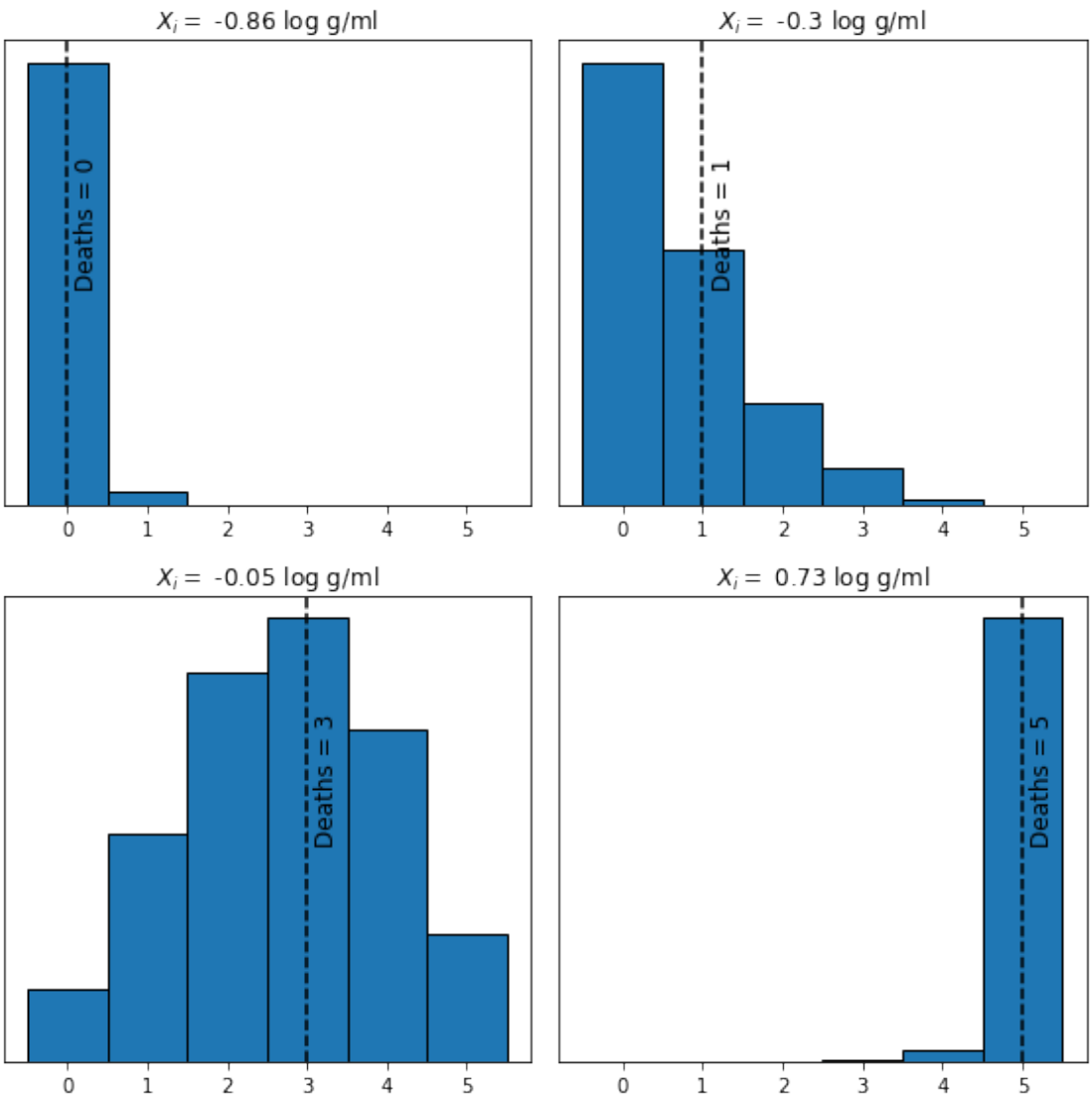
[23]:
```
fig, axes = plt.subplots(figsize=(7, 4))
axes.scatter(alpha_prior, beta_prior, 20)
axes.scatter(alpha, beta, 20)
axes.set_xlabel(r'$\alpha$')
axes.set_ylabel(r'$\beta$',rotation=0)
axes.set_title('Prior and posterior sample comparision')
axes.legend(['Prior samples','Posterior samples']);
```



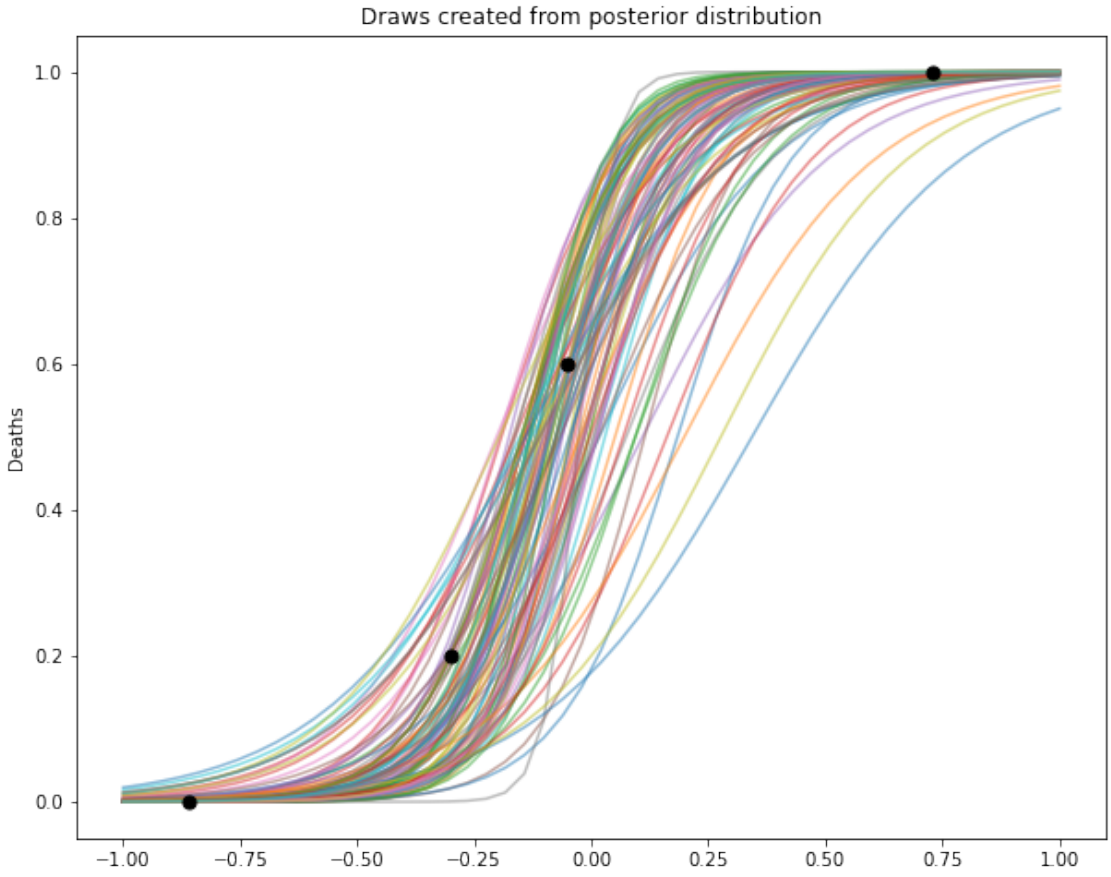
[24]:
```
fig, ax = plt.subplots(1, 2, figsize=(8, 4))
ax[0].hist(alpha,bins=40, density=True, edgecolor='black')
ax[0].set_title(r'$\alpha$')
ax[0].set_yticks(());

ax[1].hist(beta,bins=40, density=True, edgecolor='black')
ax[1].set_title(r'$\beta$')
ax[1].set_yticks(());
```



[25]:
```
y_sim=fit_post.stan_variable('y_sim')
fig, axes = plt.subplots(2, 2, figsize=(8, 8))
ax=axes.ravel()
for k in range(len(ax)):
    ax[k].hist(y_sim[:,k], bins=np.linspace(-0.5,5.5,7), density=True,␣
 ↪edgecolor ='black')
    ax[k].set_title('$X_i =$ ' + str(x[k]) + ' log g/ml')
    ax[k].axvline(y[k],linestyle='--',color='black')
    ax[k].text(y[k]+0.1,(ax[k].get_ylim()[1])/2.1, 'Deaths = '+str(y[k])␣
 ↪,rotation=90, fontsize=12)
    ax[k].set_yticks([])

fig.tight_layout()
```



[26]:
```
fig, ax = plt.subplots(figsize=(10, 8))
xt = np.linspace(-1, 1)
fs = expit(alpha[:, None] + beta[:, None]*xt)
ax.plot(xt, fs[:100].T, alpha=0.5,zorder=0)
ax.scatter(x, y/n, 50, color='black')
ax.set_ylabel('Deaths')
ax.set_title('Draws created from posterior distribution');
```



Now we have to present posterior distribution of the LD50.

$$\text{LD50 :E}\left(\frac{y_i}{n_i}\right) = \text{logit}^{-1}(\alpha + \beta x_i) = 0.5$$

$$\alpha + \beta x_i = logit(0.5)$$

knowing that $logit(0.5) = 0$:

$$\alpha + \beta x_i = 0$$

$$x_i = \frac{-\alpha}{\beta}$$

Now to get LD50 we need to compute $\frac{-\alpha}{\beta}$ for draws given by our posterior.

[27]:
```
LD50 = -alpha/beta
fig, ax = plt.subplots(figsize=(8, 4))
ax.hist(LD50, bins=40,color = '#FF7F50' ,edgecolor='#00008B')
ax.set_xlabel("LD50")
ax.set_yticks([])
ax.set_title("Histogram of LD50");
ax.axvline(np.median(LD50), linestyle='--',color='black')
meanLD = np.median(LD50)
```

```
ax.text(meanLD+0.02,(ax.get_ylim()[1])/1.2, 'Median = '+str(np.
 ↪around(meanLD,decimals=3)) ,rotation=0, fontsize=12);
percentiles = np.percentile(LD50,[5,95],axis=0)
ax.axvline(percentiles[0], color='black');
ax.text(percentiles[0]-0.19,(ax.get_ylim()[1])/2, '5% = '+str(np.
 ↪around(percentiles[0],decimals=3)) ,rotation=0, fontsize=12);
ax.axvline(percentiles[1], color='black');
ax.text(percentiles[1]+0.01,(ax.get_ylim()[1])/2, '95% = '+str(np.
 ↪around(percentiles[1],decimals=3)) ,rotation=0, fontsize=12);
```