lab03

June 12, 2021

1 Exercise 1

We will analyze the effect of the non-informative prior distribution on Bayesian inference. You aim to compare two models, one with a uniform prior, second with the beta distribution. Please, generate 200 draws of a biased coin with a probability of getting a tail equal to 0.7 and compare inference results as a function of flips number. Plot and interpret the results.

[1]:
```
from numpy.random import binomial
from cmdstanpy import CmdStanModel
import pandas as pd
import numpy as np
N = 10
draws = binomial(1, 0.7, size=N)
data={'N': N,
      'y': draws.tolist()}
```

[2]:
```
# build stan models
uniform_model = CmdStanModel(stan_file='stan_code_uniform.stan')
beta_model = CmdStanModel(stan_file='stan_code_beta.stan')
normal_model = CmdStanModel(stan_file='stan_code_gauss.stan')
```

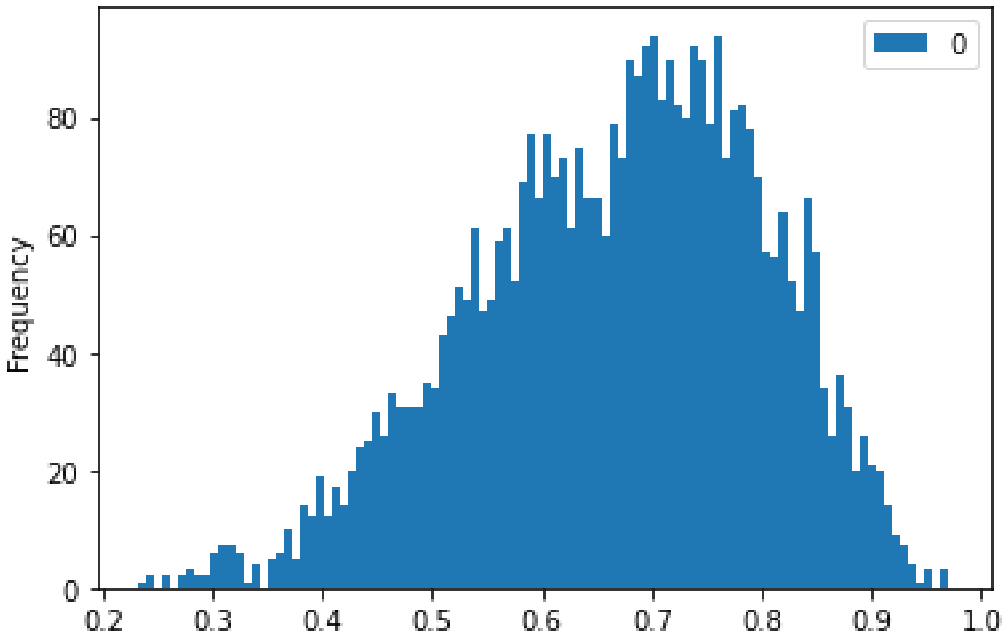
```
INFO:cmdstanpy:compiling stan program, exe file:
/mnt/c/ola/DataAnalytics/lab03/stan_code_uniform
INFO:cmdstanpy:compiler options: stanc_options=None, cpp_options=None
INFO:cmdstanpy:compiled model file:
/mnt/c/ola/DataAnalytics/lab03/stan_code_uniform
INFO:cmdstanpy:compiling stan program, exe file:
/mnt/c/ola/DataAnalytics/lab03/stan_code_beta
INFO:cmdstanpy:compiler options: stanc_options=None, cpp_options=None
INFO:cmdstanpy:compiled model file:
/mnt/c/ola/DataAnalytics/lab03/stan_code_beta
INFO:cmdstanpy:compiling stan program, exe file:
/mnt/c/ola/DataAnalytics/lab03/stan_code_gauss
INFO:cmdstanpy:compiler options: stanc_options=None, cpp_options=None
INFO:cmdstanpy:compiled model file:
/mnt/c/ola/DataAnalytics/lab03/stan_code_gauss
```

[3]:
```python
# Uniform
with open('stan_code_uniform.stan', 'r') as code:
    print(code.read())

sample_uni = uniform_model.sample(data=data)

post_uni = sample_uni.stan_variable(name='p')
pd.DataFrame(data=post_uni).plot(kind='hist', bins=100);
print(np.mean(post_uni))
```

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

```
data{
    int<lower=1> N;
    int<lower=0,upper=1> y[N];
}
parameters{
    real<lower=0,upper=1> p;
}
model{
    p ~ uniform(0, 1);
    y ~ binomial(1, p);
}
```

```
0.6688861200000001
```



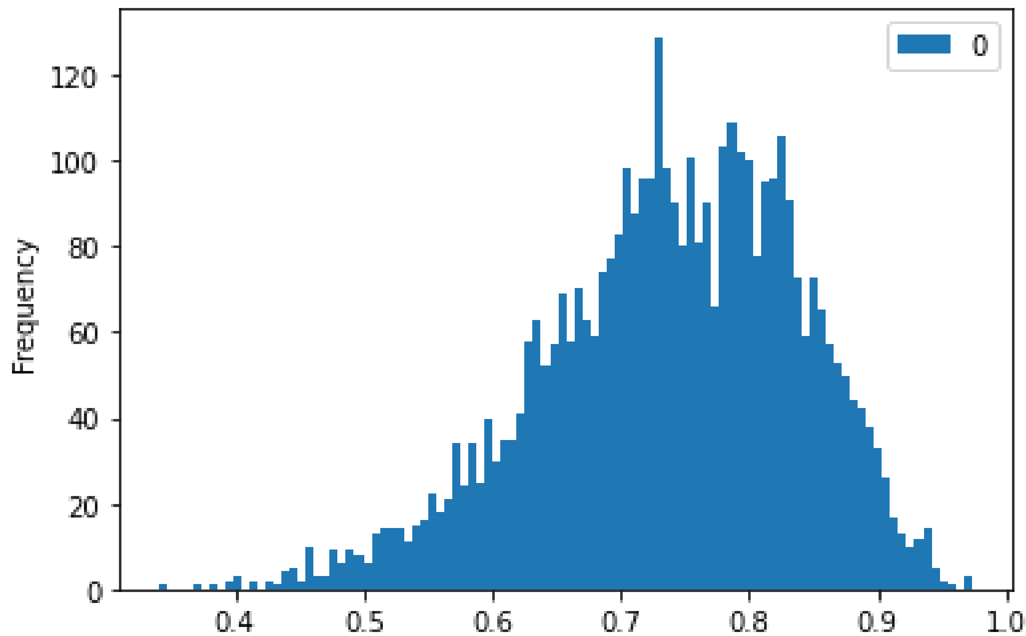
[3]:
```
with open('stan_code_beta.stan', 'r') as code:
    print(code.read())

sample_beta = beta_model.sample(data=data)

post_beta = sample_beta.stan_variable(name='p')
pd.DataFrame(data=post_beta).plot(kind='hist', bins=100);
print(np.mean(post_beta))
```

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

```
data{
    int<lower=1> N;
    int<lower=0,upper=1> y[N];
}
parameters{
    real<lower=0,upper=1> p;
}
model{
    p ~ beta(6, 3);
```

```
    y ~ binomial(1, p);
}
```

```
0.738261434
```



2 Exercise 2

We consider the number of fatal accidents and deaths on scheduled airline flights per year over a ten-year period Source: Gelman et al. 2014 Reproduced from Statistical Abstract of the United States. Our goal is to create a model predicting such number in 1986.

[4]:
```
from cmdstanpy import CmdStanModel
import pandas as pd
import arviz as az
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
```

Using the data from the following table create a model predicting the number of passenger deaths. Use Poisson distribution assuming that the accident rate is constant for all years and not depending on anything.

[5]:
```
dts=[24,734,25,516,31,754,31,877,22,814,21,362,26,764,20,809,16,223,22,1066]
c1=dts[::2]
c2=dts[1::2]
```

```
Airline_data=pd.DataFrame({'Year':
 ↪[1976,1977,1978,1979,1980,1981,1982,1983,1984,1985],
'Fatal accidents':c1,
'Passenger deaths':c2,
'Death rate':[0.19,0.12,0.15,0.16,0.14,0.06,0.13,0.13,0.03,0.15]}).
 ↪set_index('Year')
Airline_data['Miles flown [100 mln miles]']=np.round(Airline_data['Passenger␣
 ↪deaths']/Airline_data['Death rate'])
Airline_data
```

[5]:

```
      Fatal accidents  Passenger deaths  Death rate  \
Year
1976               24               734        0.19
1977               25               516        0.12
1978               31               754        0.15
1979               31               877        0.16
1980               22               814        0.14
1981               21               362        0.06
1982               26               764        0.13
1983               20               809        0.13
1984               16               223        0.03
1985               22              1066        0.15

      Miles flown [100 mln miles]
Year
1976                       3863.0
1977                       4300.0
1978                       5027.0
1979                       5481.0
1980                       5814.0
1981                       6033.0
1982                       5877.0
1983                       6223.0
1984                       7433.0
1985                       7107.0
```

3 1. Deaths independent from other variables

Assuming passenger deaths distribution as Poisson distribution

$$y_i \sim \text{Poisson}(\lambda)$$

with mass function given as:

$$p(y|\lambda) = \frac{e^{-\lambda}\lambda^y}{y!}$$

Asumming i as number of years and λ as expected number of accidents in a year we are going to estimate parameter λ.

Our model looks like:

$$y_i \sim \text{Poisson}(\lambda)$$

$$\lambda \sim \text{Gamma}(\alpha, \beta)$$

Assuming noninformative prior distribution for Gamma distribution:

$$Gamma(\alpha, \beta) = Gamma(0, 0)$$

we can calculate posterior Gamma distribution using equation:

$$\lambda|y \sim \text{Gamma}\left(\alpha + \sum_{\mathrm{i1}}^{\mathrm{n}} \mathrm{y_i}, \beta + \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{x_i}\right)$$

Following the given formula we can get posterior values for α and β:

$$\alpha = 6919, \beta = 10$$

Stan code:

3.1 Prior predictive distribution

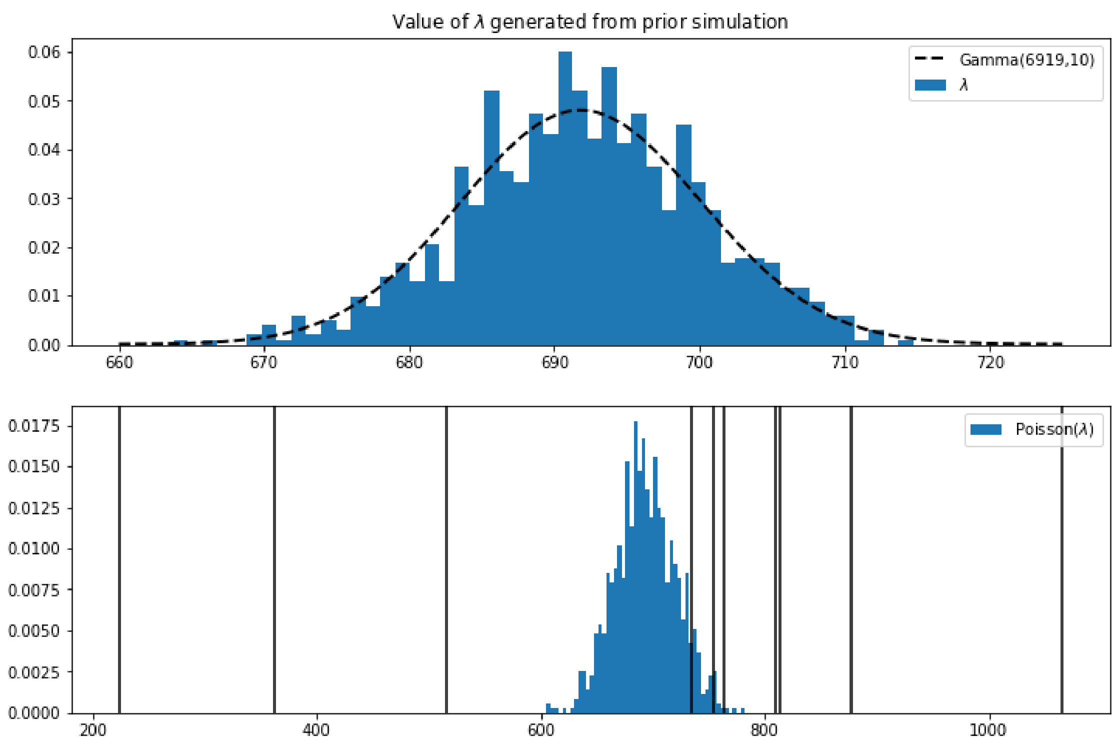
[6]:
```
with open('plane_prior.stan', 'r') as code:
    print(code.read())

model_prior = CmdStanModel(stan_file='plane_prior.stan')
data_prior = model_prior.sample(data={'N':1}, fixed_param=True,␣
 ↪iter_sampling=1000, iter_warmup=0, chains=1)
lbd=data_prior.stan_variable('lambda')
y_sim=data_prior.stan_variable('y')
```

```
INFO:cmdstanpy:compiling stan program, exe file:
/mnt/c/ola/DataAnalytics/lab03/plane_prior
INFO:cmdstanpy:compiler options: stanc_options=None, cpp_options=None
```

```
data {
  int N;
}
generated quantities {
  real lambda = gamma_rng(6919,10);
  vector[N] y;
  for (n in 1:N) {
    y[n] = poisson_rng(lambda);
  }
}
```

```
INFO:cmdstanpy:compiled model file: /mnt/c/ola/DataAnalytics/lab03/plane_prior
INFO:cmdstanpy:start chain 1
INFO:cmdstanpy:finish chain 1
```

[7]:
```python
fig, ax = plt.subplots(2,1, figsize=(12, 8))
ax[0].hist(lbd, bins=50, density=True)
x=np.linspace(660,725,1000)
ax[0].set_title(r'Value of $\lambda$ generated from prior simulation')
ax[0].plot(x, stats.gamma.pdf(x, 6919, scale=1/10), color='black',␣
 ↪linestyle='dashed', linewidth=2)
ax[0].legend(['Gamma(6919,10)','$\lambda$'])

ax[1].hist(y_sim.flatten(), bins=50, density=True, zorder=1)
ax[1].legend(['Poisson($\lambda$)'])
for i in range(0,len(c2)):
    ax[1].axvline(x = c2[i], color='black')

plt.show()
```



[8]:
```python
with open('stan_accidents.stan', 'r') as code:
    print(code.read())
```

```
data{
    int N;
    int y[N];
}
parameters{
    real<lower=0> lambda;
```

```
}
model{
    lambda ~ gamma(6919,10);

    for (n in 1:N) {
      y[n] ~ poisson(lambda);
    }
}
generated quantities {
  vector[N] y_new;
  for (n in 1:N)
    y_new[n] = poisson_rng(lambda);
}
```

[9]:
```
model = CmdStanModel(stan_file='stan_accidents.stan')
```

```
INFO:cmdstanpy:compiling stan program, exe file:
/mnt/c/ola/DataAnalytics/lab03/stan_accidents
INFO:cmdstanpy:compiler options: stanc_options=None, cpp_options=None
INFO:cmdstanpy:compiled model file:
/mnt/c/ola/DataAnalytics/lab03/stan_accidents
```

[10]:
```
data = dict(N = len(c2),
            y = c2)
fit = model.sample(data=data)

lbda = fit.stan_variable('lambda')
y_new = fit.stan_variable('y_new')
```

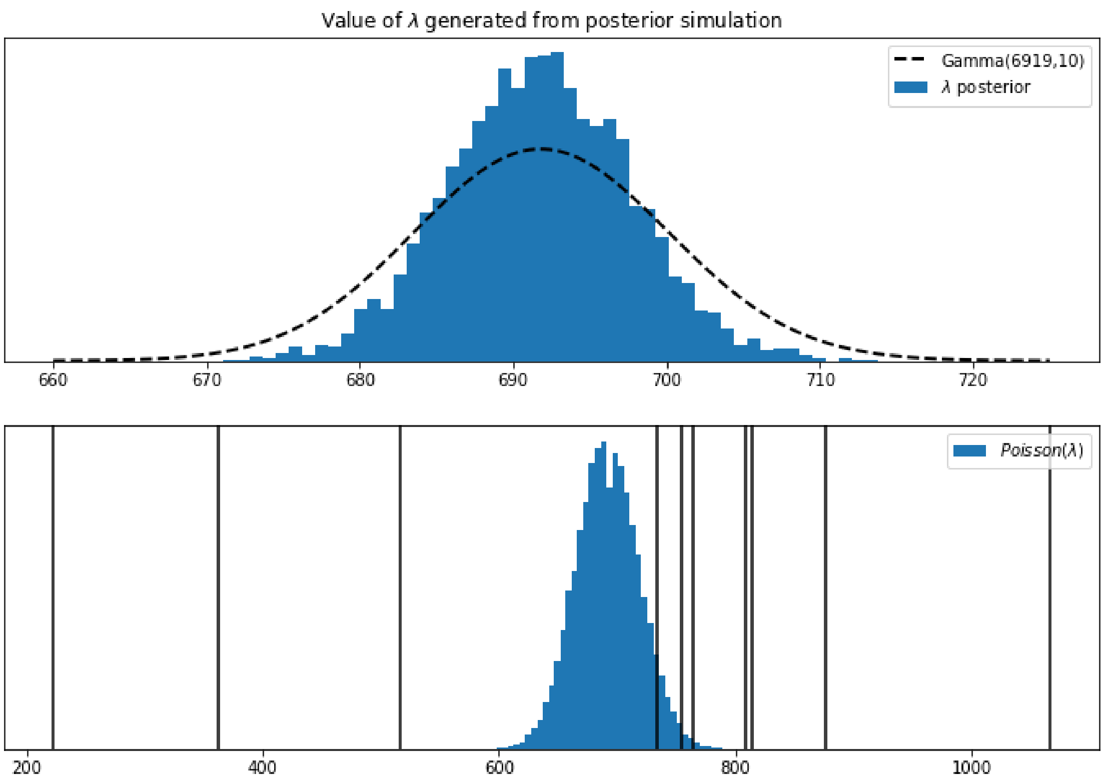
```
INFO:cmdstanpy:start chain 1
INFO:cmdstanpy:start chain 2
INFO:cmdstanpy:start chain 3
INFO:cmdstanpy:start chain 4
INFO:cmdstanpy:finish chain 3
INFO:cmdstanpy:finish chain 1
INFO:cmdstanpy:finish chain 4
INFO:cmdstanpy:finish chain 2
```

[11]:
```
lambda_mean = np.mean(lbda)
hdi_89 = az.hdi(lbda, 0.89)
print('mean(lambda) : ', lambda_mean)
print('HDI 89%: [',*['{:4.2f}'.format(k) for k in hdi_89],']')
```

```
mean(lambda) :  691.7246415000002
HDI 89%: [ 682.52 701.10 ]
```

[12]:
```
fig, ax = plt.subplots(2,1, figsize=(12, 8))
ax[0].hist(lbda, bins=50, density=True)
```

```python
x=np.linspace(660,725,1000)
ax[0].set_title(r'Value of $\lambda$ generated from posterior simulation')
ax[0].plot(x, stats.gamma.pdf(x, 6919, scale=1/10), color='black',␣
 ↪linestyle='dashed', linewidth=2)
ax[0].legend(['Gamma(6919,10)','$\lambda$ posterior'])
ax[0].set_yticks([])

ax[1].hist(y_new.flatten(), bins=50, density=True, zorder=1)
ax[1].legend(['$Poisson(\lambda)$'])
for i in range(0,len(c2)):
    ax[1].axvline(x = c2[i], color='black')
ax[1].set_yticks([])

plt.show()
```



4 Setting subjective prior

We can assume prior for passenger deaths by analyzing other informations.
Knowing the time period for our data we can:
1.Using information from "Statistical Abstract of the United States: 1990" we can see that the most popular aircraft models in operation in this time period where Boings 727 and 737. Both of this models have carrying capacity of around 100 people. We can assume that mostly used aircraft

worldwide had this kind of capacity.
2.We can assume failure rate for Boeing 737 (accident rate or approx 3 per year or one every 2.5 million flight hours [http://www.b737.org.uk/accident_reports.htm]).
3.Knowing average fly lenght in this time period was 1,5h [https://www.researchgate.net/figure/The-average-flight-length-hours-The-graph-above-illustrates-that-the-average-flight_fig3_2178852]
4.Checking departure statistics we can see that departure rate in this time period was around 10 million. [https://data.worldbank.org/indicator/IS.AIR.DPRT]
5.Knowing the number of departures and failure rate we can calculate possible number of fatal accidents. $(10 * 1.5)/2.5 = 6$.
6.Now we must add possible crashes caused by human error, we assume the number is 50% of all crashes [https://www.lawfirms.com/resources/personal-injury/aviation/causes-aviation-accidents.htm].
7.Knowing that most of the plane crashes ends in no survivors, let's assume 80% victim of all crashes dies.
8.Now we can set our prior mean for passenger death as $((6 + 6) * 100) * 0.8 = 960$, because we are uncertain about our calculations let's set variance = 100.

[13]:
```python
with open('air_prior.stan', 'r') as code:
    print(code.read())

model_prior = CmdStanModel(stan_file='air_prior.stan')
data_prior = model_prior.sample(data={'N':1}, fixed_param=True,␣
 ↪iter_sampling=1000, iter_warmup=0, chains=1)
lbd=data_prior.stan_variable('lambda')
y_sim=data_prior.stan_variable('y')
```

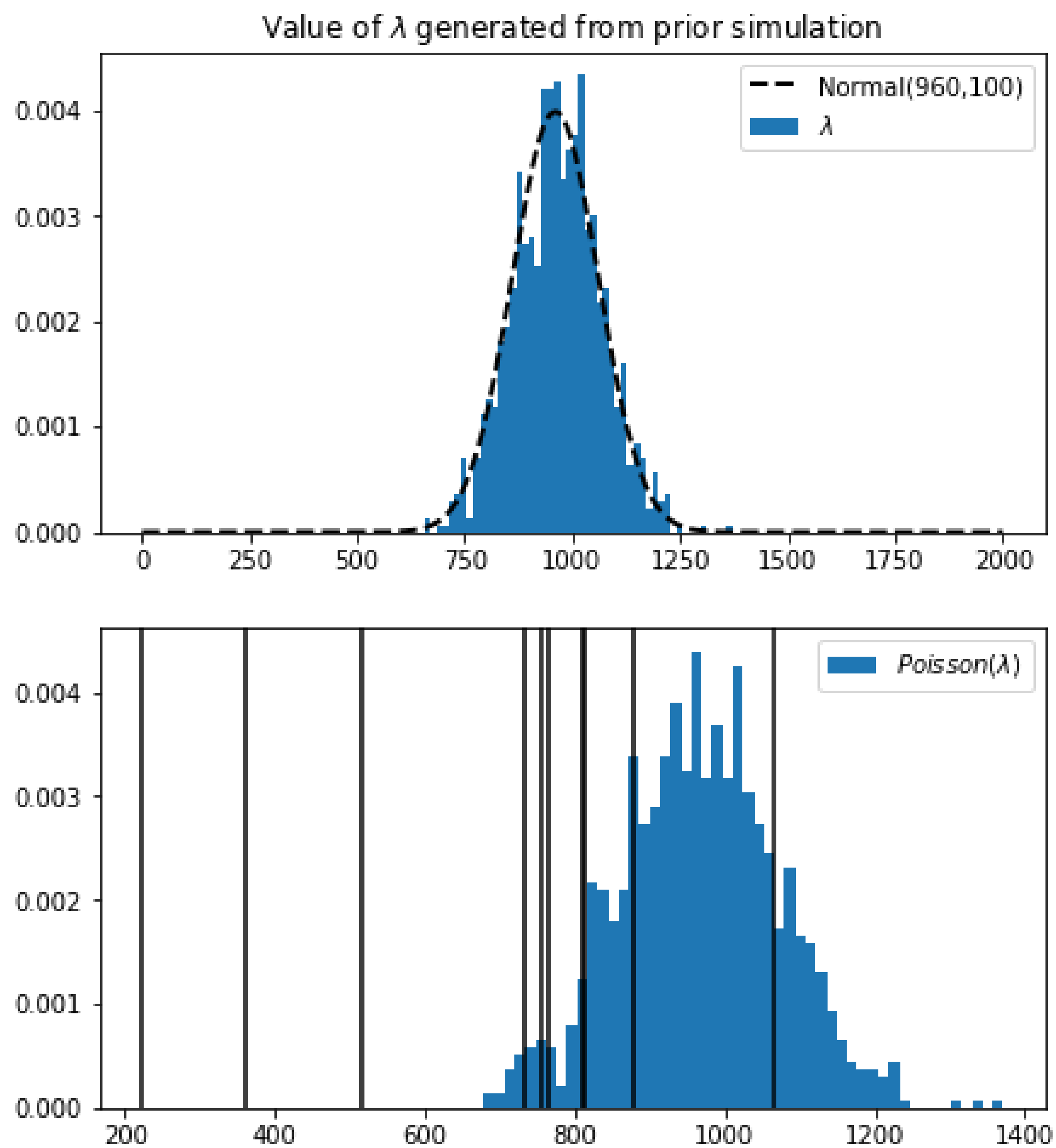
```
INFO:cmdstanpy:compiling stan program, exe file:
/mnt/c/ola/DataAnalytics/lab03/air_prior
INFO:cmdstanpy:compiler options: stanc_options=None, cpp_options=None
```

```
data {
  int N;
}
generated quantities {
  real lambda = normal_rng(960,100);
  vector[N] y;
  for (n in 1:N) {
    y[n] = poisson_rng(lambda);
  }
}
```

```
INFO:cmdstanpy:compiled model file: /mnt/c/ola/DataAnalytics/lab03/air_prior
INFO:cmdstanpy:start chain 1
INFO:cmdstanpy:finish chain 1
```

[14]:
```python
fig, ax = plt.subplots(2,1, figsize=(7, 8))
ax[0].hist(lbd, bins=50, density=True)
```

```
x=np.linspace(0,2000,4000)
ax[0].set_title(r'Value of $\lambda$ generated from prior simulation')
ax[0].plot(x, stats.norm.pdf(x, 960, 100), color='black', linestyle='dashed',␣
↪linewidth=2)
ax[0].legend(['Normal(960,100)','$\lambda$'])

ax[1].hist(y_sim.flatten(), bins=50, density=True, zorder=1)
ax[1].legend(['$Poisson(\lambda)$'])
for i in range(0,len(c2)):
    ax[1].axvline(x = c2[i], color='black')

plt.show()
```



[15]:
```
with open('sp_air.stan', 'r') as code:
    print(code.read())

model = CmdStanModel(stan_file='sp_air.stan')
data = dict(N = len(c2),
            y = c2)
fit = model.sample(data=data)

lbda = fit.stan_variable('lambda')
y_new = fit.stan_variable('y_new')
```

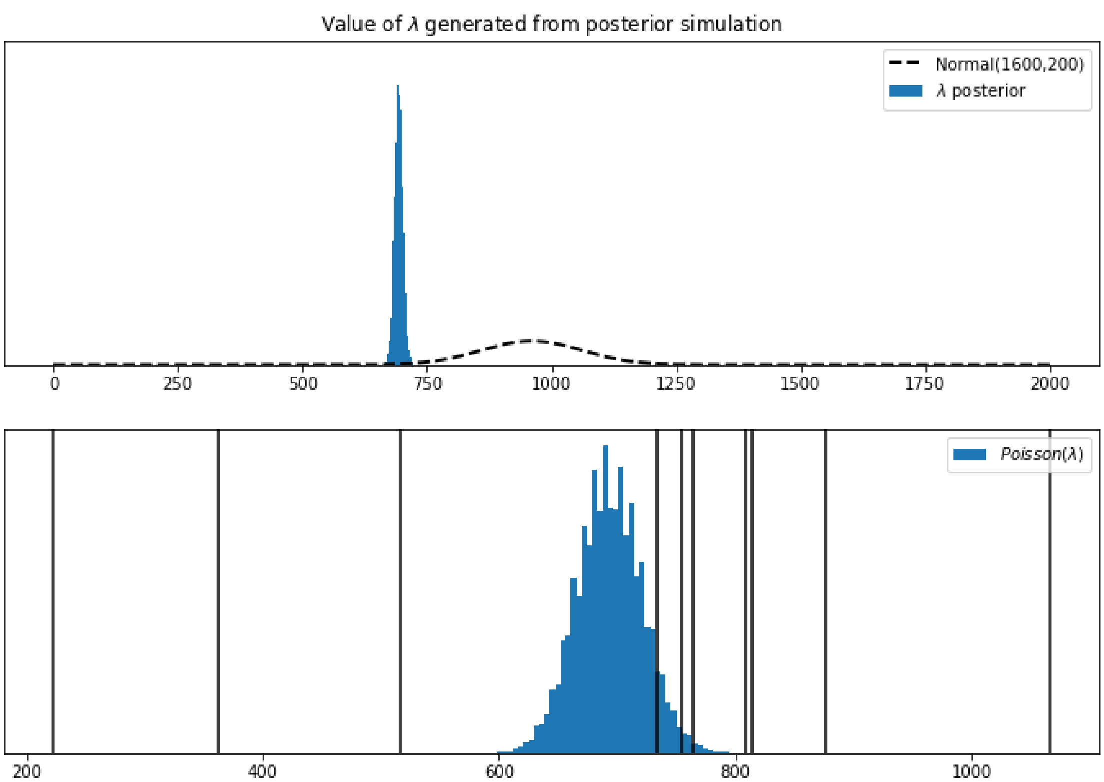
```
INFO:cmdstanpy:compiling stan program, exe file:
/mnt/c/ola/DataAnalytics/lab03/sp_air
INFO:cmdstanpy:compiler options: stanc_options=None, cpp_options=None
```

```
data{
    int N;
    int y[N];
}
parameters{
    real<lower=0> lambda;
}
model{
    lambda ~ normal(960,100);

    for (n in 1:N) {
      y[n] ~ poisson(lambda);
    }
}
generated quantities {
  vector[N] y_new;
  for (n in 1:N)
    y_new[n] = poisson_rng(lambda);
}


INFO:cmdstanpy:compiled model file: /mnt/c/ola/DataAnalytics/lab03/sp_air
INFO:cmdstanpy:start chain 1
INFO:cmdstanpy:start chain 2
INFO:cmdstanpy:start chain 3
INFO:cmdstanpy:start chain 4
INFO:cmdstanpy:finish chain 3
INFO:cmdstanpy:finish chain 2
INFO:cmdstanpy:finish chain 1
INFO:cmdstanpy:finish chain 4
```

[16]:
```python
lambda_mean = np.mean(lbda)
hdi_89 = az.hdi(lbda, 0.89)
print('mean(lambda) : ', lambda_mean)
print('HDI 89%: [',*['{:4.2f}'.format(k) for k in hdi_89],']')
```

```
mean(lambda) :  693.65051725
HDI 89%: [ 680.04 706.85 ]
```

[17]:
```python
fig, ax = plt.subplots(2,1, figsize=(12, 8))
ax[0].hist(lbda, bins=50, density=True)
x=np.linspace(0,2000,2000)
ax[0].set_title(r'Value of $\lambda$ generated from posterior simulation')
ax[0].plot(x, stats.norm.pdf(x, 960, 100), color='black', linestyle='dashed',␣
 ↪linewidth=2)
ax[0].legend(['Normal(1600,200)','$\lambda$ posterior'])
ax[0].set_yticks([])

ax[1].hist(y_new.flatten(), bins=50, density=True, zorder=1)
ax[1].legend(['$Poisson(\lambda)$'])
for i in range(0,len(c2)):
    ax[1].axvline(x = c2[i], color='black')
ax[1].set_yticks([])

plt.show()
```



We can see that we missed with our priors, but there was enough data to overwrite the shape given by priors.

5 2. Passenger death depends upon miles flown

Assuming that the yearly number of passenger deaths is related to miles flown that same year, we can define the model as:

$$y_i \sim \text{Poisson}(\lambda)$$

$$\lambda = aM_f$$

$$a \sim \text{Normal}(\mu, \sigma)$$

where: M_f is a number of miles flown. We can plot number of passenger deaths against miles flown and observe the results:

We need to assume values of μ and σ without knowing the real values of passenger deaths (before we observe data). To do so we'll use data from previous calculations. We assumed that there should be normal distribution of passenger deaths with mean around 1600 people, now taking into consideration data of miles flown we can calculate how it should impact our previous model.

[18]:
```
960/np.mean( Airline_data['Miles flown [100 mln miles]'].values[:])
```

[18]:
```
0.1679554917946744
```

Calculation done this way won't be precise, but we can at least know the scales of values we are operating on. Knowing that in the previous example there was enough data to shape our posterior while suppressing priors we can try and see what'll happen. The model now looks like this:

$$y_i \sim \text{Poisson}(\lambda)$$

$$\lambda = aM_f$$

$$a \sim \text{Normal}(0.168, 0.05)$$

where: M_f is a number of miles flown.

[19]:
```
with open('PD_MF1.stan', 'r') as file:
    print(file.read())
```

```
data {
  int N;
  vector[N] Mf;
  int y[N];

  int M;
  vector[M] toget;
}
parameters {
  real<lower=0> alpha;
```

```
}
model {
  alpha ~ normal(0.168,0.05);
  for (k in 1:N) {
    y[k] ~ poisson(alpha*Mf[k]);
  }
}
generated quantities {
  int y_sim[N];
  real lambda[N];
  int y_toget[M];

  for (k in 1:N) {
    lambda[k] = alpha*Mf[k];
    y_sim[k] = poisson_rng(alpha*Mf[k]);
  }
  for (k in 1:M) {
    y_toget[k] = poisson_rng(alpha*toget[k]);
  }
}
```

[20]:

```
Mf_model = CmdStanModel(stan_file='PD_MF1.stan')
```

```
INFO:cmdstanpy:compiling stan program, exe file:
/mnt/c/ola/DataAnalytics/lab03/PD_MF1
INFO:cmdstanpy:compiler options: stanc_options=None, cpp_options=None
INFO:cmdstanpy:compiled model file: /mnt/c/ola/DataAnalytics/lab03/PD_MF1
```

[21]:

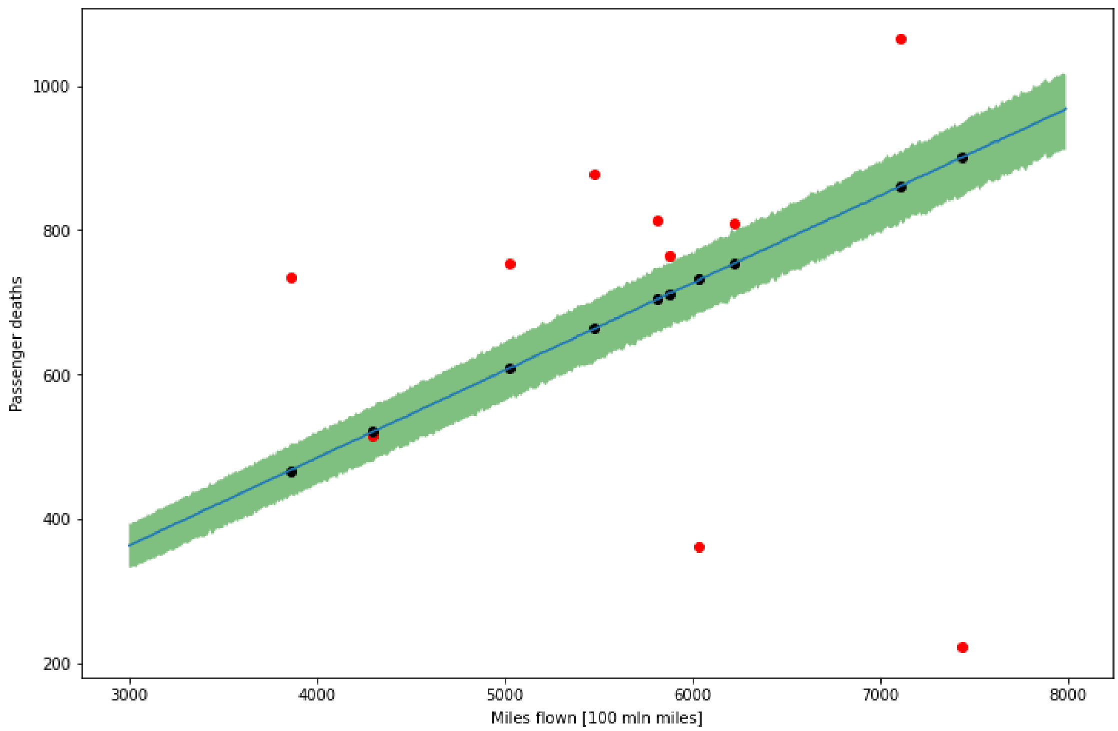
```
 toget = list(range(3000,8000,10));
data = dict(N = len(Airline_data),
            Mf = Airline_data['Miles flown [100 mln miles]'].values,
            y = Airline_data['Passenger deaths'].values,
           toget = toget,
           M = len(toget))
fit = Mf_model.sample(data=data)

alpha=fit.stan_variable('alpha')
lbda=fit.stan_variable('lambda')
y_sim=fit.stan_variable('y_sim')
y_got = fit.stan_variable('y_toget')
```

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

```
INFO:cmdstanpy:finish chain 4
```

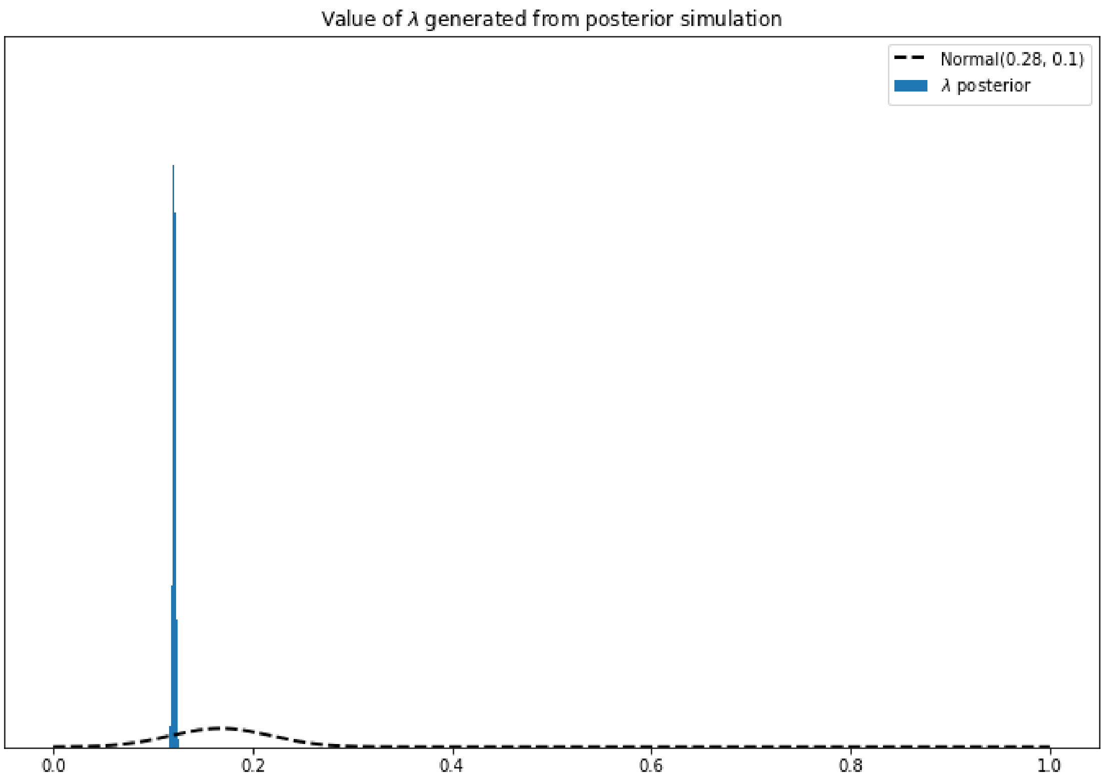
[22]:
```
hdi_ysim = az.hdi(y_got,0.89)
fig, ax = plt.subplots(figsize=(12, 8));
ax.fill_between(toget, hdi_ysim[:,0], hdi_ysim[:,1], facecolor='green', alpha=0.
 ↪5);
ax.scatter( Airline_data['Miles flown [100 mln miles]'].values[:],␣
 ↪Airline_data['Passenger deaths'].values[:],color='red');
ax.scatter( Airline_data['Miles flown [100 mln miles]'].values[:], np.
 ↪mean(y_sim,0),color='black');
ax.plot( toget, np.mean(y_got,0));
ax.set_ylabel('Passenger deaths');ax.set_xlabel('Miles flown [100 mln miles]');
```

```
/mnt/c/Users/jkurek1/Desktop/DataAnalytics/venv/lib/python3.6/site-
packages/arviz/stats/stats.py:459: FutureWarning: hdi currently interprets 2d
data as (draw, shape) but this will change in a future release to (chain, draw)
for coherence with other functions
  FutureWarning,
```

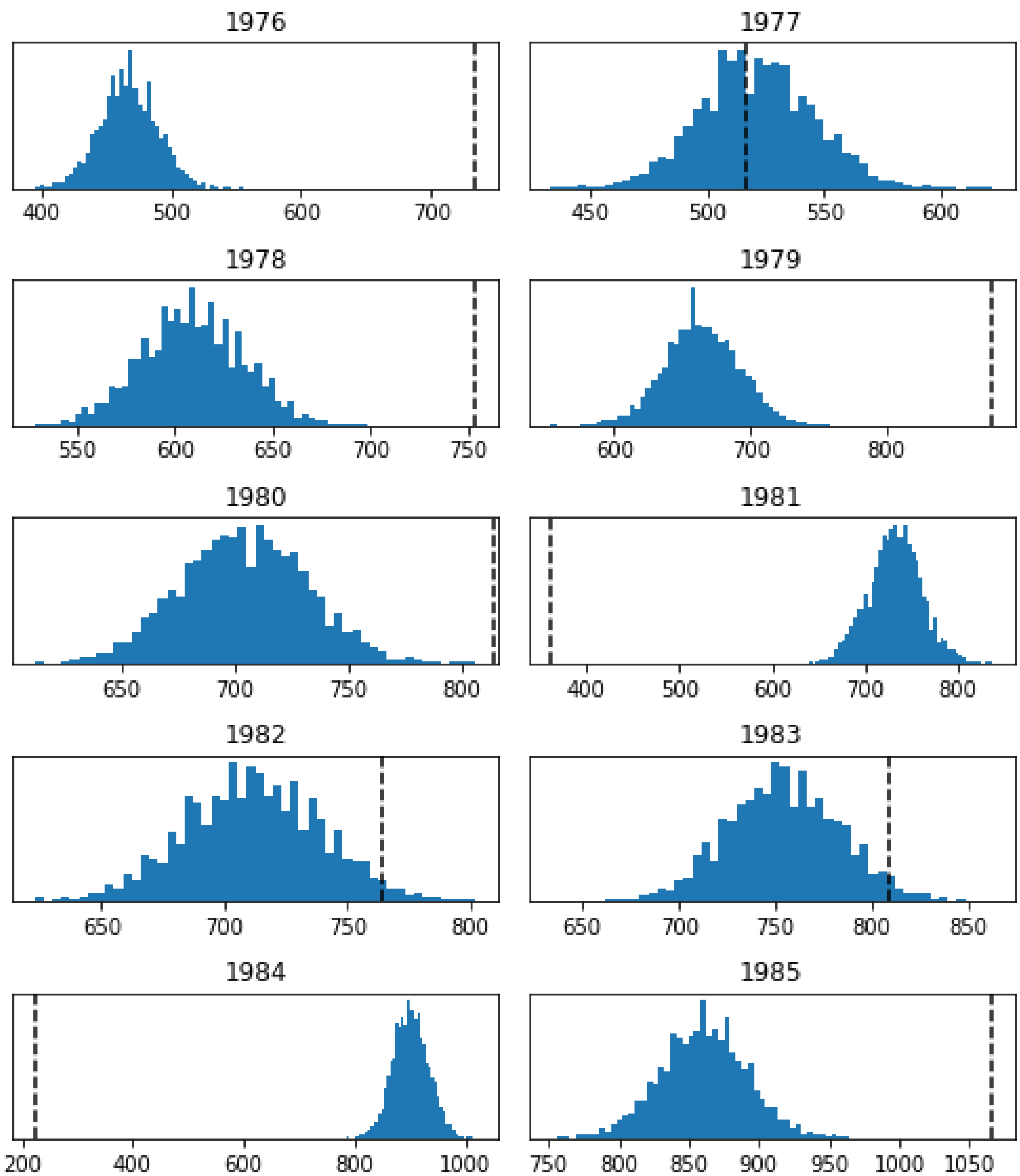


[23]:
```
fig, ax = plt.subplots(figsize=(12, 8))
ax.hist(alpha, bins=50, density=True)
x=np.linspace(0,1,2000)
ax.set_title(r'Value of $\lambda$ generated from posterior simulation')
```

```
ax.plot(x, stats.norm.pdf(x, 0.168, 0.05), color='black', linestyle='dashed',␣
↪linewidth=2)
ax.legend(['Normal(0.28, 0.1)','$\lambda$ posterior'])
ax.set_yticks([]);
```



[24]:
```
fig, ax = plt.subplots(5, 2, figsize=(7, 8))
axr= ax.ravel()
for i in range(len(axr)):
    axr[i].hist(y_sim[:,i],bins=50,density=True)
    axr[i].set_title(Airline_data.index[i])
    axr[i].axvline(x = Airline_data['Passenger deaths'].values[i],␣
↪color='black', ls='--')
    axr[i].set_yticks([])

fig.tight_layout()
plt.show()
```


1976
400 500 600 700
1977
450 500 550 600
1978
550 600 650 700 750
1979
600 700 800
1980
650 700 750 800
1981
400 500 600 700 800
1982
650 700 750 800
1983
650 700 750 800 850
1984
200 400 600 800 1000
1985
750 800 850 900 950 1000 1050