# STAT529
# Applied Bayesian Decision Theory
# Homework 2

Fangda Li
li1208@purdue.edu

March 10, 2017

# 1 Sensitive Question 1

First of all, we obtain the restricted *generalized* Beta distribution that best fits the elicited histogram, where the best fitted Beta distribution minimizes the maximum difference between its cdf and the cdf of the data histogram. The results of fitting are $\alpha = 5.8, \beta = 6.7$ for the international histogram, as shown in Figure 1. Simulation programs for this question can be found in Subsection 5.1.

a) The interval loss function for interval $(a, b)$ is defined as:

$$L[(a,b),\theta] = \begin{cases} \frac{b-a}{M}, \text{ if } \theta \in (a,b) \\ 1, \text{ else} \end{cases} .$$

Then, we can further define the expected posterior loss:

$$\begin{aligned} E\{L[(a,b),\theta]|y\} &= \int\limits_{\theta \notin (a,b)} 1\pi(\theta|y)d\theta + \int\limits_{\theta \in (a,b)} \frac{b-a}{M}\pi(\theta|y)d\theta \\ &= P(\theta \notin (a,b)|y) + \frac{b-a}{M}P(\theta \in (a,b)|y) \\ &= 1 + (\frac{b-a}{M} - 1)P(\theta \in (a,b)|y). \end{aligned}$$

The Bayes interval will make the expected posterior loss minimal. Since our prior is a generalized $Beta(\alpha, \beta)$, our posterior is its conjugate $Beta(\alpha+y, \beta+n-y)$, which is also a generalized Beta on $(-0.5, 1.5)$. Now, we restrict the posterior $\pi(\theta|y, m)$ on the interval $(0, 1)$:

$$\pi_r(\theta|y,m) = \frac{\pi(\theta|y,m)}{B(1|y,m) - B(0|y,m)},$$

where $B(x|y, m)$ represents the cdf of $\pi(\theta|y, m)$. Then, the probability in the expected posterior loss can be represented as:

$$\begin{aligned} P(\theta \in (a,b)|y) &= \int\limits_{\theta \in (a,b)} \pi_r(\theta|y,m)d\theta \\ &= \frac{\int\limits_{\theta \in (a,b)} \pi(\theta|y,m)d\theta}{B(1|y,m) - B(0|y,m)} \\ &= \frac{B(b|y,m) - B(a|y,m)}{B(1|y,m) - B(0|y,m)}. \end{aligned}$$

So far, we have obtained a representation of the expected posterior loss in terms of $a, b$. Next, we iterate through all the possible pairs of $(a, b)$ to find the interval with the minimal expected posterior loss. According to my simulation, the Bayes interval is **(0.26, 0.63)**.

b) In order to find the 95% Bayesian credible region, we first generate sufficient $\theta$ samples using the posterior distribution. Then, we sort the samples and knock off the first 2.5%

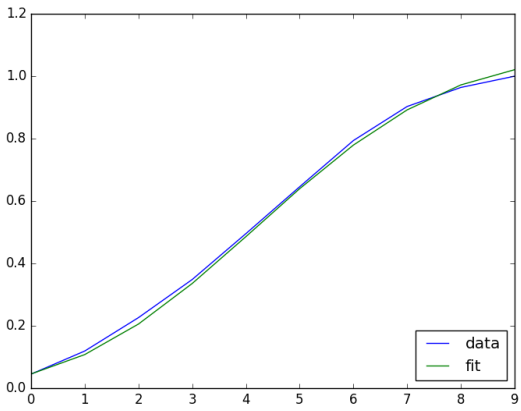


Figure 1: Fitted restricted generalized Beta cdf on the data histogram cdf (interpolated).

and last 2.5% of the sorted samples. The interval in which the remaining samples reside will be our 95% Bayesian credible region. According to my simulation, the 95% Bayesian credible region is **(0.21, 0.78)**.

# 2 Sensitive Question 2

The *international* data is used in this problem. Simulation programs for this question can be found in Subsection 5.2.

a) Recall that the likelihood function is the pdf of a binomial distribution:

$$f(y|\theta) = \binom{m}{y} \frac{(\theta + 0.5)^y (1.5 - \theta)^{m-y}}{2^{-m}}.$$

In order to find the Maximum Likelihood Estimate (MLE) of $\theta$, we first differentiate $logf(y|\theta)$ with respect to $\theta$:

$$\begin{aligned} \frac{d(logf(y|\theta))}{d\theta} &= \frac{d(Constant + ylog(\theta + 0.5) + (m - y)log(1.5 - \theta)}{d\theta} \\ &= \frac{y}{\theta + 0.5} + \frac{(y - m)}{1.5 - \theta}. \end{aligned}$$

By setting the derivative above to 0, we obtain the MLE of $\theta$:

$$\hat{\theta} = \frac{2y}{m} - 0.5 = \frac{2 \times 16}{33} - 0.5 \approx \mathbf{0.47}.$$

b) In order to find the 95% Bayesian credible region with a *standard* Beta prior, we first find the best fit of the international data histogram using a standard Beta distribution. The best fitted standard Beta distribution minimizes the maximum difference between its cdf and the cdf of the data histogram. The results of fitting are $\alpha_{int} = 1.2, \beta_{int} = 1.5$ as shown in Figure 2.

Since the posterior distribution of a standard Beta prior cannot be easily recognized, we use the *Accept/Reject* algorithm to generate samples of the posterior distribution using directly the prior and the likelihood function. More specifically, the Accept/Reject algorithm tells us to keep the $\theta$s generated from $\pi(\theta)$ with the following probability:

$$P = \frac{t(\theta)}{Mg(\theta)} = \frac{f(y|\theta)\pi(\theta)}{M\pi(\theta)} = \frac{f(y|\theta)}{f(y|\hat{\theta})}.$$

After we have generated a sufficient number of keepers, we apply the same counting method as in Question 1b to obtain the 95% Bayesian credible region. According to my simulation, the 95% Bayesian credible region is found to be **(0.20, 0.72)**.

c) In this problem, we use a non-informative prior: Jeffery's Prior. First, recall the Fisher Information Function, $I(\theta)$:

$$I(\theta) = -E[\frac{\partial^2}{\partial\theta^2} logf(y|\theta)].$$

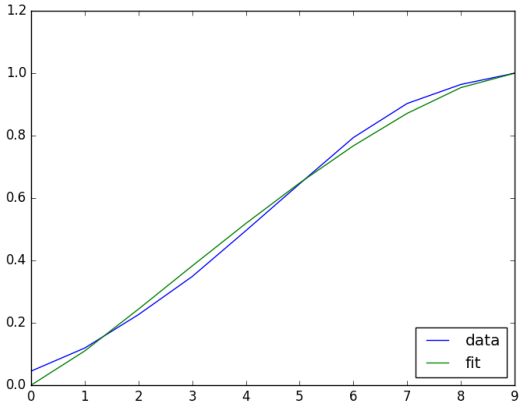


Figure 2: Fitted standard Beta cdf on the data histogram cdf (interpolated).

Then, Jeffery's prior is obtained by taking $\pi(\theta) \propto [I(\theta)]^{0.5}$. Now, let's plug our binomial likelihood function into the representation of Fisher Information Function:

$$\begin{aligned} I(\theta) &= -E[\frac{\partial^2}{\partial\theta^2}(Constant + ylog\theta + (m-y)log(1-\theta))] \\ &= -E[\frac{\partial}{\partial\theta}(\frac{y}{\theta} + \frac{(y-m)}{1-\theta})] \\ &= -E[-\frac{y}{\theta^2} + \frac{(y-m)}{(1-\theta)^2}] \\ &= \frac{E[y]}{\theta^2} + \frac{(E[y]-m)}{(1-\theta)^2} \\ &= \frac{m\theta}{\theta^2} + \frac{(m\theta-m)}{(1-\theta)^2} \\ &= \frac{m}{\theta(1-\theta)}. \end{aligned}$$

Subsequently, we have $\pi(\theta) \propto [I(\theta)]^{0.5} \propto \theta^{0.5-1}(1-\theta)^{0.5-1} \propto Beta(0.5, 0.5)$. Using the standard Beta distribution $Beta(0.5, 0.5)$ to generate $\theta$ samples, we apply the Accept/Reject algorithm to obtain the keepers. Finally, we use the counting technique again to obtain the 95% Bayesian credible region. According to my simulation, the 95% Bayesian credible region is found to be **(0.17, 0.78)**.

d) Now, we use the data histogram directly as our prior to generate $\theta$ samples. By applying the Accept/Reject algorithm and the counting technique on the keepers, we can obtain the 95% Bayesian credible region. According to my simulation, the 95% Bayesian credible region is found to be **(0.20, 0.70)**.

When comparing the three credible regions, we found the following:

- The credible region from Jeffery's prior is the largest. This makes sense since Jeffery's prior is non-informative, which leads to more uncertainty in the

- Since standard Beta prior is just a continuous approximation of the histogram prior, the two priors tends to provide the same amount of information. As a result, the lengths of the two credible regions are similar.

# 3 Defects in Cloth

Let $X$ denotes the number of defects in a random cloth sample on a given day. The distribution of $X$ is Poisson with mean $\theta$:

$$f(x|\theta) = \frac{e^{-\theta}\theta^x}{x!}.$$

The prior $\pi(\theta)$ is unknown. Simulation programs for this question can be found in Subsection 5.3.

a) Two Empirical Bayes procedures are presented as follows.

**Method I**: If we recognize the form of the Bayes procedure, we can use the past observations to approximate the true Bayes procedure $t_\pi(x)$. If the number of past observations approaches infinity, the approximated Bayes procedure $t_k(x)$ will converge to the true Bayes procedure. Now, suppose the form of the Bayes procedure is to minimize the overall expected loss $R(t_\pi, \pi)$ using a quadratic loss function. Subsequently, the Bayes procedure $t_\pi(x)$ is simply choosing the *posterior mean*, $E[\theta|x]$:

$$\begin{aligned}
t_\pi(x) &= E[\theta|x] \\
&= \int \theta\pi(\theta|x)d\theta \\
&= \int \theta \frac{f(x|\theta)\pi(\theta)}{\int \text{above } d\theta} d\theta \\
&= \int \theta \frac{\frac{e^{-\theta}\theta^x}{x!}\pi(\theta)}{\int \frac{e^{-\theta}\theta^x}{x!}\pi(\theta)d\theta} d\theta \\
&= \frac{\int \theta\frac{e^{-\theta}\theta^x}{x!}\pi(\theta)d\theta}{\int \frac{e^{-\theta}\theta^x}{x!}\pi(\theta)d\theta} \\
&= \frac{\int (x+1)\frac{e^{-\theta}\theta^{x+1}}{(x+1)!}\pi(\theta)d\theta}{\int \frac{e^{-\theta}\theta^x}{x!}\pi(\theta)d\theta} \\
&= \frac{(x+1)P(\{X=x+1\})}{P(\{X=x\})}.
\end{aligned}$$

Since the current observation is $x = 4$, based on counting the past observations, our Empirical Bayes procedure is:

$$\begin{aligned}
t_k(x=4) &= \frac{(4+1)P(\{X=4+1\})}{P(\{X=4\})} \\
&= \frac{5 \times \frac{6}{50}}{\frac{13}{50}} \approx 2.31.
\end{aligned}$$

**Method II**: If we recognize the form of the prior, we can use the past observations to approximate the true prior $\pi(\theta)$. If the number of past observations approaches infinity, the approximated prior $\pi_k(\theta)$ will converge to the true prior. Now, suppose the prior is in the Gamma family with unknown parameters, that is:

$$\pi(\theta) = \frac{1}{\beta^\alpha \Gamma(\alpha)} \theta^{\alpha-1} e^{-\frac{\theta}{\beta}}.$$

Then, the pmf of of the observations becomes (using total probability theorem):

$$\begin{aligned} P(\{X = x\}) &= \int f(x|\theta)\pi(\theta)d\theta \\ &= \int \frac{e^{-\theta}\theta^x}{x!} \frac{1}{\beta^\alpha \Gamma(\alpha)} \theta^{\alpha-1} e^{-\frac{\theta}{\beta}} d\theta. \\ &= \frac{1}{x!} \frac{\Gamma(\alpha + x)}{\Gamma(\alpha)} \frac{\beta^x}{(1+\beta)^{\alpha+x}} \end{aligned}$$

Now, we can iterate through possible $(\alpha, \beta)$ pairs to find the pair that gives the least difference between the pmf and the observations ("toothpicks") at two particular points: $X = 4$ and $X = 5$. Subsequently, we can obtain the prior by plugging in the best pair, $(\hat{\alpha}, \hat{\beta})$. With the prior, we can write out the posterior:

$$\begin{aligned} \pi(\theta|x) &= \frac{f(x|\theta)\pi(\theta|\hat{\alpha}, \hat{\beta})}{\int \text{above } d\theta} \\ &\propto \frac{e^{-\theta}\theta^x}{x!} \frac{1}{\hat{\beta}^{\hat{\alpha}} \Gamma(\hat{\alpha})} \theta^{\hat{\alpha}-1} e^{-\frac{\theta}{\hat{\beta}}}. \\ &\propto \theta^{\hat{\alpha}+x-1} e^{-\frac{\frac{-\theta}{\hat{\beta}}}{1+\hat{\beta}}} \end{aligned}$$

Note that the posterior is also a Gamma distribution: $Gamma(\hat{\alpha} + x, \frac{\hat{\beta}}{1+\hat{\beta}})$, where $x = 4$. According to my simulation, $\hat{\alpha} = 2.2$ and $\hat{\beta} = 0.3$.

In order to find the 95% Bayesian credible region, we first generate sufficient $\theta$ samples using the posterior distribution. Then, we sort the samples and knock off the first 2.5% and last 2.5% of the sorted samples. The interval in which the remaining samples reside will be our 95% Bayesian credible region. According to my simulation, the 95% Bayesian credible region is **(1.05, 6.95)**.

However, I believe the result is not so convincing. The reason is that even the fitted Gamma distribution approximates the histogram well on $X = 4$ and $X = 5$, it fits very poorly on other points, as shown in Figure 3. A wide range of parameters of the Gamma distribution has been attempted, yet it still fails to resemble the histogram.

b) In lecture, the following distributions are provided as hyper parameters: $h_1(\alpha)$ is uniform on interval $[1, 6]$ and $h_2(\beta|\alpha)$ is uniform on interval $[\frac{2}{\alpha}, \frac{6}{\alpha}]$. Now, based on the hyper parameters, many hyper priors $\pi(\theta|\alpha, \beta)$ can be generated. Then, from each generated

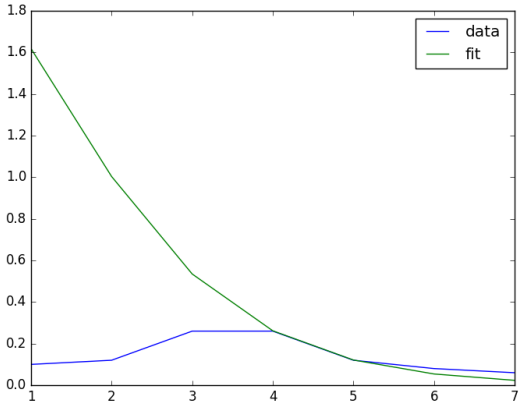


Figure 3: Fitted Gamma distribution on the cloth histogram.

prior, we can generate many samples of $\theta$. Finally, we keep all the generated samples of $\theta$ with the following probability:

$$P = \frac{f(x=4|\theta)}{max_\theta f(x=4|\theta)}$$

According to my simulation, the 95% Bayesian credible region is **(1.81, 6.18)**.

# 4 Sensitive Question 3

Simulation programs for this question can be found in Subsection 5.4.

a) First, we find the best standard Beta fit on the two given histograms: $\alpha_{int} = 1.2, \beta_{int} = 1.5$ and $\alpha_{dom} = 0.9, \beta_{dom} = 1.4$. Using the two fitted Beta distribution to generate $\theta_{int}$ and $\theta_{dom}$ samples, we apply the Accept/Reject algorithm again just like in the previous problems to obtain the keepers. The probability $P$ of making a keeper is:

$$\begin{aligned} P &= \frac{f(x_{int}|\theta_{int})f(x_{dom}|\theta_{dom})}{f(x_{int}|\hat{\theta}_{int})f(x_{dom}|\hat{\theta}_{dom})} \\ &= \frac{(\theta_{dom}+0.5)^{16}(1.5-\theta_{dom})^{11}(\theta_{int}+0.5)^{16}(1.5-\theta_{int})^{17}}{(\hat{\theta}_{dom}+0.5)^{16}(1.5-\hat{\theta}_{dom})^{11}(\hat{\theta}_{int}+0.5)^{16}(1.5-\hat{\theta}_{int})^{17}}, \end{aligned}$$

where $\hat{\theta}_{dom} = \frac{2\times16}{27} - 0.5$ and $\hat{\theta}_{int} = \frac{2\times16}{33} - 0.5$ are the MLEs. Among the keepers, we count the number of $(\theta_{int}, \theta_{dom})$ pairs that $\theta_{int} \leq \theta_{dom}$, and divide that number by the total number of keepers to obtain the posterior probability of $\theta_{int} \leq \theta_{dom}$. According to my simulation, $P(\{\theta_{dom} \geq \theta_{int}\})$ is **0.755**.

b) Now we need to use Hierarchical Bayes model to obtain the same probability. Note that the hyper parameters $\alpha, \beta$ have the following distribution: $h_1(\alpha)$ is uniform on interval $[1.8, 6]$ and $h_2(\beta|\alpha)$ is uniform on interval $[0.3\alpha + 1.75, 0.75\alpha + 3]$. Based on the hyper

parameters, many hyper priors $\pi(\theta_{int}, \theta_{dom}|\alpha, \beta)$ can be generated. Then, from each generated prior, we can generate many samples of $\theta$. Finally, we keep all the generated samples of $\theta_{int}, \theta_{dom}$ with the same probability in the previous step. According to my simulation, $P(\{\theta_{dom} \geq \theta_{int}\})$ is **0.697**.

# 5 Code

## 5.1 Q1

```
from pylab import *
from scipy.stats import beta as std_beta
from scipy.special import gamma

def get_gen_beta_pdf(x, alpha, beta, a, b):
  '''
    Return the pdf of a generalized beta function on given points.
  '''
  return gamma(alpha+beta)*1.0 / (gamma(alpha)*gamma(beta)) * \
      ( (x-a)**(alpha-1) * (b-x)**(beta-1) *1.0 / (b-a)**(alpha+beta-1) )

def find_best_gen_beta_fit(hist_data):
  '''
    Find the (alpha, beta) of the closest generalized (-0.5 to 1.5) Beta
 ↪  distribution.
  '''
  hist_data = hist_data / sum(hist_data)
  hist_cdf = cumsum(hist_data)
  best_alpha, best_beta = 0, 0
  best_err = 99
  for alpha in linspace(0.1,10,100):
    for beta in linspace(0.1,10,100):
      cdf = cumsum( get_gen_beta_pdf(linspace(-0.5,1.5,20), alpha, beta,
       ↪  -0.5, 1.5) ) * 0.1
      # Restrict on [0,1]
      cdf = cdf[4:14] / (cdf[14] - cdf[4])
      max_err = max(abs(cdf - hist_cdf))
      if max_err < best_err:
        best_err = max_err
        best_alpha, best_beta = alpha, beta
  print "Best gen Beta fit is alpha = %f, beta = %f" % (best_alpha,
   ↪  best_beta)
  plot(hist_cdf, label='data')
  cdf = cumsum( get_gen_beta_pdf(linspace(-0.5,1.5,20), best_alpha,
   ↪  best_beta, -0.5, 1.5) ) * 0.1
```

```
  cdf = cdf[4:14] / (cdf[14] - cdf[4])
  plot( cdf , label='fit')
  legend(loc=4)
  show()
  return best_alpha, best_beta

def find_bayes_interval(alpha, beta):
  '''
    Find the interval that gives the minimal expected posterior loss using
 ↪  gen Beta.
  '''
  cdf = cumsum( get_gen_beta_pdf(linspace(-0.5,1.5,200), alpha, beta, -0.5,
  ↪  1.5) ) * 0.01
  best_a, best_b = 0, 0
  best_loss = 999
  for a in linspace(0.01,1.00,100):
    for b in arange(a,1.0,0.01):
      loss = 1 + (b-a-1) * (cdf[int((b+0.5) / 0.01)] - cdf[int((a+0.5) /
       ↪  0.01)]) / (cdf[149] - cdf[49])
      if loss < best_loss:
        best_loss = loss
        best_a, best_b = a, b
  print "Best Bayes interval is (%f, %f)" % (best_a, best_b)
  return best_a, best_b

def find_bayes_credible_region(alpha, beta, perc):
  '''
    Find the Bayes credible region by knocking off the head and tail using
 ↪  gen Beta.
  '''
  cdf = cumsum( get_gen_beta_pdf(linspace(-0.5,1.5,200), alpha, beta, -0.5,
  ↪  1.5) ) * 0.01
  cdf = cdf[49:149] / (cdf[149] - cdf[49])
  for a in arange(len(cdf)):
    b = len(cdf)-a-1
    if cdf[b]-cdf[a] < perc:
      print "Bayes credible region is (%f, %f)" % (a*1.0/len(cdf),
       ↪  b*1.0/len(cdf))
      return a, b

def main():
  hist_int = array([0.45,0.74,1.07,1.22,1.47,1.50,1.48,1.09,0.61,0.36])
  hist_dom = array([0.55,0.90,1.54,1.59,1.52,1.30,1.04,0.82,0.47,0.25])
  alpha, beta = find_best_gen_beta_fit(hist_int) # Best gen Beta fit is
  ↪  alpha = 5.800000, beta = 6.700000
```

```
  find_bayes_interval(alpha, beta) # Best Bayes interval is (0.180000,
  ↪ 0.630000)
  find_bayes_credible_region(alpha, beta, 0.95) # Bayes credible region is
  ↪ (0.060000, 0.930000)

if __name__ == '__main__':
  main()
```

## 5.2 Q2

```
from pylab import *
from scipy.stats import beta as std_beta
from scipy.interpolate import interp1d

def find_best_std_beta_fit(hist_data):
  '''
    Find the (alpha, beta) of the closest standard Beta distribution.
  '''
  hist_data = hist_data / sum(hist_data)
  hist_cdf = cumsum(hist_data)
  best_alpha, best_beta = 0, 0
  best_err = 99
  for alpha in linspace(0.1,10,100):
    for beta in linspace(0.1,10,100):
      max_err = max(abs(std_beta.cdf(linspace(0.0,1.0,10), alpha, beta) -
      ↪ hist_cdf))
      if max_err < best_err:
        best_err = max_err
        best_alpha, best_beta = alpha, beta
  print "Best std Beta fit is alpha = %f, beta = %f" % (best_alpha,
  ↪ best_beta)
  plot(hist_cdf, label='data')
  plot( std_beta.cdf(linspace(0.0,1.0,10), best_alpha, best_beta) ,
  ↪ label='fit')
  legend(loc=4)
  show()
  return best_alpha, best_beta

def find_bayes_credible_region_std_beta(alpha, beta, perc, y=16, n=33,
↪ thetah=0.47):
  '''
    Find the Bayes credible region by knocking off the head and tail using
↪ std Beta.
  '''
  N = 10000
```

```
  thetas = std_beta.rvs(alpha, beta, size=N)
  denom = (thetah+0.5)**y * (1.5-thetah)**(n-y)
  P = (thetas+0.5)**y * (1.5-thetas)**(n-y) / denom
  keepers = thetas[rand(N) <= P]
  keepers = sort(keepers)
  offset = int((1 - perc) * len(keepers))
  print "Bayes credible region is (%f, %f)" % (keepers[offset],
  ↪  keepers[-offset])
  return keepers[offset], keepers[-offset]

def find_bayes_credible_region_jeffery(perc, y=16, n=33, thetah=0.47):
  '''
    Find the Bayes credible region by knocking off the head and tail using
↪  Jeffery's prior.
  '''
  return find_bayes_credible_region_std_beta(0.5, 0.5, perc, y, n, thetah)

def find_bayes_credible_region_hist(hist_data, perc, y=16, n=33,
↪  thetah=0.47):
  '''
    Find the Bayes credible region by knocking off the head and tail using
↪  histogram data prior.
  '''
  N = 10000
  hist_data = hist_data / sum(hist_data)
  # Use interpolation to generate continuous cdf
  thetas = choice(arange(0.0,1.0,0.1), size=N, p=hist_data)
  denom = (thetah+0.5)**y * (1.5-thetah)**(n-y)
  P = (thetas+0.5)**y * (1.5-thetas)**(n-y) / denom
  keepers = thetas[rand(N) <= P]
  keepers = sort(keepers)
  offset = int((1 - perc) * len(keepers))
  print "Bayes credible region is (%f, %f)" % (keepers[offset],
  ↪  keepers[-offset])
  return keepers[offset], keepers[-offset]

def main():
  hist_int = array([0.45,0.74,1.07,1.22,1.47,1.50,1.48,1.09,0.61,0.36])
  hist_dom = array([0.55,0.90,1.54,1.59,1.52,1.30,1.04,0.82,0.47,0.25])
  find_best_std_beta_fit(hist_int) # Best std Beta fit is alpha = 1.200000,
  ↪  beta = 1.500000
  find_best_std_beta_fit(hist_dom) # Best std Beta fit is alpha = 0.900000,
  ↪  beta = 1.400000
  find_bayes_credible_region_std_beta(1.2, 1.5, 0.95) # Bayes credible
  ↪  region is (0.202, 0.717)
```

```
  find_bayes_credible_region_jeffery(0.95) # Bayes credible region is
  ↪  (0.165, 0.776)
  find_bayes_credible_region_hist(hist_int, 0.95) # Bayes credible region
  ↪  is (0.200000, 0.700000)


if __name__ == '__main__':
  main()
```

### 5.3 Q3

```
from pylab import *
from scipy.special import gamma
from scipy.special import factorial

def get_gamma_pdf(x, alpha, beta):
  '''
    Return the pdf of gamma function on given points.
  '''
  return 1.0 / (beta**alpha * gamma(alpha)) * x**(alpha-1) * exp(-x/beta)

def get_possion_pmf(x, theta):
  '''
    Return the pdf of possion function on given points.
  '''
  return exp(-theta) * theta**x / factorial(x)

def find_best_gamma_fit(hist_data):
  '''
    Find the best alpha and beta for the gamma prior.
  '''
  hist_data = hist_data / sum(hist_data)
  hist_cdf = cumsum(hist_data)
  best_alpha, best_beta = 0, 0
  best_err = 99
  X = arange(1,8)
  for alpha in arange(0.1,10,0.01):
    for beta in arange(0.1,10,0.01):
      P = 1.0 / factorial(X) * factorial(alpha+X-1) / factorial(alpha-1) *
      ↪  beta**X * (1+beta)**(alpha+X)
      max_err = max( abs(P[3:5] - hist_data[3:5]) )
      # max_err = max( abs(P - hist_data) )
      if max_err < best_err:
        best_err = max_err
        best_alpha, best_beta = alpha, beta
```

```
        best_P = copy(P)
  print "Best gamma fit is alpha = %f, beta = %f" % (best_alpha, best_beta)
  plot(range(1,8), hist_data, label='data')
  plot(range(1,8), best_P, label='fit')
  legend(loc=1)
  show()
  return best_alpha, best_beta

def find_bayes_credible_region_gamma(alpha, beta, perc):
  '''
    Find the Bayes credible region by knocking off the head and tail using
↪   Gamma.
  '''
  r = arange(1,7.01,0.01)
  pdf = get_gamma_pdf(r, alpha, beta)
  pdf = pdf / sum(pdf)
  cdf = cumsum( pdf )
  for a in arange(len(cdf)):
    b = len(cdf)-a-1
    if cdf[b]-cdf[a] < perc:
      print "Bayes credible region is (%f, %f)" % (r[a], r[b])
      return a, b

def find_bayes_credible_region_hb(perc):
  '''
    Find the Bayes credible region using A/R and hyper priors.
  '''
  N = 100
  keepers = array([])
  for i in range(1000):
    # Generate prior distribution
    alpha = rand()*5 + 1 # uniform on [1,6]
    beta = rand()*(6/alpha - 2/alpha) + 2/alpha # uniform on [2/alpha,
↪     6/alpha]
    r = arange(1,7.01,0.01)
    pdf = get_gamma_pdf(r, alpha, beta)
    pdf = pdf / sum(pdf)
    # Generate thetas
    thetas = choice(r, size=N, p=pdf)
    M = max(get_possion_pmf(4, thetas))
    P = get_possion_pmf(4, thetas) / M
    keepers = append(keepers, thetas[rand(N) <= P])
  keepers = sort(keepers)
  # Knocking off the head and tail 2.5%
  offset = int((1 - perc) * len(keepers))
```

```
  print "Bayes credible region is (%f, %f)" % (keepers[offset],
  ↪ keepers[-offset])
  return keepers[offset], keepers[-offset]

def main():
  data =
  ↪ array([4,3,5,5,1,3,1,4,3,5,6,2,4,4,3,4,3,4,4,3,4,2,4,4,6,4,5,3,2,3,7,7,2,5,1,3,6,5
  hist_data, _ = histogram(data, bins=7)
  hist_data = array(hist_data, dtype=float)
  alpha, beta = find_best_gamma_fit(hist_data) # Best gamma fit is alpha =
  ↪ 2.200000, beta = 0.300000
  find_bayes_credible_region_gamma(alpha+4, beta/(1+beta), 0.95) # Bayes
  ↪ credible region is (1.050000, 6.950000)
  find_bayes_credible_region_hb(0.95) # Bayes credible region is (1.810000,
  ↪ 6.180000)

if __name__ == '__main__':
  main()
```

## 5.4 Q4

```
from pylab import *
from scipy.stats import beta as std_beta

def find_probability_posterior(alpha_int, beta_int, alpha_dom, beta_dom):
  '''
    Compute the probability that theta_dom >= theta_int using posterior.
  '''
  N = 10000
  thetah_int = 2.0*16 / 33 -0.5
  thetah_dom = 2.0*16 / 27 -0.5
  thetas_int = std_beta.rvs(alpha_int, beta_int, size=N)
  thetas_dom = std_beta.rvs(alpha_dom, beta_dom, size=N)
  denom = (thetah_dom+0.5)**16 * (1.5-thetah_dom)**11 * (thetah_int+0.5)**16
  ↪ * (1.5-thetah_int)**17
  P_keep = (thetas_dom+0.5)**16 * (1.5-thetas_dom)**11 *
  ↪ (thetas_int+0.5)**16 * (1.5-thetas_int)**17 / denom
  indices = rand(N) <= P_keep
  keepers_int = thetas_int[indices]
  keepers_dom = thetas_dom[indices]
  bools = keepers_dom >= keepers_int
  P_result = sum(bools)*1.0 / bools.size
  print "Probability of theta_dom >= theta_int is %f" % P_result
  return P_result
```

```
def find_probability_hb():
  '''
    Compute the probability that theta_dom >= theta_int using hyper
↪   priors.
  '''
  N = 1000
  thetah_int = 2.0*16 / 33 -0.5
  thetah_dom = 2.0*16 / 27 -0.5
  num_keepers = 0
  num_total = 0
  for i in range(1000):
    # Generate prior distribution
    alpha = rand()*4.2 + 1.8 # uniform on [1.8,6]
    beta = rand()*(0.75*alpha+3 - 0.3*alpha+1.75) + 0.3*alpha+1.75 #
    ↪  uniform on [0.3*alpha+1.75, 0.75*alpha+3]
    thetas_int = std_beta.rvs(alpha, beta, size=N)
    thetas_dom = std_beta.rvs(alpha, beta, size=N)
    # Same as using posterior
    denom = (thetah_dom+0.5)**16 * (1.5-thetah_dom)**11 *
    ↪  (thetah_int+0.5)**16 * (1.5-thetah_int)**17
    P_keep = (thetas_dom+0.5)**16 * (1.5-thetas_dom)**11 *
    ↪  (thetas_int+0.5)**16 * (1.5-thetas_int)**17 / denom
    indices = rand(N) <= P_keep
    keepers_int = thetas_int[indices]
    keepers_dom = thetas_dom[indices]
    bools = keepers_dom >= keepers_int
    num_keepers += sum(bools)
    num_total += bools.size
  P_result = num_keepers*1.0 / num_total
  print "Probability of theta_dom >= theta_int is %f" % P_result
  return P_result

def main():
  alpha_int, beta_int = 1.2, 1.5 # Best std Beta fit is alpha = 1.200000,
  ↪  beta = 1.500000
  alpha_dom, beta_dom = 0.9, 1.4 # Best std Beta fit is alpha = 0.900000,
  ↪  beta = 1.400000
  find_probability_posterior(alpha_int, beta_int, alpha_dom, beta_dom) #
  ↪  Probability of theta_dom >= theta_int is 0.754662
  find_probability_hb() # Probability of theta_dom >= theta_int is
  ↪  0.696833


if __name__ == '__main__':
  main()
```