Bayesian Data Analysis - Assignment 3

October 2, 2017

The language used is Python. The source code is attached in the appendix.

1 Inference for normal mean and diviation

observation model:

$$\frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{1}{2\sigma^2}(y-\mu)^2)$$

$$p(y \mid \mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{1}{2\sigma^2}(y-\mu)^2)$$

uninformative prior:

$$p(\mu, \sigma^2) \propto \sigma^{-2}$$

posterior distribution:

$$\begin{aligned} p(\mu, \sigma^2 \mid y) &\propto \sigma^{-n-2}\exp(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\mu)^2) \\ &= \sigma^{-n-2}\exp(-\frac{1}{2\sigma^2}[(n-1)s^2 + n(\bar{y}^2-\mu)^2]) \end{aligned}$$

where $s = \frac{1}{n-1}\sum_{i=1}^{n}(y_i-\bar{y})^2$

$$\begin{aligned} p(\mu \mid y) &= \int_0^\infty p(\mu, \sigma^2 \mid y) d\sigma^2 \\ &\propto [1 + \frac{n(\mu-\bar{y})^2}{(n-1)s^2}]^{-n/2} \end{aligned}$$

$$\mu \mid y \quad \sim \quad t_{n-1}(\bar{y}, \tfrac{s^2}{n})$$

a)

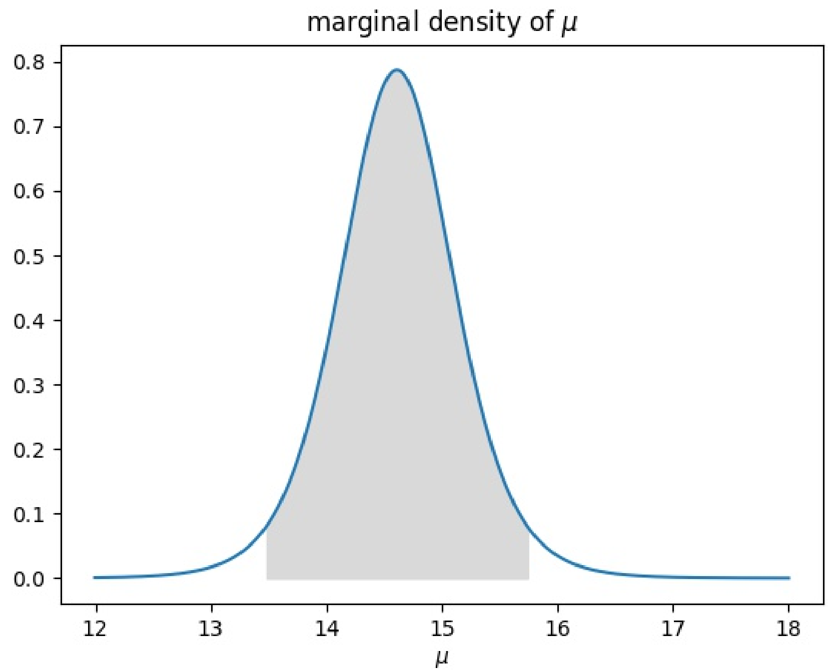

Figure 1: marginal density of μ

mean: 14.611222
median: 14.611222
variance: 0.321949
The central 95% interval: [13.478081,15.744363]

b)

$$\tilde{y} \mid y \sim t_{n-1}(\bar{y}, (1 + \frac{1}{n})s^2)$$

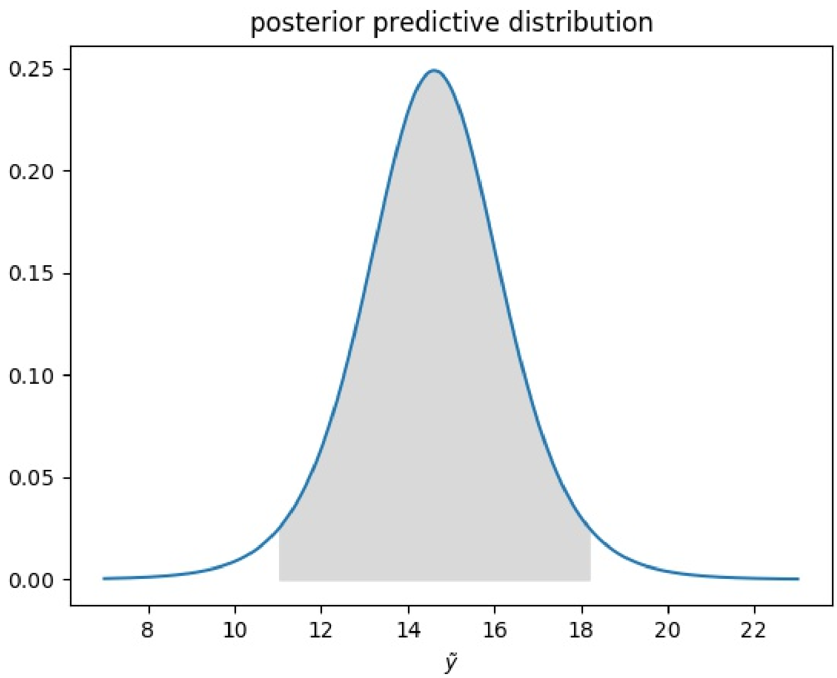

Figure 2: posterior predictive distribution

mean: 14.611222
median: 14.611222
variance: 3.219486
The central 95% interval: [11.027916,18.194529]

2 Inference for difference between proportions

noninformative prior for each group seperately:

$$p(\theta) \propto \theta^{-1/2}(1-\theta)^{-1/2}$$
$$Beta(\tfrac{1}{2},\tfrac{1}{2})$$

posterior for binomial proportion:

$$\begin{aligned} p(\pi \mid y) &\propto \pi^{y}(1-\pi)^{n-y}\pi^{\alpha-1}(1-\pi)^{\beta-1} \\ &= Beta(\pi \mid \alpha+y, \beta+n-y) \end{aligned}$$

$$\text{for } p_0 : n_0 = 674, y_0 = 39$$
$$\text{for } p_1 : n_1 = 680, y_1 = 22$$

a)

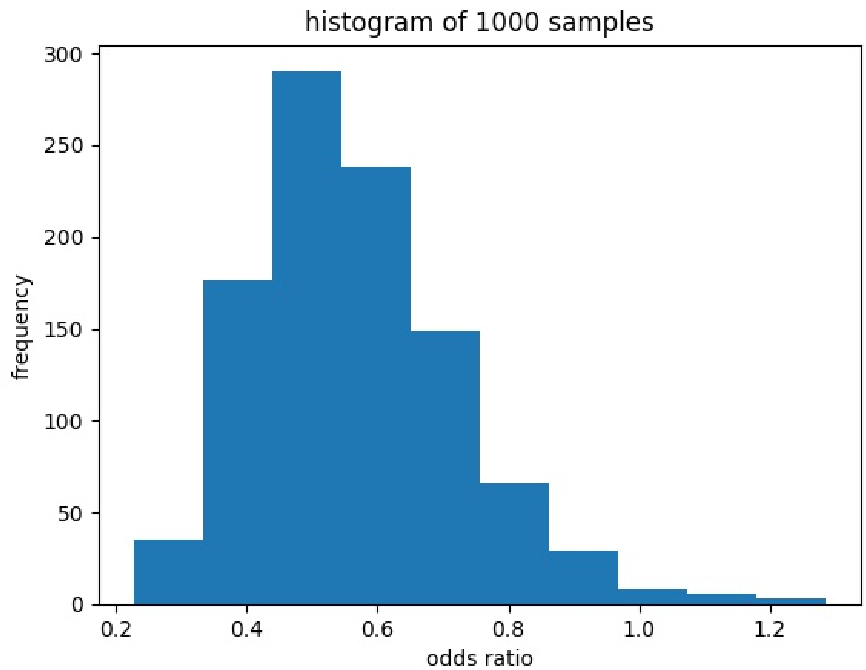

Figure 3: histogram of odds ratio

mean: 0.565833
median: 0.543667
variance: 0.025048
The central 95% interval: [0.319662, 0.946305]

b)

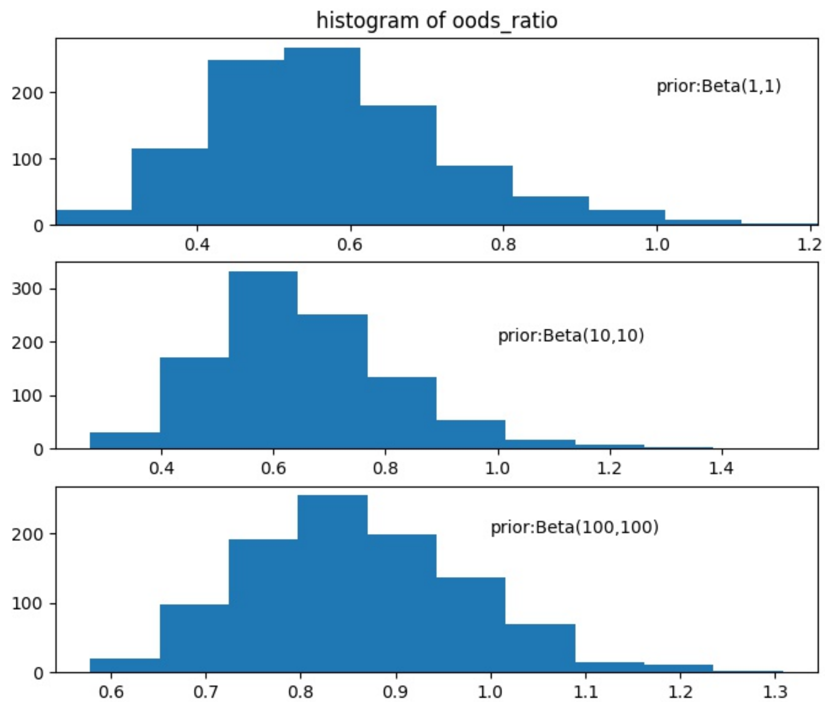

Figure 4: histograms of odds ratio with different priors

Parameters of the prior distrubution		Summaries of the posterior distribution			
$\frac{\alpha}{\alpha+\beta}$	$\alpha+\beta$	mean	median	variance	central 95%interval
0.5	2	0.570945	0.555143	0.023832	[0.317775, 0.933914]
0.5	20	0.654659	0.633300	0.026419	[0.389969, 1.026930]
0.5	200	0.860727	0.853760	0.013966	[0.660620, 1.116229]

Table 1: Summaries of the posterior distribution

Posterior inferences are not particularly sensitive to the prior distribution when $\alpha + \beta$ (prior observations) is relatively small. With $\alpha + \beta$ increasing, the means (and the medians) of posterior distribution increase.

3 Inference for difference between normal means

uninformative prior for each group seperately:

$$p(\mu, \sigma^2) \propto \sigma^{-2}$$

a)

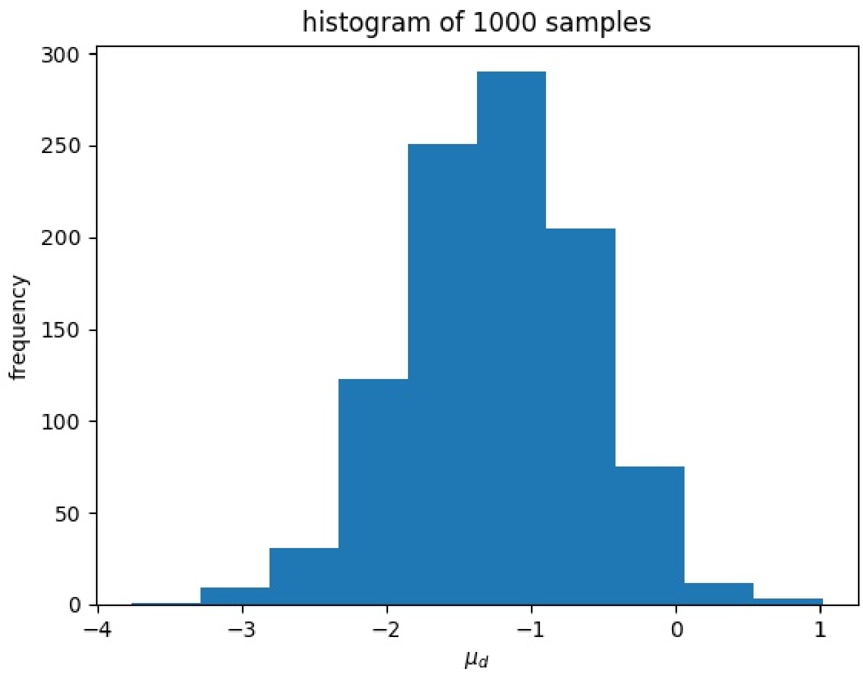

Figure 5: histograms of μ_d

mean: -1.243900
median: -1.236537
variance: 0.394835
The central 95% interval: [-2.488768, -0.045077]

b)

The means are not the same. Because the mean of $\mu_d = \mu_1 - \mu_2 \approx -1.24 \neq 0$, and 0 is outside the central 95% interval.

Appendix

Source code

problem 1

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import sinvchi2
import plot_tools

# data
y = np.loadtxt("windshieldy1.txt")
n = len(y)  # The amount of samples
sample_mean = np.mean(y)
sample_variance = np.var(y, ddof=1)

# set random number generator with seed
rng = np.random.RandomState(seed=0)
# factorize the joint posterior p(mu,sigma2|y) to p(sigma2|y)p(mu|sigma2,y
# sample from the joint posterior using this factorization

# sample from p(sigma2|y)
sigma2 = sinvchi2.rvs(n-1, sample_variance, size=1000, random_state=rng)
# sample from p(mu|sigma2,y)
mu = sample_mean + np.sqrt(sigma2/n)*rng.randn(*sigma2.shape)
# display sigma instead of sigma2
sigma = np.sqrt(sigma2)
# sample from the predictive distribution p(ynew|y)
# for each sample of (mu, sigma)
ynew = rng.randn(*mu.shape)*sigma + mu

# for mu compute the density in these points
```

```
t1 = np.linspace(12, 18, 1000)
#for sigma compute the density in these points
t2 = np.linspace(1,5,1000)
# for ynew compute the density in these points
xynew = np.linspace(7,23,1000)

# compute the exact marginal density for mu
mu_pm = stats.t.pdf((t1-sample_mean)/np.sqrt(sample_variance/n), n-1)/np.s
# mu_pm = stats.t.pdf(t1, df=(n-1), loc=sample_mean,  scale=np.sqrt(sample
mu_mean = stats.t.mean(df=n-1, loc=sample_mean, scale=np.sqrt(sample_varia
mu_median = stats.t.median(df=n-1, loc=sample_mean, scale=np.sqrt(sample_v
mu_variance = stats.t.var(df=n-1, loc=sample_mean, scale=np.sqrt(sample_va
print("For mu:")
print("mean: {:.6f}, median:{:.6f}, variance: {:.6f}"
        .format(mu_mean,mu_median,mu_variance))
# 95% interval
interval_95 = stats.t.interval(0.95, df=n-1, loc=sample_mean, scale=np.sqr
point_min = stats.t.ppf(0.025,df=n-1, loc=sample_mean, scale=np.sqrt(sampl
point_max = stats.t.ppf(0.975,df=n-1, loc=sample_mean, scale=np.sqrt(sampl
print("The central 95% interval:[{:.6f},{:.6f}]".format(point_min,point_ma
x_95_idx = (t1 > point_min) & (t1 < point_max)

plt.plot(t1, mu_pm)
plt.xlabel(r'$\mu$')
plt.fill_between(t1[x_95_idx], mu_pm[x_95_idx], color='0.85')
plt.title("marginal density of " r'$\mu$')

# b)
p_new = stats.t.pdf((xynew-sample_mean)/np.sqrt(sample_variance*(1+1/n)), 

y_mean = stats.t.mean(df=n-1,loc=sample_mean,scale=np.sqrt(sample_variance
y_median = stats.t.median(df=n-1,loc=sample_mean,scale=np.sqrt(sample_vari
y_variance = stats.t.var(df=n-1,loc=sample_mean,scale=np.sqrt(sample_varia
print("For y:")
print("mean: {:.6f}, median:{:.6f}, variance: {:.6f}"
        .format(y_mean,y_median,y_variance))
interval2_95 = stats.t.interval(0.95, df=n-1,loc=sample_mean,scale=np.sqrt
point2_min = stats.t.ppf(0.025,df=n-1,loc=sample_mean,scale=np.sqrt(sample
point2_max = stats.t.ppf(0.975,df=n-1,loc=sample_mean,scale=np.sqrt(sample
print("The central 95% interval:[{:.6f},{:.6f}]".format(point2_min,point2_m
```

```
x_95_idx2 = (xynew > point2_min) & (xynew < point2_max)
plt.figure()
plt.plot(xynew,p_new)
plt.xlabel(r'$\tilde{y}$')
plt.fill_between(xynew[x_95_idx2], p_new[x_95_idx2], color='0.85')
plt.title("posterior predictive distribution")
plt.show()
```

problem 2

```
import numpy as np
from scipy.stats import beta
import matplotlib.pyplot as plt

# a)
# data
n0 = 674
y0 = 39
n1 = 680
y1 = 22

# prior Beta(1/2,1/2)
a = 1/2
b = 1/2

# posterior distribution for p0,p1 seperately
dist0 = beta(a+y0, b+n0-y0)
dist1 = beta(a+y1, b+n1-y1)

# samples from dist0 and dist1
s0 = dist0.rvs(size=1000)
s1 =dist1.rvs(size=1000)

# generate posterior of odds ratio
oods_ratio = (s1/(1-s1))/(s0/(1-s0))
mean = np.mean(oods_ratio)
median = np.median(oods_ratio)
variance = np.var(oods_ratio,ddof=1)
interval_95_min = np.percentile(oods_ratio,2.5)
interval_95_max = np.percentile(oods_ratio,97.5)
```

```python
print("mean: {:.6f}\nmedian: {:.6f}\nvariance:{:.6f}\n"
          "The central 95% interval: [{:.6f}, {:.6f}]"
          .format(mean,median,variance,interval_95_min,interval_95_max))
plt.hist(oods_ratio)
plt.xlabel('odds ratio')
plt.ylabel('frequency')
plt.title("histogram of 1000 samples")

# b)
# case 1: beta(1,1)
pa1 = 1
pb1 = 1

dist_case10 = beta(pa1+y0, pb1+n0-y0)
dist_case11 = beta(pa1+y1, pb1+n1-y1)

s_case10 = dist_case10.rvs(size=1000)
s_case11 =dist_case11.rvs(size=1000)

oods_ratio1 = (s_case11/(1-s_case11))/(s_case10/(1-s_case10))
mean1 = np.mean(oods_ratio1)
median1= np.median(oods_ratio1)
variance1= np.var(oods_ratio1)
interval_95_min1 = np.percentile(oods_ratio1,2.5)
interval_95_max1 = np.percentile(oods_ratio1,97.5)
print("case1:")
print("mean: {:.6f}\nmedian: {:.6f}\nvariance:{:.6f}\n"
          "The central 95% interval: [{:.6f}, {:.6f}]"
          .format(mean1,median1,variance1,interval_95_min1,interval_95_max1


# case 2: beta(10,10)
pa2 = 10
pb2 = 10

dist_case20 = beta(pa2+y0, pb2+n0-y0)
dist_case21 = beta(pa2+y1, pb2+n1-y1)

s_case20 = dist_case20.rvs(size=1000)
s_case21 =dist_case21.rvs(size=1000)
```

```
oods_ratio2 = (s_case21/(1-s_case21))/(s_case20/(1-s_case20))
mean2 = np.mean(oods_ratio2)
median2= np.median(oods_ratio2)
variance2= np.var(oods_ratio2)
interval_95_min2 = np.percentile(oods_ratio2,2.5)
interval_95_max2 = np.percentile(oods_ratio2,97.5)
print("case2:")
print("mean: {:.6f}\nmedian: {:.6f}\nvariance: {:.6f}\n"
          "The central 95% interval: [{:.6f}, {:.6f}]"
          .format(mean2,median2,variance2,interval_95_min2,interval_95_max2

# case 3: beta(100,100)
pa3 = 100
pb3 = 100

dist_case30 = beta(pa3+y0, pb3+n0-y0)
dist_case31 = beta(pa3+y1, pb3+n1-y1)

s_case30 = dist_case30.rvs(size=1000)
s_case31 =dist_case31.rvs(size=1000)

oods_ratio3 = (s_case31/(1-s_case31))/(s_case30/(1-s_case30))
mean3 = np.mean(oods_ratio3)
median3= np.median(oods_ratio3)
variance3= np.var(oods_ratio3)
interval_95_min3 = np.percentile(oods_ratio3,2.5)
interval_95_max3 = np.percentile(oods_ratio3,97.5)
print("case3:")
print("mean: {:.6f}\nmedian: {:.6f}\nvariance: {:.6f}\n"
          "The central 95% interval: [{:.6f}, {:.6f}]"
          .format(mean3,median3,variance3,interval_95_min3,interval_95_max3

# plot the histograms
fig,axes = plt.subplots(nrows=3, ncols=1, figsize=(8, 15))
axes[0].hist(oods_ratio1)
axes[1].hist(oods_ratio2)
axes[2].hist(oods_ratio3)
axes[0].set_title("histogram of oods_ratio")
axes[0].annotate("prior:Beta(1,1)",xy=(1,200))
```

```
axes[1].annotate("prior:Beta(10,10)",xy=(1,200))
axes[2].annotate("prior:Beta(100,100)",xy=(1,200))
axes[0].autoscale(axis='x', tight=True)
plt.show()
```

problem 3

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import sinvchi2
import plot_tools

# data
y1 = np.loadtxt("windshieldy1.txt")
n1 = len(y1)  # The amount of samples
sample_mean1 = np.mean(y1)
sample_variance1 = np.var(y1, ddof=1)

y2 = np.loadtxt("windshieldy2.txt")
n2 = len(y2)  # The amount of samples
sample_mean2 = np.mean(y2)
sample_variance2 = np.var(y2, ddof=1)

#generate samples
s1 = stats.t.rvs(df=n1-1,loc=sample_mean1,scale=np.sqrt(sample_variance1/n
s2 = stats.t.rvs(df=n2-1,loc=sample_mean2,scale=np.sqrt(sample_variance2/n
s_mean_dif = s1 - s2

mean = np.mean(s_mean_dif)
median = np.median(s_mean_dif)
variance = np.var(s_mean_dif,ddof=1)
interval_95_min = np.percentile(s_mean_dif,2.5)
interval_95_max = np.percentile(s_mean_dif,97.5)
print("mean: {:.6f}\nmedian: {:.6f}\nvariance:{:.6f}\n"
            "The central 95% interval: [{:.6f}, {:.6f}]"
            .format(mean,median,variance,interval_95_min,interval_95_max))
plt.hist(s_mean_dif)
plt.xlabel(r'$\mu_{d}$')
plt.ylabel('frequency')
```

```
plt.title("histogram of 1000 samples")
plt.show()
```