Bayesian Statistics HW-6

Weijia Zhao

April 10, 2022

Caution: I keep 6 decimal digits for infinite decimals.

Exercise 1

Rinderpest Virus in Rabbits with Missing Data
(a) We use the following hierarchical prior for this question:

$$\begin{aligned}
\mu(\beta) &\sim Normal(0, \sigma^2 = 1/\tau = 10^6) \\
\tau(\beta) &\sim Gamma(0.001, 0.001) \\
\beta &\sim Normal(\mu(\beta), \sigma^2 = 1/\tau(\beta)) \\
x &\sim TruncatedNormal(61.4, \sigma^2 = 100) \\
\tau_y &\sim Gamma(0.001, 0.001) \\
y &\sim Normal(x\beta_1 + \beta_0, \tau_y)
\end{aligned}$$

For x, we use a truncated normal prior with lower bound 0 and mean to be the empirical sample mean (61.4), ignoring missing value. The prior for precision of regression coefficient is given by normal with mean to be another non informative normal distribution and precision to be a non informative gamma distribution. The outcome variable is assumed to be normal with mean as the product of regression coefficient and x variable, precision to be drawn from a non informative gamma distribution.[1] In general the regression results (including the size and significance) change when we choose different priors, but the overall pattern should be fairly similar across models (i.e. which model is better, the overall fit).

For R^2, since there is no standard definition of R^2 in Bayesian statistics, especially when the model involves missing values, I provide two alternative calculations. The first one (reported in the table as $br2$) is directly borrowed from the online examples and it throw way the observation with missing y. The second one include the missing value by using the imputed value obtained from the model (I do it in the end rather than directly compute this quantity as a deterministic variable in PyMC model building step because it's easier for me to do vector manipulations such as concatenation). We can optionally choose to take the $\max(0, R^2)$ in the end for this average R^2 since it is negative.[2]

[1]It is standard in practice to choose normal prior for the mean and gamma prior for the precision

[2]note that Bayesian R^2 is not defined in a standard and well accepted way, especially when it involves missing data. Different definitions may lead to different results but we should expect the general pattern to be there regardless of exact way how we compute this R^2 as long as it measures the goodness of fit

	mean	sd	hdi_2.5%	hdi_97.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
mu_beta	51.379	211.876	-359.812	475.244	3.038	2.573	9054.0	5675.0	1.0
tau_beta	0.0	0.0	0.0	0.001	0.0	0.0	6134.0	6747.0	1.0
beta[0]	105.811	2.202	101.406	110.238	0.023	0.016	9518.0	10213.0	1.0
beta[1]	-0.019	0.035	-0.086	0.053	0.0	0.0	9261.0	10519.0	1.0
x_m[0]	60.105	10.175	39.613	79.401	0.086	0.061	13994.0	11094.0	1.0
tau_y	0.274	0.147	0.037	0.556	0.002	0.002	8040.0	8943.0	1.0
y_m[0]	104.352	2.459	99.495	109.253	0.022	0.016	13483.0	11727.0	1.0
br2	-0.41	1.127	-2.262	0.7	0.013	0.009	8040.0	8943.0	1.0

Note that $x\_m[0]$ and $y\_m[0]$ are the imputed value for missing x and y respectively. mu_beta and tau_beta are the mean and precision of regression coefficient $beta$, $beta[0]$ and $beta[1]$ are the regression coefficient for constant term and x respectively. tau_y is the precision of y

The R^2 of the regression is given by 0 (-0.41 if not take max and not include the imputed observation, or -0.2193 if not take max and include the imputed observation). As we can see, the relationship between y and x is not linear: as x increases, the value of y first increases and then decreases, so the regression with constant and one regressor (first order) cannot capture this relationship very well and generate negative R^2.

For slope, as well can see β_1 takes value -0.019 and the 95% credible set is given by $(-0.086, 0.053)$ it contains 0. This is mainly because y first increase and then decrease with x. If we only include the first order term, the effect cancels out and the net effect is close to 0.

The estimator for missing value of x is given by 60.105 with credible set (39.163,79.401) and the estimator for missing value of y is given by 104.352 with credible set (99.495,109.253). Notice that the estimator for missing value of x is highly sensitive to the mean of prior distribution, the main reason is that the model itself doesn't provide a clear relationship between x and y, so it is hard to make inference for x based on y.

(b) We use exactly the same prior as specified in part (a) and the only difference is that now we have one additional dimension for β since we include the second order term in the regression. [3]. In addition, to keep the relationship that the newly added component x^2 is indeed the square of the component x, even for the missing value, I choose to add it in the form of $beta[0] \cdot x[:,0] + beta[1] \cdot x[:,1] + beta[2] \cdot x[:,2]$ rather than add a column at the beginning, so we still only need to impute one missing value for x, then the imputed value for x^2 is just the square of the imputed value for x. (For the other option, the model is free to impute the two missing values in column x and x^2 and there is no guarantee that the square relationship holds for the imputed value)

Now, by including an addition square term, the model is able to capture the non-linear relationship between y and x and thus we get a much better $R^2 = 0.5560$ if include the imputed value or $R^2 = 0.528$ if not include the imputed value.

Note as mentioned above, I cannot run the code successfully without split out the constant term from β, now α is the constant term, β_0 and β_1 are the coefficients for the first order term

[3]Unfortunately somehow I cannot get the program run successfully if I only change it here, it always returns some random error indicating some problems with division by 0 in a function call called "true_divide" in one of the chains, and there is very little documentation about this error anywhere. This error goes away if I split the constant term out from β by defining a new variable called α, so I decide to do it this way and I create a separate script file for part (b)

	mean	sd	hdi_2.5%	hdi_97.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_ha
mu_alpha	82.221	324.11	-723.213	779.584	7.605	6.045	1714.0	1203.0	1.13
mu_beta	-0.079	2.662	-1.914	1.751	0.123	0.087	456.0	328.0	1.08
tau_alpha	0.035	0.097	0.0	0.115	0.007	0.005	25.0	75.0	1.14
tau_beta	66.751	146.983	0.016	266.294	24.577	17.522	17.0	351.0	1.16
alpha[0]	99.124	3.279	92.786	104.668	0.731	0.528	24.0	845.0	1.11
beta[0]	0.253	0.118	0.047	0.485	0.025	0.018	25.0	678.0	1.11
beta[1]	-0.002	0.001	-0.004	-0.001	0.0	0.0	27.0	1844.0	1.1
tau_y	0.746	0.417	0.087	1.576	0.056	0.043	107.0	102.0	1.12
x_m[0]	55.263	8.915	41.519	72.177	2.423	1.751	14.0	211.0	1.2
y_m[0]	104.853	1.451	101.633	107.628	0.051	0.036	597.0	497.0	1.11
br2	0.528	0.482	-0.15	0.92	0.018	0.013	107.0	102.0	1.08

Note that $x\_m[0]$ and $y\_m[0]$ are the imputed value for missing x and y respectively. mu_alpha and tau_alpha are the mean and precision of regression coefficient $alpha[0]$, which is the the coefficient for constant term. mu_beta and tau_beta are the mean and precision of regression coefficient $beta$, $beta[0]$ and $beta[1]$ are the regression coefficient for x and x^2 respectively. tau_y is the precision of y

and the squared term respectively. We can see from the table that the coefficient for x is positive, equals 0.118 and its credible set is $(0.047, 0.485)$, not include 0, the coefficient for x^2 is negative, equals -0.002 and its credible set is $(-0.004, -0.001)$, not include 0.

The estimator for missing value of x is given by 55.263 with credible set (41.519,72.177) and the estimator for missing value of y is given by 104.853 with credible set (101.633,107.628)

Exercise 2

Bladder Cancer Data
As instructed in the question, we use the following hierarchical structure, where subscript c indicate the observations that are censored (we do not observe cancer recurrence by the end of the experiment, corresponds to observations where column “observed"=0) and subscript u indicate the observations that are not censored (we already observe cancer recurrence by the end of the experiement, corresponds to observatiosn where column “observed"=1)

$$\begin{aligned}
\beta_0 &\sim Normal(1, \sigma^2 = 10000)\\
\beta_1 &\sim Normal(0, \sigma^2 = 10000)\\
\lambda_c &= e^{\beta_0+\beta_1\cdot x_c}\\
\lambda_u &= e^{\beta_0+\beta_1\cdot x_u}\\
\mu_0 &= e^{-\beta_0}\\
\mu_1 &= e^{-\beta_0-\beta_1}
\end{aligned}$$

The censored data are modeled as exponentials left truncated by the censoring time. The regression results are listed in the following table (imputed values for missing regressors are omitted from the table but they are available as output of the code attached)

	mean	sd	hdi_5%	hdi_95%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
beta0	-3.281	0.187	-3.582	-2.968	0.001	0.001	21870.0	23968.0	1.0
beta1	-0.542	0.306	-1.044	-0.042	0.002	0.002	20193.0	24688.0	1.0
mu0	27.068	5.192	18.757	35.013	0.036	0.026	21870.0	23968.0	1.0
mu1	47.102	11.782	28.823	64.54	0.075	0.054	26434.0	26461.0	1.0
mudiff	20.034	12.938	-0.266	40.716	0.09	0.063	21075.0	25197.0	1.0
effective	0.965	0.183	1.0	1.0	0.001	0.001	26957.0	40000.0	1.0

(a) The value $\mu_1 - \mu_0$ is calculated in row “mudiff", as we can see the 90% credible set is given by $(-0.266, 40.716)$, it is not all positive but very close as the negative coverage is very small compared to the positive coverage.

(b) The posterior probability of hypothesis is displayed in row “effective" (the mean of an indicator variable such that takes value 1 if $\mu_1 > \mu_0$). It's value is 0.965 and the 90% credible set is given by $(1, 1)$

(c) Both (a) and (b) suggest that the chemotherapy is effective: result in (a) shows that it is effective in close to 90% probability and the average number of days it can delay the cancer recurrence is about 20 months. (b) suggest that it yield higher time to cancer recurrence in 96.5% of the cases.

Q1 (a)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pymc as pm
import arviz as az
import copy

if __name__=='__main__':
    np.random.seed(12345)
    data=[(24,102.8),(32,104.5),(48,106.5),(56,107.0),(np.nan,107.1),
          (70,105.1),(72,103.9),(75,np.nan),(80,103.2),(96,102.1)]
    x = np.array([[i[0]] for i in data])
    x = np.concatenate((np.ones((x.shape[0], 1)), x), axis=1)
    y = np.array([i[1] for i in data])
    y=y.copy()
    y=np.nan_to_num(y,nan=-1)
    y=np.ma.masked_values(y,value=-1)
    x=x.copy()
    x=np.nan_to_num(x,nan=-1)
    x=np.ma.masked_values(x,value=-1)
    mp_y=np.where(y.mask==True)[0][0]

    with pm.Model() as m:
        mu_beta = pm.Normal("mu_beta", 0, tau=1e-6)
        tau_beta = pm.Gamma('tau_beta', 0.001, 0.001)
        beta = pm.Normal("beta", mu_beta, tau=tau_beta, shape=x.shape[1])
        x_imputed = pm.TruncatedNormal("x_imputed", mu=np.nanmean([i[0] f
        tau_lld=pm.Gamma('tau_lld',0.001,0.001)
        lld = pm.Normal("lld", beta[0]*x_imputed[:,0]+beta[1]*x_imputed[:
        #r2
        cy=y-y.mean()
        sst=pm.math.dot(cy,cy)
        sse = (10 - 2)/tau_lld
        br2 = pm.Deterministic("br2", 1 - sse / sst)
        trace = pm.sample(10000,tune=1000,cores=4,init="jitter+adapt_diag
    with m:
        print(az.summary(trace, hdi_prob=0.95))
        az.summary(trace,hdi_prob=0.95).to_csv('q1_part1_n.csv')
        #calculate r2
        r2_collector=[]
        all_y,all_tau=np.concatenate(trace.posterior.lld_missing),np.conc
        for i in range(len(all_y)):
            imputed_y_vector=np.concatenate((y[:mp_y],np.array(all_y[i]),
            sst=np.sum((imputed_y_vector-np.mean(imputed_y_vector))**2)
            sse=(10-2)/all_tau[i]
            r2_collector.append(1-sse/sst)
        print('r2 is given by: '+str(np.mean(r2_collector)))
print('part 1 finished')
```

Q1 (b)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pymc as pm
import arviz as az
import copy

if __name__=='__main__':
    np.random.seed(123456)
    data=[(24,102.8),(32,104.5),(48,106.5),(56,107.0),(np.nan,107.1),
                    (70,105.1),(72,103.9),(75,np.nan),(80,103.2),(96,102
    y = np.array([i[1] for i in data]).reshape(len(data), -1).copy()
    y = np.nan_to_num(y, nan=-1)
    y = np.ma.masked_values(y, value=-1)
    x = np.array([i[0] for i in data]).reshape(len(data), -1).copy()
    x = np.nan_to_num(x, nan=-1)
    x = np.ma.masked_values(x, value=-1)

    mp_y=np.where(y.mask==True)[0][0]

    with pm.Model() as m:
        mu_alpha=pm.Normal("mu_alpha",0,tau=1e-6)
        tau_alpha=pm.Gamma("tau_alpha",.001,.001)
        mu_beta=pm.Normal("mu_beta",0,tau=1e-6)
        tau_beta=pm.Gamma("tau_beta",.001,.001)
        alpha=pm.Normal("alpha",mu_alpha,tau=tau_alpha,shape=x.shape[1])
        beta=pm.Normal("beta",mu_beta,tau=tau_beta,shape=x.shape[1]+1)
        tau_lld=pm.Gamma("tau_lld",.001,.001)
        x_imputed=pm.TruncatedNormal("x_imputed",mu=60,sigma=10,lower=0,o
        mu=alpha+beta[0]*x_imputed+beta[1]*x_imputed**2
        lld=pm.Normal("lld",mu,tau=tau_lld,observed=y)
        #r2
        cy = y - y.mean()
        sst = pm.math.dot(cy.squeeze(), cy.squeeze())
        sse = (10 - 3) / tau_lld
        br2 = pm.Deterministic("br2", 1 - sse / sst)
        trace=pm.sample(10000,tune=1000,cores=4,init="jitter+adapt_diag",

    with m:
        print(az.summary(trace, hdi_prob=0.95))
        az.summary(trace,hdi_prob=0.95).to_csv('q1_part2_n.csv')
        # calculate r2
        r2_collector=[]
        all_y,all_tau=np.concatenate(trace.posterior.lld_missing),np.conc
        for i in range(len(all_y)):
            imputed_y_vector=np.concatenate((y[:mp_y].squeeze(),np.array(
            sst=np.sum((imputed_y_vector-np.mean(imputed_y_vector))**2)
        sse=(10-3)/all_tau[i]
        r2_collector.append(1-sse/sst)
    print('r2 is given by: '+str(np.mean(r2_collector)))
print('part 2 finished')
```

Q2

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pymc as pm
import arviz as az
import copy
from pymc.math import exp


if __name__=='__main__':
    data=pd.read_csv('bladderc.csv')
    y=np.array(data['time'])
    x=np.array(data['group'])
    censored_vals=np.array(data['observed'])
    #separate the observed values and the censored values
    observed_mask=censored_vals==1
    censored=y[~observed_mask]
    y_uncensored=y[observed_mask]
    x_censored = x[~observed_mask]
    x_uncensored = x[observed_mask]
    #model
    with pm.Model() as m:
        beta0 = pm.Normal("beta0", 1, tau=0.0001)
        beta1 = pm.Normal("beta1", 0, tau=0.0001)
        lambda_censored = exp(beta0 + beta1 * x_censored)
        lambda_uncensored = exp(beta0 + beta1 * x_uncensored)
        impute_censored = pm.Bound("impute_censored",
                                   pm.Exponential.dist(lam=lambda_censored),lower=c
                          )
        likelihood = pm.Exponential("likelihood",
                                   lam=lambda_uncensored,observed=y_uncensored,shape
                         )
        median0 = pm.Deterministic("mu0", exp(-beta0))
        median1 = pm.Deterministic("mu1", exp(-beta0 - beta1))
        mudiff=pm.Deterministic('mudiff',median1-median0)
        effectice=pm.Deterministic('effective',median0<median1)
        trace = pm.sample(10000, tune=1000, cores=4, init="auto", step=[p
    with m:
        print(az.summary(trace, hdi_prob=0.9))
        az.summary(trace, hdi_prob=0.9).to_csv('q2.csv')
    #probability that it works

    print('finished')
```