FINAL EXAM

ISyE6420

Fall 2020

Released November 24, 12:00pm – due December 8, 11:55pm. This exam is not proctored and not time limited except the due date. Late submissions will not be accepted.

Use of all available electronic and printed resources is allowed except direct communication that violates Georgia Tech Academic Integrity Rules.

Name Chen-Yang(Jim), Liu

Problem	1	2	3	Total
Score	/33	/33	/34	/100

ISyE6420_final

December 9, 2020

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[2]:
```
import arviz as az
import pymc3 as pm
```

1 Vasoconstriction

The data give the presence or absence (y_i = 1 or 0) of vasoconstric- tion in the skin of the fingers following inhalation of a certain volume of air (v_i) at a certain average rate (r_i). Total number of records is 39. The candidate models for analyzing the relationship are the usual logit, probit, cloglog, loglog, and cauchyit models.

[3]:
```
y_1 = np.array([1,1,1,1,1,1,0,0,0,0,0,0,0,1,1,1,1,1,
   0,1,0,0,0,0,1,0,1,0,1,0,1,0,0,1,1,1,0,0,1])
v_1 = np.array([3.7, 3.5, 1.25, 0.75, 0.8, 0.7, 0.6, 1.1, 0.9, 0.9, 0.8, 0.55, 0.
 ↪6, 1.4, 0.75, 2.3, 3.2, 0.85, 1.7,
                 1.8, 0.4, 0.95, 1.35, 1.5, 1.6, 0.6, 1.8, 0.95, 1.9, 1.6, 2.7, 2.
 ↪35, 1.1, 1.1, 1.2, 0.8, 0.95, 0.75, 1.3])
r_1 = np.array([0.825, 1.09, 2.5, 1.5, 3.2, 3.5, 0.75, 1.7, 0.75, 0.45, 0.57, 2.
 ↪75, 3, 2.33, 3.75, 1.64, 1.6, 1.415,
1.06, 1.8, 2, 1.36, 1.35, 1.36, 1.78, 1.5, 1.5, 1.9, 0.95, 0.4, 0.75, 0.3, 1.83,␣
 ↪2.2, 2, 3.33, 1.9, 1.9, 1.625])
```

1.1 (a) Transform covariates v and r as x_1 = log(10×v), x_2 = log(10×r).

[4]:
```
x_1 = np.log(10 * v_1)
x_1
```

[4]:
```
array([3.61091791, 3.55534806, 2.52572864, 2.01490302, 2.07944154,
       1.94591015, 1.79175947, 2.39789527, 2.19722458, 2.19722458,
       2.07944154, 1.70474809, 1.79175947, 2.63905733, 2.01490302,
```

```
        3.13549422, 3.4657359 , 2.14006616, 2.83321334, 2.89037176,
        1.38629436, 2.2512918 , 2.60268969, 2.7080502 , 2.77258872,
        1.79175947, 2.89037176, 2.2512918 , 2.94443898, 2.77258872,
        3.29583687, 3.15700042, 2.39789527, 2.39789527, 2.48490665,
        2.07944154, 2.2512918 , 2.01490302, 2.56494936])
```

[5]:
```
x_2 = np.log(10 * r_1)
x_2
```

[5]:
```
array([2.1102132 , 2.38876279, 3.21887582, 2.7080502 , 3.4657359 ,
       3.55534806, 2.01490302, 2.83321334, 2.01490302, 1.5040774 ,
       1.74046617, 3.314186  , 3.40119738, 3.14845336, 3.62434093,
       2.79728133, 2.77258872, 2.64971462, 2.360854  , 2.89037176,
       2.99573227, 2.61006979, 2.60268969, 2.61006979, 2.87919846,
       2.7080502 , 2.7080502 , 2.94443898, 2.2512918 , 1.38629436,
       2.01490302, 1.09861229, 2.90690106, 3.09104245, 2.99573227,
       3.5055574 , 2.94443898, 2.94443898, 2.78809291])
```

[6]:
```
df = pd.DataFrame({"vasoconstriction": y_1, "air_log": x_1, "rate_log": x_2})
df.head()
```

[6]:

	vasoconstriction	air_log	rate_log
0	1	3.610918	2.110213
1	1	3.555348	2.388763
2	1	2.525729	3.218876
3	1	2.014903	2.708050
4	1	2.079442	3.465736

1.2 (b) Estimate posterior means for coefficients in the logit model. Use noninformative priors on all coefficients.

[7]:
```
names = df.index.values
N = len(names)
dims={
    "air_log": ["developer"],
    "rate_log": ["developer"]
}
```

[8]:
```
'''
pymc3 priors are
default_regressor_prior = Normal.dist(mu=0, tau=1.0E-6) and
default_intercept_prior = Flat.dist()
'''
with pm.Model() as logistic_model:
    pm.glm.linear.GLM(y=df["vasoconstriction"], x= df[["air_log", "rate_log"]],␣
 ↪intercept=True,
```

```
                             family=pm.glm.families.Binomial())
    trace_log = pm.sample(5000, tune=5000, init='adapt_diag')
    posterior_predictive = pm.sample_posterior_predictive(trace_log)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [rate_log, air_log, Intercept]
```

```
Sampling 4 chains for 5_000 tune and 5_000 draw iterations (20_000 + 20_000
draws total) took 54 seconds.
There was 1 divergence after tuning. Increase `target_accept` or reparameterize.
There were 26 divergences after tuning. Increase `target_accept` or
reparameterize.
The acceptance probability does not match the target. It is 0.6210488848956406,
but should be close to 0.8. Try to increase the number of tuning steps.
There were 2 divergences after tuning. Increase `target_accept` or
reparameterize.
The acceptance probability does not match the target. It is 0.7165668834870483,
but should be close to 0.8. Try to increase the number of tuning steps.
The number of effective samples is smaller than 10% for some parameters.
```

[9]:
```
az.summary(trace_log)
```

[9]:

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_mean	\
Intercept	-30.883	10.584	-50.062	-12.367	0.408	0.289	672.0	
air_log	6.269	2.174	2.536	10.256	0.084	0.059	671.0	
rate_log	5.595	2.021	2.366	9.520	0.073	0.051	776.0	

	ess_sd	ess_bulk	ess_tail	r_hat
Intercept	672.0	519.0	322.0	1.01
air_log	671.0	523.0	336.0	1.01
rate_log	776.0	630.0	789.0	1.01

The means of intercept, air_log and rate_log are -30.883, 6.269 and 5.595, respectively.

1.3 (c) For a subject with $v = r = 1.5$, find the probability of vasoconstriction.

$P(vasoconstriction=1) = p $ = $\frac{1}{1+e^(-(intercept+6.461\times air_log+5.719\times rate_log))}$

[42]:
```
prob_v = 1/ (1 + np.exp(-(-30.883 + 6.269 * np.log(1.5 * 10) + 5.595 * np.log(1.
↪5 * 10))))
print("The probability of vasoconstriction with v = r = 1.5: ", prob_v)
```

```
The probability of vasoconstriction with v = r = 1.5:  0.7764865250020363
```

1.4 (d) Compare with the result of probit model. Which has smaller deviance?

[11]:
```
import theano.tensor as tsr
from collections import OrderedDict
```

[12]:
```
with pm.Model() as probit_model:
    # priors
    intercept = pm.Flat("intercept")
    beta0 = pm.Normal("beta0", mu=0, tau=1.0E-6)
    beta1 = pm.Normal('beta1', mu=0, tau=1.0E-6)

    # linear predictor
    theta_p = intercept + beta0 * df["air_log"] + beta1 * df["rate_log"]

    # Probit transform
    def probit_phi(x):
        mu = 0
        sd = 1
        return 0.5 * (1 + tsr.erf((x - mu) / (sd * tsr.sqrt(2))))

    theta = probit_phi(theta_p)


    # likelihood
    y = pm.Bernoulli('y', p=theta, observed=df["vasoconstriction"])

    trace_prob = pm.sample(5000, tune=5000, init='adapt_diag')
    posterior_predictive_prob = pm.sample_posterior_predictive(trace_prob)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using adapt_diag...
ERROR (theano.gof.opt): Optimization failure due to: local_grad_log_erfc_neg
ERROR (theano.gof.opt): node:
Elemwise{true_div,no_inplace}(Elemwise{mul,no_inplace}.0,
Elemwise{erfc,no_inplace}.0)
ERROR (theano.gof.opt): TRACEBACK:
ERROR (theano.gof.opt): Traceback (most recent call last):
  File "/opt/anaconda3/lib/python3.7/site-packages/theano/gof/opt.py", line
2034, in process_node
    replacements = lopt.transform(node)
  File "/opt/anaconda3/lib/python3.7/site-packages/theano/tensor/opt.py", line
6789, in local_grad_log_erfc_neg
    if not exp.owner.inputs[0].owner:
AttributeError: 'NoneType' object has no attribute 'owner'

Multiprocess sampling (4 chains in 4 jobs)
NUTS: [beta1, beta0, intercept]
```

```
Sampling 4 chains for 5_000 tune and 5_000 draw iterations (20_000 + 20_000
draws total) took 55 seconds.
There were 518 divergences after tuning. Increase `target_accept` or
reparameterize.
The acceptance probability does not match the target. It is 0.6682940921512561,
but should be close to 0.8. Try to increase the number of tuning steps.
The acceptance probability does not match the target. It is 0.906901957639155,
but should be close to 0.8. Try to increase the number of tuning steps.
The number of effective samples is smaller than 10% for some parameters.
```

[13]:
```python
model_trace_dict = {"logistic_model":trace_log, "probit_model": trace_prob}
az.compare(model_trace_dict, ic='WAIC', scale="deviance")
```

[13]:
```
                rank     waic   p_waic   d_waic    weight       se      dse  \
probit_model       0   36.107  3.18883        0  0.601672  9.70566        0
logistic_model     1  37.1122   3.9323  1.00522  0.398328  8.57837  1.59664

               warning waic_scale
probit_model      True   deviance
logistic_model    True   deviance
```

Based on the above table, we could see the probit model has smaller deviance than the logistic model.

2 Magnesium Ammonium Phosphate and Chrysanthemums

Walpole et al. (2007) provide data from a study on the effect of magnesium ammonium phosphate on the height of chrysanthemums, which was conducted at George Mason University in order to determine a possible optimum level of fertilization, based on the enhanced vertical growth response of the chrysanthemums. Forty chrysanthemum seedlings were assigned to 4 groups, each containing 10 plants. Each was planted in a similar pot containing a uniform growth medium. An increasing concentration of MgNH4PO4, measured in grams per bushel, was added to each plant. The 4 groups of plants were grown under uniform conditions in a greenhouse for a period of 4 weeks.

The treatments and the respective changes in heights, measured in centimeters, are given in the following table:

[14]:
```python
g50 = np.array([13.2, 12.4, 12.8, 17.2, 13.0, 14.0, 14.2, 21.6, 15.0, 20.0])
g100 = np.array([16.0, 12.6, 14.8, 13.0, 14.0, 23.6, 14.0, 17.0, 22.2, 24.4])
g200 = np.array([7.8, 14.4, 20.0, 15.8, 17.0, 27.0, 19.6, 18.0, 20.2, 23.2])
g400 = np.array([21.0, 14.8, 19.1, 15.8, 18.0, 26.0, 21.1, 22.0, 25.0, 18.2])
```

Solve the problem as a Bayesian one-way ANOVA. Use STZ constraints on treatment effects.

2.1 (a) Do different concentrations of MgNH4PO4 affect the average attained height of chrysanthemums? Look at the 95% credible sets for the differences between treatment effects.

Explanation on choices of priors:

In pymc3, if the priors are too non-informative, the sampling will fail because there are many zeros when the program takes derivatives. Therefore, we released the constraints and chose weakly informative priors on the mean, alpha2, alpha3 and alpha4.

For sigma, the best scenario is using Inversegamma with alpha = 0.0001 and beta = 0.0001. However, the same thing happened. So we released the constraints, too. We changed alpha and beta to 0.01 and 0.01, respectively.

[15]:
```
with pm.Model() as ANOVA:
    mu = pm.Normal("mu", mu=0, sigma=1)
    sigma = pm.InverseGamma("sigma", alpha=0.01, beta=0.01)
    alpha2 = pm.Normal("alpha2", mu=0, sigma=1)
    alpha3 = pm.Normal("alpha3", mu=0, sigma=1)
    alpha4 = pm.Normal("alpha4", mu=0, sigma=1)
    # STZ constraints
    alpha1 = pm.Deterministic("alpha1", -(alpha2+alpha3+alpha4))

    # likelihood
    treatment_50 = pm.Normal("g50", mu=mu+alpha1, sigma=sigma, observed=g50)
    treatment_100 = pm.Normal("g100", mu=mu+alpha2, sigma=sigma, observed=g100)
    treatment_200 = pm.Normal("g200", mu=mu+alpha3, sigma=sigma, observed=g200)
    treatment_400 = pm.Normal("g400", mu=mu+alpha4, sigma=sigma, observed=g400)
```

[16]:
```
with ANOVA:
    alpha1_diff_alpha2 = pm.Deterministic('a1-a2', alpha1 - alpha2)
    alpha1_diff_alpha3 = pm.Deterministic('a1-a3', alpha1 - alpha3)
    alpha1_diff_alpha4 = pm.Deterministic('a1-a4', alpha1 - alpha4)

    alpha2_diff_alpha3 = pm.Deterministic('a2-a3', alpha2 - alpha3)
    alpha2_diff_alpha4 = pm.Deterministic('a2-a4', alpha2 - alpha4)
    alpha3_diff_alpha4 = pm.Deterministic('a3-a4', alpha3 - alpha4)
```

[17]:
```
with ANOVA:
    trace_anova = pm.sample(5000, tune=5000, init='adapt_diag')
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha4, alpha3, alpha2, sigma, mu]
```

```
Sampling 4 chains for 5_000 tune and 5_000 draw iterations (20_000 + 20_000
draws total) took 30 seconds.
```

[18]:

```
az.summary(trace_anova, stat_funcs={"hdi_2.5%": lambda x:np.percentile(x, 2.5),␣
↪"hdi_97.5%": lambda x : np.percentile(x, 97.5)})
```

[18]:

```
          mean     sd  hdi_3%  hdi_97%  mcse_mean  mcse_sd  ess_mean   ess_sd  \
mu       2.427  1.085   0.378    4.434      0.008    0.006   17454.0  16639.0
alpha2   0.057  0.972  -1.689    1.970      0.006    0.007   27958.0  10118.0
alpha3   0.102  0.962  -1.654    1.953      0.006    0.007   27518.0  10682.0
alpha4   0.158  0.964  -1.661    1.958      0.006    0.007   30313.0  10545.0
sigma   16.241  2.138  12.448   20.348      0.016    0.012   17266.0  17266.0
alpha1  -0.317  1.601  -3.270    2.760      0.009    0.011   30737.0  11046.0
a1-a2   -0.374  2.285  -4.663    3.914      0.013    0.016   30231.0  10665.0
a1-a3   -0.419  2.274  -4.599    3.991      0.013    0.015   29452.0  11092.0
a1-a4   -0.475  2.287  -4.822    3.769      0.013    0.016   31153.0  10730.0
a2-a3   -0.045  1.409  -2.618    2.672      0.009    0.010   26664.0  10495.0
a2-a4   -0.101  1.393  -2.652    2.561      0.009    0.010   26810.0  10476.0
a3-a4   -0.055  1.384  -2.607    2.570      0.008    0.010   28759.0  10374.0

        ess_bulk  ess_tail  r_hat  hdi_2.5%  hdi_97.5%
mu       17461.0   14601.0    1.0     0.321      4.553
alpha2   27981.0   15530.0    1.0    -1.856      1.959
alpha3   27485.0   15232.0    1.0    -1.766      1.996
alpha4   30314.0   15572.0    1.0    -1.760      2.033
sigma    17022.0   14461.0    1.0    12.538     20.935
alpha1   30758.0   15896.0    1.0    -3.440      2.841
a1-a2    30239.0   15848.0    1.0    -4.849      4.110
a1-a3    29465.0   15846.0    1.0    -4.925      4.059
a1-a4    31199.0   15740.0    1.0    -4.911      4.056
a2-a3    26658.0   15470.0    1.0    -2.813      2.714
a2-a4    26801.0   15545.0    1.0    -2.809      2.621
a3-a4    28798.0   15198.0    1.0    -2.734      2.654
```

Do different concentrations of MgNH4PO4 affect the average attained height of chrysanthemums?

Based on the 95% credible set (*hdi*2.5% and *hdi*97.5%) in the above table, we could see the differences between alphas. These differences all cover 0 which means different treatments more likely do not affect the average attained height of chrysanthemums.

2.2 (b) Find the 95% credible set for the contrast mu1 - mu2 - mu3+mu4

[19]:

```
with ANOVA:
    mu_differences = pm.Deterministic("123+4", (mu+alpha1) - (mu+alpha2) -␣
↪(mu+alpha3) + (mu+alpha4) )
    trace_mu_differences = pm.sample(5000, tune=5000, init='adapt_diag')
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha4, alpha3, alpha2, sigma, mu]
```

```
Sampling 4 chains for 5_000 tune and 5_000 draw iterations (20_000 + 20_000
draws total) took 32 seconds.
```

[20]:
```python
az.summary(trace_mu_differences, stat_funcs={"hdi_2.5%": lambda x:np.
 ↪percentile(x, 2.5), "hdi_97.5%": lambda x : np.percentile(x, 97.5)})
```

[20]:

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_mean	\
mu	2.431	1.070	0.479	4.483	0.008	0.006	18756.0	
alpha2	0.052	0.971	-1.748	1.897	0.006	0.007	27182.0	
alpha3	0.104	0.978	-1.703	1.947	0.006	0.007	26466.0	
alpha4	0.162	0.971	-1.652	1.978	0.007	0.006	20756.0	
sigma	16.228	2.131	12.434	20.276	0.016	0.011	17872.0	
alpha1	-0.318	1.634	-3.391	2.697	0.010	0.011	26951.0	
a1-a2	-0.370	2.324	-4.827	3.944	0.014	0.016	27645.0	
a1-a3	-0.423	2.329	-4.769	3.881	0.014	0.016	27860.0	
a1-a4	-0.481	2.315	-4.933	3.740	0.015	0.015	24277.0	
a2-a3	-0.052	1.392	-2.641	2.575	0.009	0.010	26296.0	
a2-a4	-0.110	1.395	-2.741	2.486	0.009	0.010	22736.0	
a3-a4	-0.058	1.404	-2.758	2.511	0.009	0.010	21936.0	
mu1-mu2-mu3+mu4	-0.312	2.730	-5.334	4.952	0.017	0.019	27359.0	

	ess_sd	ess_bulk	ess_tail	r_hat	hdi_2.5%	hdi_97.5%
mu	17889.0	18746.0	16089.0	1.0	0.331	4.497
alpha2	9812.0	27181.0	15304.0	1.0	-1.862	1.946
alpha3	10612.0	26470.0	14907.0	1.0	-1.800	2.007
alpha4	11464.0	20765.0	15920.0	1.0	-1.721	2.068
sigma	17872.0	17655.0	14169.0	1.0	12.573	20.920
alpha1	10936.0	26919.0	15512.0	1.0	-3.510	2.845
a1-a2	10479.0	27633.0	15421.0	1.0	-4.955	4.153
a1-a3	10827.0	27869.0	15191.0	1.0	-4.938	4.088
a1-a4	11300.0	24279.0	15715.0	1.0	-5.008	4.065
a2-a3	10330.0	26357.0	15717.0	1.0	-2.786	2.668
a2-a4	10619.0	22738.0	15147.0	1.0	-2.866	2.598
a3-a4	10874.0	21939.0	15360.0	1.0	-2.786	2.702
mu1-mu2-mu3+mu4	10512.0	27357.0	14806.0	1.0	-5.659	5.032

[43]:
```python
print("95% ccredible set of mu1-mu2-mu3+mu4 : [{} , {}]".format(-5.659, 5.032) )
```

```
95% credible set of mu1-mu2-mu3+mu4 : [-5.659 , 5.032]
```

This scenario is to test $H_0 : \mu_1 + \mu_4 = \mu_3 + \mu_2 = 0$ VS. $H_1 : \mu_1 + \mu_4 \neq \mu_2 + \mu_3$

$(\mu_1 + \mu_4) - (\mu_2 + \mu_3)$ still covers 0. So two treatments which are added together more likely do not show difference of heights.

3 Hocking–Pendleton Data

This popular data set was constructed by Hocking and Pendelton (1982) to illustrate influential and outlier observations in regression. The data are organized as a matrix of size 26 × 4; the predictors x_1 , x_2 , and x_3 are the first three columns, and the response y is the fourth column. The data are given in hockpend.dat

[22]:
```
df3 = pd.read_csv("hockpend.dat", sep="\t", header=None).rename(columns={0:"x1",␣
↪1:"x2", 2:"x3", 3: "y"})
```

[23]:
```
df3.head()
```

[23]:

	x1	x2	x3	y
0	12.980	0.317	9.998	57.702
1	14.295	2.028	6.776	59.295
2	15.531	5.305	2.947	55.166
3	15.133	4.738	4.201	55.767
4	15.342	7.038	2.053	51.722

3.1 (a) Fit the linear regression model with the three covariates, report the parameter estimates and Bayesian R^2

[24]:
```
with pm.Model() as linear_model:
    pm.glm.linear.GLM(y=df3["y"], x= df3[["x1", "x2", "x3"]], intercept=True,
                      family=pm.glm.families.Normal())
    trace_lm = pm.sample(5000, tune=5000, init='adapt_diag')
    posterior_predictive_lm = pm.sample_posterior_predictive(trace_lm)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [sd, x3, x2, x1, Intercept]
```

```
Sampling 4 chains for 5_000 tune and 5_000 draw iterations (20_000 + 20_000
draws total) took 90 seconds.
There were 249 divergences after tuning. Increase `target_accept` or
reparameterize.
There were 125 divergences after tuning. Increase `target_accept` or
reparameterize.
There were 1695 divergences after tuning. Increase `target_accept` or
reparameterize.
The acceptance probability does not match the target. It is 0.4309795324404258,
but should be close to 0.8. Try to increase the number of tuning steps.
There were 26 divergences after tuning. Increase `target_accept` or
reparameterize.
The rhat statistic is larger than 1.05 for some parameters. This indicates
```

```
slight problems during sampling.
The estimated number of effective samples is smaller than 200 for some
parameters.
```

---

[25]:
```python
az.summary(trace_lm)
```

[25]:
```
             mean     sd  hdi_3%  hdi_97%  mcse_mean  mcse_sd  ess_mean  \
Intercept   9.303  9.702  -9.351   25.252      1.588    1.132      37.0
x1          3.399  0.547   2.505    4.487      0.085    0.060      42.0
x2         -1.461  0.251  -1.874   -0.972      0.027    0.019      86.0
x3          0.318  0.280  -0.162    0.840      0.040    0.029      48.0
sd          2.638  0.438   1.847    3.406      0.043    0.031     102.0

           ess_sd  ess_bulk  ess_tail  r_hat
Intercept    37.0      37.0     784.0   1.07
x1           42.0      41.0    1411.0   1.06
x2           86.0      86.0     745.0   1.03
x3           48.0      47.0     644.0   1.06
sd          102.0      76.0      47.0   1.03
```

There are five estimated parameters. The estimated coefficients of the intercept, x_1, x_2, x_3 and standard deviation are 9.303, 3.399, -1.461, 0.318 and 2.638, respectively.

[26]:
```python
r2_scores = []
y_true = df3["y"]
for i in range(len(posterior_predictive_lm["y"])):
    y_pred = posterior_predictive_lm["y"][i]
    r2 = az.r2_score(y_true, y_pred)[0]
    r2_scores.append(r2)
print("Mean of BR2: ", np.mean(r2_scores))
print("Standard deviation of BR2: ", np.std(r2_scores))
```

```
Mean of BR2:  0.7605379992456223
Standard deviation of BR2:  0.061840177107345136
```

3.2 (b) Is any of the 26 observations influential or outlier (in the sense of CPO and cumulative)?

3.2.1 Cumulative

We use Cumulative to check whether each data point is an outlier. The concpt is to use the distribution defined in each iteration and then we will check where the observed value locates in its cumulative distribution. So after simulation, we could see the means of each data point. If the data point is close to 1 or 0, it is more likely to conclude it is an outlier.

Based on the above samples, our model is:

$y_i = \beta_0 + \beta_1 * x_{i1} + \beta_2 * x_{i2} + \beta_3 * x_{i3} + \epsilon_i,\ where\ \epsilon_i \sim N(0, \sigma^2),\ i = 1, 2, \ldots, 26$

And, we define each coefficient except β_0 as $N(0, 10^{-5})$. As for the intercept, we use the Flat prior.

$y_i \sim N(\beta_0 + \beta_1 * x_{i1} + \beta_2 * x_{i2} + \beta_3 * x_{i3}, \eta = sd^2)$

[27]:
```
from scipy.stats import norm
```

[28]:
```
df_trace = pm.backends.tracetab.trace_to_dataframe(trace_lm)
df_trace.head()
```

[28]:

	Intercept	x1	x2	x3	sd
0	5.601625	3.622686	-1.233575	0.274603	2.855812
1	7.631317	3.312460	-1.100052	0.625191	2.919152
2	5.730193	3.375129	-0.908699	0.620648	2.695725
3	5.979099	3.553475	-1.339585	0.523389	3.046800
4	3.279882	3.805025	-1.496413	0.331836	2.304360

[29]:
```
cuy = np.zeros(26)

for i in range(df_trace.shape[0]):
    intercept = df_trace.iloc[i, 0]
    b1 = df_trace.iloc[i, 1]
    b2 = df_trace.iloc[i, 2]
    b3 = df_trace.iloc[i, 3]
    sd = df_trace.iloc[i, 4]
    for j in range(26):
        obs = df3["y"][j]
        cuy[j] += norm.cdf(obs, loc=intercept + b1 * df3["x1"][j] + b2 *␣
 ↪df3["x2"][j] + b3 * df3["x3"][j], scale=sd)
```

[30]:
```
outlier_check = cuy / df_trace.shape[0]
```

[31]:
```
outlier_check
```

[31]:
```
array([0.70828015, 0.79224846, 0.48248859, 0.5920942 , 0.4845708 ,
       0.72007163, 0.69536282, 0.39574399, 0.66401189, 0.69294881,
       0.22093436, 0.52572268, 0.62230292, 0.3701755 , 0.00388224,
       0.36372404, 0.94331224, 0.03018092, 0.30486989, 0.77054044,
       0.53216637, 0.68297244, 0.56086072, 0.5843513 , 0.47637694,
       0.40871561])
```

[32]:
```
outlier_check[outlier_check<0.1]
```

[32]:
```
array([0.00388224, 0.03018092])
```

[33]:
```
outlier_check[outlier_check>0.9]
```

[33]:
```
array([0.94331224])
```

[45]:
```
np.where(outlier_check==0.0038822393888119306)[0]+1
```

[45]:
```
array([15])
```

[46]:
```
np.where(outlier_check== 0.030180919979533604)[0]+1
```

[46]:
```
array([18])
```

[47]:
```
np.where(outlier_check==0.9433122366137262)[0]+1
```

[47]:
```
array([17])
```

Originally, I chose the $\alpha = 0.2$ and found the 15th, 17th and 18th observations are more likely outliers since the means of these observations are either too close to 0 and 1.

If we choose the smaller confidence level, like we change α to $\alpha = 0.05$, the obvious outlier is the 15th data point.

3.3 (c) Find the mean response and prediction response for a new observation with covariates $x_1^* = 10$, $x_2^* = 5$, and $x_3^* = 5$. Report the corresponding 95% credible sets

[37]:
```
new_df = pd.DataFrame({"x1":[10], "x2": [5], "x3": [5], "y":[4.813 + 3.648 * 10␣
↪+ (-1.400) * 5 + 0.420 * 5]})
```

[38]:
```
with pm.Model() as lm_pred:
    pm.glm.linear.GLM(y=new_df["y"],
                      x=new_df[["x1", "x2", "x3"]], intercept=True,
                      priors={'Intercept':pm.Normal.dist(mu=0, sigma=10.0)},
                      family=pm.glm.families.Normal())
    trace_lm_p = pm.sample(5000, tune=5000, init='adapt_diag')
    df = pm.trace_to_dataframe(trace_lm, include_transformed=True)
    ppc = pm.sample_posterior_predictive(trace=df.
↪to_dict('records'),samples=len(df))
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [sd, x3, x2, x1, Intercept]
```

```
Sampling 4 chains for 5_000 tune and 5_000 draw iterations (20_000 + 20_000
draws total) took 39 seconds.
There were 2345 divergences after tuning. Increase `target_accept` or
reparameterize.
The acceptance probability does not match the target. It is 0.3403953364767104,
but should be close to 0.8. Try to increase the number of tuning steps.
There were 1798 divergences after tuning. Increase `target_accept` or
```

```
reparameterize.
The acceptance probability does not match the target. It is 0.44437999849911075,
but should be close to 0.8. Try to increase the number of tuning steps.
There were 3543 divergences after tuning. Increase `target_accept` or
reparameterize.
The acceptance probability does not match the target. It is 0.05906703653418209,
but should be close to 0.8. Try to increase the number of tuning steps.
There were 2760 divergences after tuning. Increase `target_accept` or
reparameterize.
The acceptance probability does not match the target. It is 0.30560815998471047,
but should be close to 0.8. Try to increase the number of tuning steps.
The rhat statistic is larger than 1.4 for some parameters. The sampler did not
converge.
The estimated number of effective samples is smaller than 200 for some
parameters.
```

Prediction response:

This means the result is derived based on inputs and coefficients.

[1]:
```
4.813 + 3.648 * 10 + (-1.400) * 5 + 0.420 * 5
```

[1]:
```
36.393000000000001
```

Mean response:

This means that based on each iteration, we get estimated coefficients and then have multiple response values. We take expectation on these values.

[40]:
```
ppc["y"].mean()
```

[40]:
```
37.576426132208155
```

[41]:
```
az.hdi(ppc["y"], hdi_prob=0.95)
```

```
/opt/anaconda3/lib/python3.7/site-packages/arviz/stats/stats.py:487:
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,
```

[41]:
```
array([[29.97973519, 44.50649613]])
```

The prediction response is 36.393000000000001.

The mean response is 37.576426132208155 and the 95% credible set is between 29.97973519 and 44.50649613.