CS146_Assignment_4

November 26, 2020

1 CS146 | Assignment 4

1.1 Posterior Predictive Checks | Prof. Scheffler

1.1.1 Anirudh Nair

Link to github: https://github.com/anirudhnair42/CS146_Assignments/blob/master/Assignment%204/CS146

For the non-hierarchical model and the control group half of the data set, choose a test statistic to show that the non-hierarchical model does not explain the control group data well. You should generate posterior data samples from this model and compare the test statistic of the real data to the distribution under your samples from the posterior. Note that the posterior distribution for this model is already provided in the notebook, so you don't have to calculate it. Submit your test statistic function, your code for generating samples from the posterior and computing the test statistic, and a histogram analogous to Gelman Figure 6.4 (reproduced below). Also, calculate and provide your p-value. Since the test statistic is supposed to show a deficiency of the model the p-value should be less than 0.05 or greater than 0.95.

#PCW Description Excerpt

We consider the eczema medical trial data set again. This time we will compare which of 2 models explain the observed data best.

- Model 1: All studies have the same probability of success.

Study	Treatment group	Control group
Di Rienzo 2014	20 / 23	9 / 15
Galli 1994	10 / 16	11 / 18
Kaufman 1974	13 / 16	4 / 10
Qin 2014	35 / 45	21 / 39
Sanchez 2012	22 / 31	12 / 29
Silny 2006	7 / 10	0 / 10
Totals	107 / 141	57 / 121

Model 1:

- For each group (treatment and control), all 6 studies have the same fixed, but unknown, probability of success, $\theta_t, \theta_c \in [0, 1]$.
- The data follow a binomial distribution in each study, conditioned on the probability of success — θ_t for treatment or θ_c for control.

- The priors over θ_t and θ_c are uniform.

These assumptions lead to the following model.

- Likelihood: $\prod_{i=1}^{6} \text{Binomial}(s_i \mid \theta, n_i)$, where s_i is the number of successful recoveries, f_i is the number of failures (did not recover), and $n_i = s_i + f_i$ the number of patients.
- Prior: $\text{Beta}(\theta \mid 1, 1)$ for both θ_t and θ_c.
- Posterior for treatment group: $\text{Beta}(\theta_t \mid 108, 35)$.
- Posterior for control group: $\text{Beta}(\theta_c \mid 58, 65)$.

Since we have closed-form solutions for the posteriors, we can calculate the marginal likelihood by rearranging Bayes' equation: (marginal likelihood) = (likelihood) x (prior) / (posterior).

$$P(\text{data}) = \left[\prod_{i=1}^{6} \text{Binomial}(s_i \mid \theta, n_i)\right] \text{Beta}(\theta \mid \alpha_0, \beta_0) \,/\, \text{Beta}(\theta \mid \alpha_1, \beta_1)$$

where $\alpha_0 = 1$ and $\beta_0 = 1$ are the parameters of the prior, and α_1 and β_1 are the parameters of the posterior beta distribution.

Since all factors involving θ cancel out, we are just left with the normalization constants of the likelihood, the prior and the posterior:

$$\begin{aligned} P(\text{data}) &= \left[\prod_{i=1}^{6} \binom{s_i + f_i}{s_i}\right] \frac{\text{B}(\alpha_1, \beta_1)}{\text{B}(\alpha_0, \beta_0)} \\ &= \left[\prod_{i=1}^{6} \frac{1}{(s_i + f_i + 1)\text{B}(s_i + 1, f_i + 1)}\right] \frac{\text{B}(\alpha_1, \beta_1)}{\text{B}(\alpha_0, \beta_0)} \end{aligned}$$

We usually compute the log of the marginal likelihood since the results can vary over many orders of magnitude.

A word on notation in the derivation above:

- The beta *distribution* is written as $\text{Beta}(\theta \mid \alpha, \beta)$.
- The beta *function* is written as $B(\alpha, \beta)$. B is the Greek letter *capital beta*.
- The beta function is part of the normalization constant of the beta distribution.

This is similar to the gamma distribution and the gamma function, where

- the distribution is written as $\text{Gamma}(x \mid \alpha, \beta)$,
- the function is written as $\Gamma(\alpha)$,
- the gamma function is part of the normalization constant of the gamma distribution.

A word on simplifying the expression in the derivation above:

Just as the gamma function is related to factorials, the beta function is related to combinations:

- $n! = \Gamma(n + 1)$ for integer n.
- $\binom{n}{k} = ((n + 1) \cdot B(n - k + 1, k + 1))^{-1}$

The beta function can also be written in terms of the gamma function:

- $B(x,y) = \Gamma(x)\,\Gamma(y)\ /\ \Gamma(x+y)$

[1]:
```
#importing the required libraries

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
sns.set()
```

[2]:
```
#defining the success cases and the total data points in the control group data
s_control = [9,11,4,21,12,0]
t_control = [15,18,10,39,29,10]
```

[10]:
```
#defining a function for the test statistic for explaining how the drawn samples␣
 ↪explain the control group data
def test_stat(samples):
  for i in range(len(t_control)):
    success = [samples[i]/t_control[i] for i in range(len(t_control))]
  stat = max(success) - min(success)
  return stat
```

[16]:
```
#calculating the test statistic for the actual data
test_stat_1 = test_stat(s_control)
print("Original test statistic is :", test_stat_1)
```

```
Original test statistic is : 0.6111111111111112
```

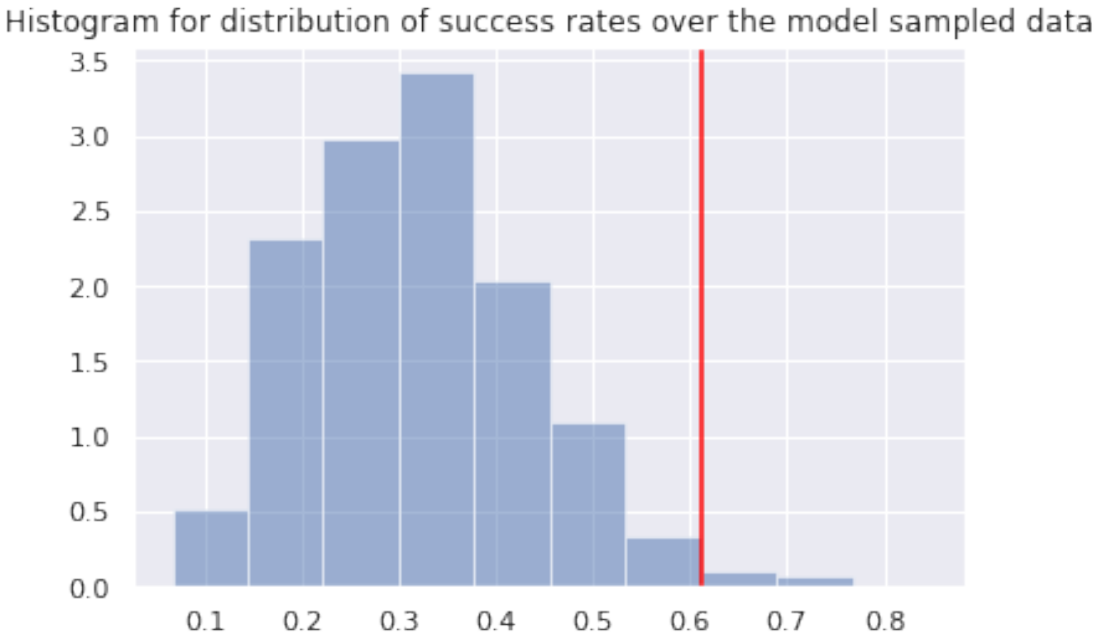
[13]:
```
#drawing samples from the posterior for the control group
posterior = stats.beta.rvs(a = 58, b = 65, size = 1000)

#strong the generated test statistic values from our function
test_stats = []

r_control = []
#recreating control group data using the chosen posterior distribution
for i in posterior:
  r_control = [int(stats.binom.rvs(t_control[j], i, size = 1)) for j in␣
 ↪range(len(t_control))]
  test_stats.append(test_stat(r_control))
```

[17]:
```
#test stats from the samples generated from the non-hierarchial model
test_stat_2 = np.mean(test_stats)
print("Test statistic for the model is :", test_stat_2)
```

```
Test statistic for the model is : 0.3191739463601532
```

[18]:
```
#plotting the histogram analogous to Gelman Figure 6.4
plt.hist(test_stats, alpha = 0.5, density = True)
plt.title("Histogram for distribution of success rates over the model sampled␣
 ↪data")
plt.axvline(test_stat_1, color='red')
```

[18]:
```
<matplotlib.lines.Line2D at 0x7f6ab69702e8>
```



[20]:
```
pvalue = sum([1 for i in test_stats if i > test_stat_1])/len(test_stats)
print('The value for our model comes out to be :', pvalue)
```

```
The value for our model comes out to be : 0.014
```