ETC5543 Report - The conjBayes R package development

Pranali Angne, Student ID: 32355068

2022-11-16

1 Abstract

The aim of this internship was to develop a whole new package named conjBayes for the unit ETC4541/ETC5410, Bayesian inference and data analysis. The *conjBayes* packages provides functions for updating conjugate Bayesian prior distributions to the posterior distribution. All the functions have been specially formatted using the *roxygen* format above the source code for each function. Then these defined functions in this package have been added to the corresponding test files converting the existing unit tests cases to *testthat* format. During the process of developing *conjBayes* package, a few R packages were applied, including *usethis*, *devtools*, *roxygen2*, *pkgdown*, and *testthat*. I'll be reviewing my work on the *conjBayes* package in this report. I discovered my weaknesses during the internship and got better at solving challenging issues when developing or upgrading R packages.

2 Background and motivation

The *conjBayes* is an R package designed to help students (understand) and apply Conjugate Bayesian models. The project my supervisor had in mind involves creating a package of R programs that she has written to do Markov chain Monte Carlo algorithms to fit various Bayesian statistical models. In particular my supervisor was looking for a package that do

- Beta/ Binomial (Beta/ Bernoulli)
- Gamma Exponential
- Gamma/Poisson
- Normal/Normal
- Normal Inverse Gamma/ Normal

She also wanted to have a vignettes. Therefore, my job was to help my supervisor develop the *conjBayes* package and help her converting the format of documentation and unit tests, as well as creating the vignette.

I believe that this internship gives me the chance to put my knowledge to use in a difficult, real-world setting. It might allow me to put my knowledge to use, identify my weaknesses, and enhance my capacity for handling challenging R-package issues.

Furthermore, Conjugate Bayesian prior distributions can be updated to the posterior distribution using the functions provided by the *conjBayes* package, because it was designed to help students (understand) and apply Conjugate Bayesian models and, I hope to help my supervisor to develop this package and provide it to more users. Overall, those are my motivations that joining the internship and working on developing the *conjBayes* package as this my first time working for a internship which built a lot of confidence and this will definitely help me for my future.

3 The development of the *conjBayes* package

The name of this package is *conjBayes* which offers "Functions for updating conjugate priors for Bayesian analysis". Currently, the *conjBayes* is being developed to version 1.0.0. The author for this package is **Dr. Catherine Forbes** . The package **conjBayes** is an R package. The *conjBayes* packages provides functions for updating conjugate Bayesian prior distributions to the posterior distribution under GPL (>= 3) license.

I primarily assist my supervisor during the internship with increasing the number of unit tests, converting the original unit test cases format to the *testthat* format, and documenting functions using the *roxygen* format (or converting the original format to the *roxygen* format). I also assist with creating vignettes. While *conjBayes* is being developed, running the R-CMD-check, debugging, and submitting the modification to Github for reproducibility are also important and routine. Additionally, to build the package, the R-CMD-check, *pkgdown*, and test-coverage have all been posted to Github Actions.

3.1 Unit test format transformation

To carry out unit testing. The *testthat* package was used and create new unit tests for each of the six tested functions as the functions were being created. All the functions have been tested and they work perfectly fine.

3.2 Vignette of *conjBayes* package

There is a vignette of *conjBayes* in the version 1.0.0. My supervisor generated the function, therefore, I helped her to create a vignette/ directory, ading necessary dependencies to the description file and drafting the vignette. vignettes.

The vignette is made to make it easier for users to comprehend both how to use the package and its key features. Therefore, there are four sections the vignette that contain the explanation of the two most important functions in the *conjBayes* package: 1. Introduction 2. Gamma prior for Poisson data model 3. Beta prior for Bernoulli data model 4. and an example of Zombie Apocalypsein

1. Introduction The initial section is the introductory section which explains The conjBayes packages provides functions for updating conjugate Bayesian prior distributions.

When data are modelled conditional on an unknown parameter, say θ, the marginal distribution assumed for θ is known as the **prior** distribution and the **posterior** distribution is the conditional distribution for θ given the observed data. This posterior distribution is determined by Bayes theorem. When the posterior distribution can be calculated in closed form, and belongs to the same distributional family as the prior distribution, then the prior is called a **conjugate prior** for the given data model.

Each conjBayes function reports the hyperparameters of the relevant prior distribution corresponding to the prior and model combination as indicated by the function name. Let's look at an example to see how the functions work.

2. Gamma prior for Poisson data model:

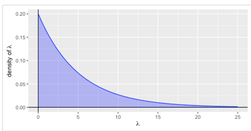
A part of the vignette is shown in Figure 3.1:

```
# Before we use the GammaPoisson function we need to library 'conjBayes' package.

library(tidyverse)
library(conjBayes)

lambda <- seq(0.01, 25, 0.01)
densprior <- dgamma(x = lambda, shape = 1, rate = 0.2)
pt <- tibble(lambda, densprior)

p1 <- pt %>% ggplot(aes(x = lambda, y = densprior)) +
  geom_line(color = "blue") +
  geom_vline(xintercept = 0) +
  geom_hline(xintercept = 0) +
  xlab(expression(lambda)) +
  ylab(expression(paste("density of ",lambda))) +
  geom_area(fill = "blue", alpha=0.3)
p1
```



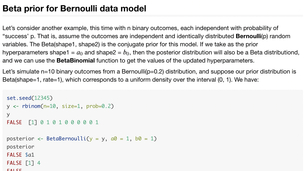
Gamma(shape = 1, rate = 0.2) prior density function for Poisson model.

Figure 3.1: Vignette

For the Gamma prior for Poisson data model we considered the dataset `discoveries`, which contains the number of "great" inventions and scientific discoveries in each year of the 100 year period 1860 to 1959. And after updating the hyper parameters prior probabilities can be confirmed. We use the **GammaPoisson** function to update the Gamma distribution hyperparameters, given the data values in `discoveries`.

3. Beta prior for Bernoulli data model:

Next section was about Beta prior for Bernoulli data model. Meanwhile, in the section introducing the *BetaBernoulli* function, a complete process including the trial design, data generation and data analysis has been created to guide users to apply this function. Besides, this section also explains three important arguments to help users better use the function to get the appropriate result. The application of the *BetaBernoulli* function is shown in Figure ??:



Beta prior for Bernoulli data model

Let's consider another example, this time with n binary outcomes, each independent with probability of "success" p. That is, assume the outcomes are independent and identically distributed **Bernoulli**(p) random variables. The Beta(shape1, shape2) is the conjugate prior for this model. If we take as the prior hyperparameters shape1 = a_0 and shape2 = b_0, then the posterior distribution will also be a Beta distributionb, and we can use the **BetaBinomial** function to get the values of the updated hyperparameters.

Let's simulate n=10 binary outcomes from a Bernoulli(p=0.2) distribution, and suppose our prior distribution is Beta(shape=1, rate=1), which corresponds to a uniform density over the interval (0, 1). We have:

```
set.seed(12345)
y <- rbinom(n=10, size=1, prob=0.2)
y
FALSE  [1] 0 1 0 1 0 0 0 0 0 1

posterior <- BetaBernoulli(y = y, a0 = 1, b0 = 1)
posterior
FALSE 5a1
FALSE [1] 4
FALSE
FALSE 5b1
FALSE [1] 8
```

A plot of the prior and posterior densities are shown below.

```
pp <- seq(0.001, 0.999, 0.01)
densprior <- dbeta(x = pp, shape1 = 1, shape2 = 1)
denspost <- dbeta(x = pp, shape1 = posterior5a1, shape2 = posterior5b1)
pt2 <- tibble(pp, densprior, denspost)

p2 <- pt2 %>% ggplot(aes(x = pp, y = densprior)) +
  geom_line(color = "blue") +
  geom_vline(xintercept = 0) +
  geom_hline(xintercept = 0) +
  xlab("p") +
  ylab("density of p") +
  geom_area(fill = "blue", alpha=0.3)

p2 + geom_line(data = pt2, aes(x = pp, y = denspost), color="magenta") +
  geom_area(data = pt2, aes(x = pp, y = denspost),fill = "magenta", alpha=0.3) +
  annotate("text", label="prior = Beta(shape1 = 1, shape2 = 1)", x=0.8, y=1.2, color="blue") +
  annotate("text", label="posterior = Beta(shape = 4, rate = 8)", x=0.65, y=2.5, color="magenta") +
  geom_vline(xintercept = mean(y)) +
  annotate("text", label="MLE (sample proportion of successes): 0.3", x=0.6, y=0.2)
```

Figure 3.2: The Example of BetaBernoulli Function

4. And the last is providing an example of the Zombie Apocalypse:

A final (and fun!) example is described in the **RPub** "Fundamentals of Bayesian Data Analysis in [RPubs] (https://rpubs.com/kklinares/443186). We want to know the probability we have found a vaccine to cure a zombie outbreak.

Below we use the code provided in the RPub to generate 13 vaccine outcomes, where a success corresponds to a cure.

```
set.seed(789)
p_success <- .15 # probability of success
n_zombies <- 13 # trails

# simulate data
data <- c()
for(zombie in 1:n_zombies){
  # save if prop is between 0 & 1 and greater than our prob. of success
  data[zombie] <- runif(1, min=0, max=1) < p_success
}

data <- as.numeric(data)
data
FALSE  [1] 0 1 1 0 0 1 0 0 0 0 0 0 0
mean(data) # our posterior is close to our theorized probability of success of .15
FALSE [1] 0.231
sum(data) # number of successes
FALSE [1] 3
```

Figure 3.3: Zombie Apocalypsein Example

```
set.seed(789)
p_success <- .15 # probability of success
n_zombies <- 13 # trails

# simulate data
data <- c()
for(zombie in 1:n_zombies){
  # save if prop is between 0 & 1 and greater than our prob. of success
  data[zombie] <- runif(1, min=0, max=1) < p_success
}

data <- as.numeric(data)
data
FALSE  [1] 0 1 1 0 0 1 0 0 0 0 0 0 0
mean(data) # our posterior is close to our theorized probability of success of .15
FALSE [1] 0.231
sum(data) # number of successes
FALSE [1] 3
```

Figure 3.4: The Example of Zombie Apocalypsein

Since we have 3 successes out of 13 independent trials, we can use the **BetaBinomial** function from the **conjBayes** package. This time let's assume our prior is Beta(shape1 = 3, shape2 = 97), which corresponds to assuming the expected probability of success (that the zombie outbreak vaccine will work on any individual) is 0.03 and (before we see any data...) $\Pr(p \leq 0.0822077) = 0.99$. Then we have

```
#pp <- seq(0.001, 0.999, 0.01)
densprior <- dbeta(x = pp, shape1 = 3, shape2 = 97)
posterior <- BetaBinomial(n=mean(data), ab=3, bb=97)
posterior
FALSE 5a1
FALSE [1] 3.23
FALSE
FALSE 5b1
FALSE [1] 107

denspost <- dbeta(x = pp, shape1 = posterior5a1, shape2 = posterior5b1)
pt3 <- tibble(pp, densprior, denspost)

p3 <- pt3 %>% ggplot(aes(x = pp, y = densprior)) +
  geom_line(color = "blue") +
  geom_vline(xintercept = 0) +
  geom_hline(xintercept = 0) +
  xlab("p") +
  ylab("density of p") +
  geom_area(fill = "blue", alpha=0.3)

p3 + geom_line(data = pt2, aes(x = pp, y = denspost), color="magenta") +
  geom_area(data = pt2, aes(x = pp, y = denspost),fill = "magenta", alpha=0.3) +
  annotate("text", label="prior = Beta(shape1 = 1, shape2 = 1)", x=0.8, y=25, color="blue") +
  annotate("text", label="posterior = Beta(shape = 4, rate = 8)", x=0.71, y=3, color="magenta") +
  geom_vline(xintercept = mean(y)) +
  annotate("text", label="MLE (sample proportion of successes): 0.3", x=0.6, y=10)
```

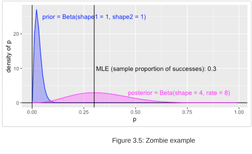


Figure 3.5: Zombie example

4 Conclusion

In conclusion, writing a package is a complicated job, especially when we need to develop it from scratch. However, this is also a good opportunity for me to apply my learning and practice my knowledge, as well as learn some new knowledge. Besides, resolving the errors and the debugging process was quite interesting to learn. I will say I could learn the most out of the internship and extremely thankful towards my supervisor who is so much patient with abundance of knowledge and provided me with first hand experience through her work.

Overall, this is a good opportunity for me and I think it could help my future career. Lastly I would like to thank Monash University especially Prof. Di Cook and Prof. Rob Hyndman who gave me an opportunity by providing me this internship.

5 References

https://rpubs.com/kklinares/443186

https://r-pkgs.org