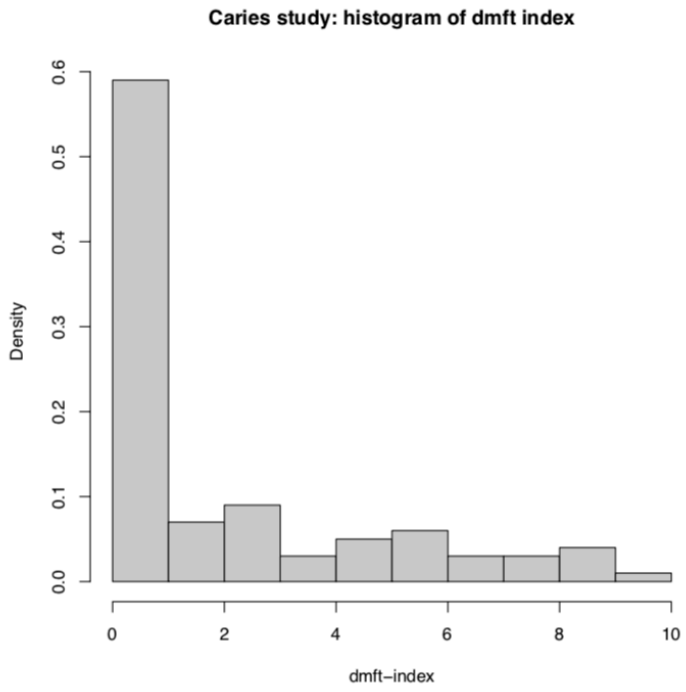
Workshop 2: Solutions

1. (i) Hereby I will let y=dmft. The first sensible check is to look at the histogram of the data:



As can be observed the value 0 has a high frequency when compared to the remaining values. A further check is to compute the mean and the variance of the data (for the Poisson distribution we know that the mean should be equal to the variance). We obtain the values listed below, clearly showing that the data is overdispersed.

> mean(y)
[1] 2.17
> var(y)
[1] 8.102121

As the authors state in the book: "While the Poisson distribution is usually the first choice to describe the distribution of counts, in medical applications it is often not the best choice. For the Poisson distribution, the counts represent the sum of independent events that happen with a constant average. The dmft-index is the sum of binary responses expressing the caries experience in each of the 20 primary teeth. However, cavities in the same mouth are correlated. This leads to (Poisson-) overdispersion, which means that the variance is larger than the mean." Therefore, the Poisson distribution is possibly not the best option to model this dataset. Possibly, a zero inflated Poisson distribution or a negative binomial distribution would be better options. For the sake of simplicity, we proceed with the Poisson distribution.

(ii) The likelihood is Poisson and θ is assigned a Gamma(a, b) prior distribution. We have

$$
\begin{aligned}
p(\theta \mid \mathbf{y}) &\propto f(\mathbf{y};\theta)p(\theta) \\
&= \left\{ \prod_{i=1}^{n} \frac{e^{-\theta}\theta^{y_i}}{y_i!} \right\} \frac{b^a}{\Gamma(a)} \theta^{a-1} e^{-b\theta} \\
&\propto e^{-n\theta}\theta^{\sum_{i=1}^{n} y_i} \theta^{a-1} e^{-b\theta} \\
&= \theta^{a+\sum_{i=1}^{n} y_i - 1} e^{-\theta(b+n)},
\end{aligned}
$$

i.e., $\theta \mid \mathbf{y} \sim \text{Gamma}(a + \sum_{i=1}^{n} y_i, b + n)$. For this particular dataset we have $a = 3$, $b = 1$, $n = 100$, and $\sum_{i=1}^{n} y_i = 217$, thus leading to $\theta \mid \mathbf{y} \sim \text{Gamma}(220, 101)$. For a $\text{Gamma}(a_1, b_1)$ distribution, we know that its mean is $\frac{a_1}{b_1}$ and its variance is $\frac{a_1}{b_1^2}$, and thus

$$
E(\theta \mid \mathbf{y}) = \frac{220}{101} \approx 2.178, \qquad \sqrt{\text{var}(\theta \mid \mathbf{y})} = \sqrt{\frac{220}{101^2}} \approx 0.147.
$$

Using the function qgamma in R, to compute the 2.5% and 97.5% quantiles, we obtain that a 95% credible interval for θ is (1.900, 2.475).

(iii) In order to use the rejection sampling algorithm to simulate from the (unnormalised) posterior distribution, we need to choose a proposal distribution. The suggestion is to use an exponential distribution with mean equal to the mean of the data, i.e.,

$$
g(\theta) = \lambda e^{-\lambda\theta}, \quad \lambda = \frac{1}{\bar{\mathbf{y}}}.
$$

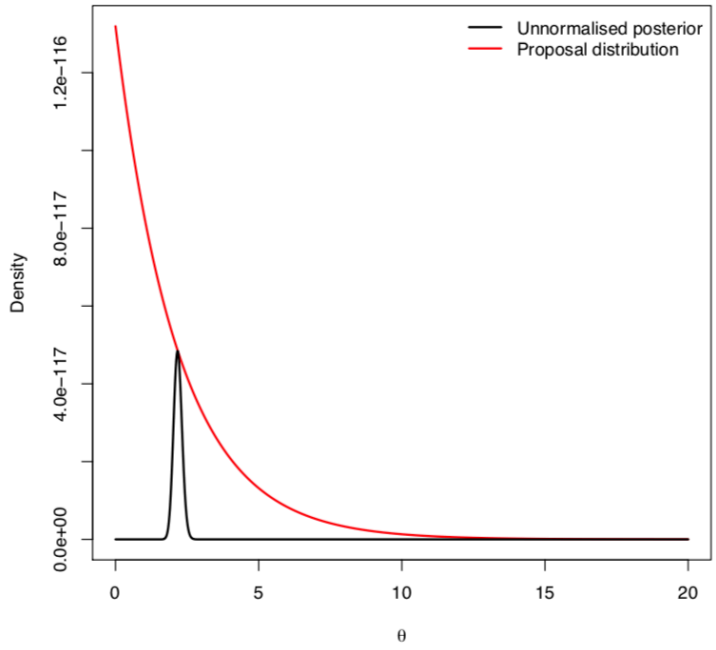
The next task is to find a constant M such that

$$
f(\mathbf{y};\theta)p(\theta) \leq M g(\theta).
$$

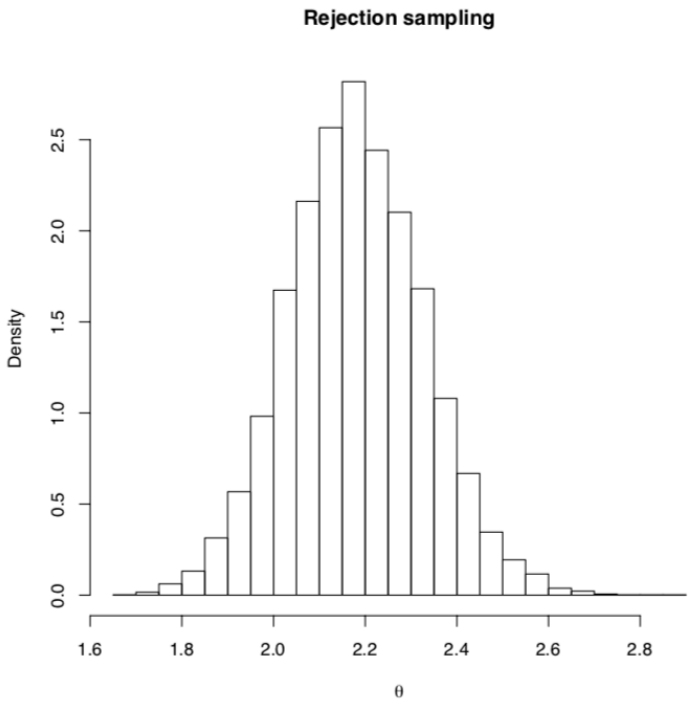
The optimal M is such that

$$
M = \max_{\theta} \left\{ \frac{f(\mathbf{y};\theta)p(\theta)}{g(\theta)} \right\}.
$$

We use the command `optimize` to find the optimal value of M (for further details see the `R` script `Practical2_solutions_Q1.R`).

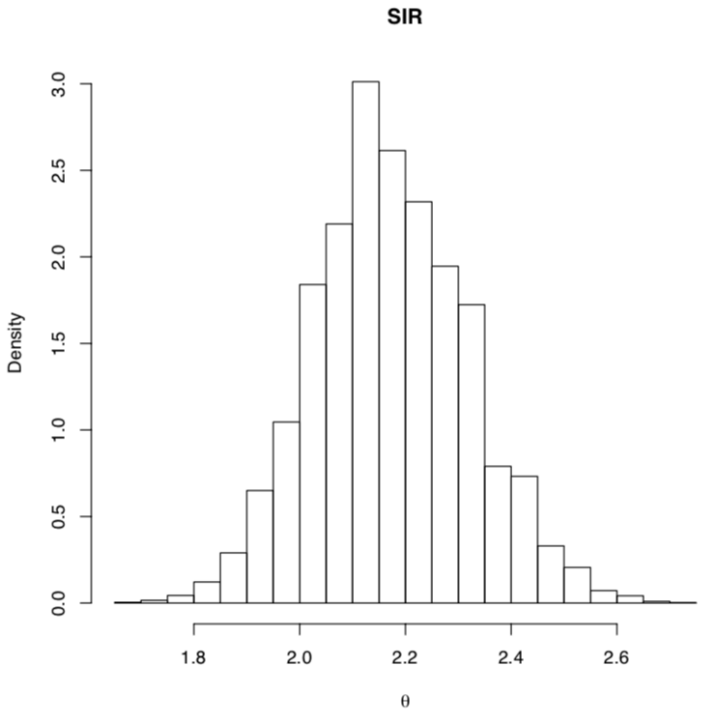


From the above figure we can appreciate that there is a 'large rejection area'. This is confirmed by the acceptance rate of the algorithm, about 6% (see R script for details on the implementation). The histogram of the sampled θ's is plotted below.



For this ensemble of θ values we obtain that the posterior mean of θ is 2.181, its posterior standard deviation is 0.148, and a 95% credible interval is (1.898, 2.477). These values are very similar to the ones obtained with exact calculations. Note that due to the stochastic nature of the algorithm, you might obtain slightly different values from the ones presented here.

(iv) Weneedtoanswerthesamequestionbutnowusingthesampleimportanceresampling(SIR)algorithm. We use the same proposal distribution as in (iii) and we follow the algorithm described on slides 15/16 of the second lecture (file BDA Lecture2.pdf). Note that the SIR algorithm keeps all the sampled values, but they are weighted according to their 'importance'. Applying the algortihm (see R script for details on the implementation) we obtain



for which the posterior mean of θ is 2.173, its standard deviation 0.145, and a 95% credible interval is (1.902, 2.472).

(v) We will now use the rjags package to answer the same question. We need to write in BUGS language the model (i.e., likelihood and prior). The bulk of the code is

```
  model_string <- "model{
  # Likelihood
  for(i in 1:n){
  y[i]~dpois(theta)

}

  # Prior
  theta~dgamma(a,b)
  }"
```

We then pass the model to JAGS, along with the data and hyperparameter values. Here we are not providing initial values, so rjags will generate them.

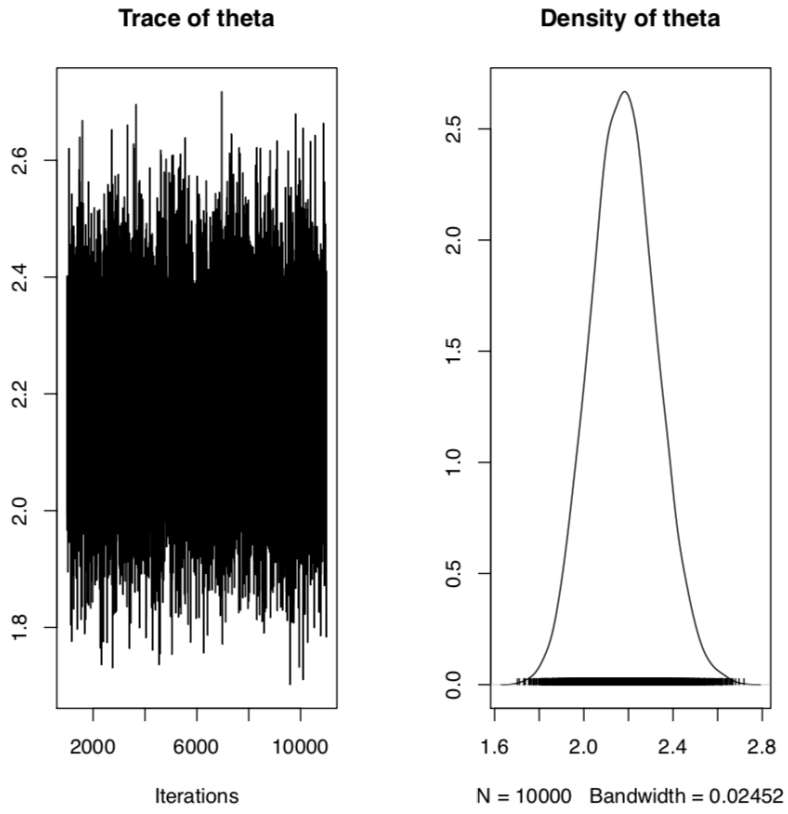
```
  model=jags.model(textConnection(model_string),n.chains=1, data=list(y=y,n=n,a=3,b=1))
```

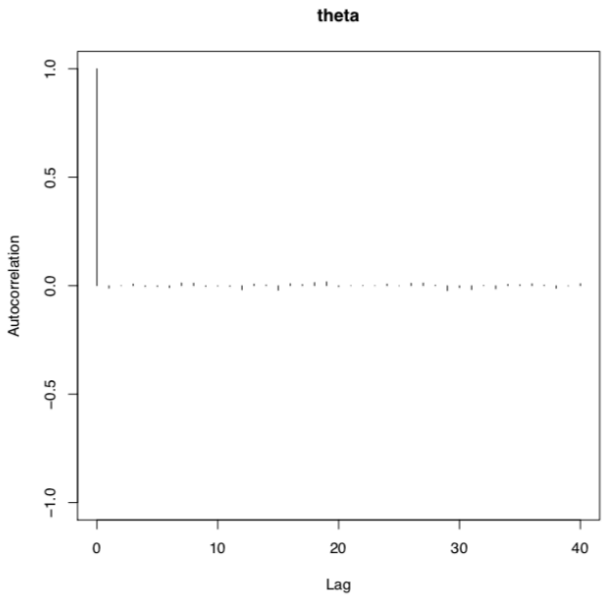
We start with a burn in of 1000 iterations

```
update(model,1000,progress.bar="none")
```

and we then sample a further 10000 iterations

```
res=coda.samples(model, variable.names=c("theta"),n.iter=10000,progress.bar="none")
```





As can be seen above, the traceplot and autocorrelation function look OK (this is a very simple model). The summary statistics of the posterior are

> summary(res)

```
Iterations = 1001:11000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 10000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean             SD       Naive SE Time-series SE
       2.17871        0.14595        0.00146        0.00146

2. Quantiles for each variable:

 2.5%   25%   50%   75% 97.5%
1.902 2.079 2.177 2.276 2.472
```

We can conclude that the results are very similar to the ones obtained in the previous subquestions.

2. The aim of this question is to re-do what we have done in questions 1 and 2 of practical 1 but now using `rjags`.

   (i) We will write a simple JAGS program that consists simply of the binomial likelihood and the beta prior for the success rate θ. The bulk of the code is

   ```
   model_string <- "model{
   #likelihood
   y~dbin(theta,n)

   #prior
   theta~dbeta(a, b)
   }"
   ```

   Proceeding in a similar way as before

   ```
   n=20; y=15; a=9.2; b=13.8
   model=jags.model(textConnection(model_string),data = list(n=n,y=y,a=a,b=b),n.chains=1)
   update(model,1000,progress.bar="none")
   res=coda.samples(model,variable.names=c("theta"), n.iter=20000,progress.bar="none")
   ```

   Traceplot and autocorrelation function do not indicate lack of convergence (see `R` script). The summary statistics are as follows

   ```
   > summary(res)

   Iterations = 2001:22000
   Thinning interval = 1
   Number of chains = 1
   Sample size per chain = 20000

   1. Empirical mean and standard deviation for each variable,
      plus standard error of the mean:
   ```

```
           Mean             SD       Naive SE Time-series SE
      0.5620081      0.0743911      0.0005260      0.0006641

2. Quantiles for each variable:

  2.5%    25%    50%    75%  97.5%
0.4179 0.5111 0.5628 0.6137 0.7049
```

Results are similar to the ones obtained with exact calculations.

(ii) To compute the probability that the true success rate is greater than 0.6 we use the chain of sampled values of θ and compute the proportion of values (over all 20000 iterations) that are greater than 0.6.

```
> mean(res[[1]]>0.6)
[1] 0.313
```

(iii) To compute the required posterior predictive probability in JAGS we simply add the following two lines at the bottom of the existing code

```
#model in BUGS syntax
model_string <- "model{
#likelihood
y~dbin(theta,n)

#prior
theta~dbeta(a, b)

#predictive distribution
ypred~dbin(theta,m)
p25=step(ypred-25)      #equals 1 if ypred-25 >= 0, so an indicator
}"
```

We then repeat the process of compiling the model, running the burn-in and sampling the values to be kept

```
n=20; y=15; a=9.2; b=13.8; m=40
model=jags.model(textConnection(model_string),data = list(n=n,y=y,a=a,b=b,m=m),n.chains=1)
update(model,1000,progress.bar="none")
res=coda.samples(model,variable.names=c("theta","ypred","p25"), n.iter=20000,progress.bar="r
> summary(res)

Iterations = 2001:22000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 20000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

         Mean      SD  Naive SE Time-series SE
p25    0.3273 0.46926 0.0033182      0.0035804
theta  0.5623 0.07452 0.0005269      0.0006503
ypred 22.4912 4.30447 0.0304372      0.0336931

2. Quantiles for each variable:
```

```
         2.5%     25%     50%     75%   97.5%
p25    0.0000  0.0000  0.0000  1.0000  1.0000
theta  0.4138  0.5119  0.5627  0.6138  0.7057
ypred 14.0000 20.0000 23.0000 25.0000 31.0000
```

The quantity `p25` takes the value 1 when the predicted value of y is greater than 25 and 0 otherwise. Then, the chance of observing at least 25 positive responses out of 40 new patients is simply given by the proportion of 1s in the vector `p25` (i.e., by its mean). From the information given above we see that it is 0.3273. Note that in fact the quantity `p25` is not really needed, all we need is the quantity `ypred`, the predicted number of positive responses out of the 40 new patients. This quantity takes values between 0 and 40. From the variable `ypred` we could count the proportion of values that are equal or greater than 25

```
> mean(res[[1]][,"ypred"]>=25)
[1] 0.32735
```

(iv) To compute the required probability, we simulate data under this prior. The code is as follows:

```
model_string <- "model{
theta ~ dbeta(a, b)
y~dbin(theta,n)
p15=step(y-15)
}"

model=jags.model(textConnection(model_string),data=list(n=n,a=a,b=b),n.chains=1)
update(model,1000,progress.bar="none")
res=coda.samples(model,variable.names=c("y","p15"),n.iter=20000,progress.bar="none")
summary(res)

Iterations = 1001:21000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 20000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

      Mean     SD  Naive SE Time-series SE
p15 0.0145 0.1195 0.0008453      0.0008453
y   7.9714 2.9139 0.0206041      0.0206041

2. Quantiles for each variable:

    2.5% 25% 50% 75% 97.5%
p15    0   0   0   0     0
y      3   6   8  10    14
```

Note that we are not passing to `jags.model` the data `y`. Again, the quantity `p15` takes the value 1 when the generated value y is greater than 15 and 0 otherwise. Then, the probability of observing at least 15 successes under this prior is 0.0145. Note that as before, we do not really need `p15`; all information that we need is contained in `y`. A similar result is obtained by typing

```
> mean(res[[1]][,"y"]>=15)
[1] 0.0145
```

(v) We first need to write in JAGS the model with the mixture prior, which we do as follows

```
model_string <- "model{
y~dbinom(theta,n)
theta~dbeta(a[pick], b[pick])    # a[1]=9.2, b[1]=13.8;  a[2]=12;  b[2] = 3
pick~dcat(p[1:2])                # pick takes value 1 or 2 with prior prob p[1] or p[2]
ypred~dbin(theta,m)
p25=step(ypred-25)
}"
```

Note that the main difference is that we now have a categorical (in fact binary!) variable `pick` which takes the value 1 with probability p_1 and value 2 with probability p_2. According to the values of this variable, we will then select $\theta \sim \text{Beta}(a_1, b_1)$ or $\theta \sim \text{Beta}(a_2, b_2)$. The rest of the code is similar to the ones we have been using

```
n=20; y=15; a=c(9.2,12); b=c(13.8,3); p=c(0.95,0.05); m=40
model=jags.model(textConnection(model_string),data =
                   list(y=y,n=n,m=m,a=a,b=b,p=p),n.chains=1)
update(model,1000,progress.bar="none")
res=coda.samples(model, variable.names=c("theta","ypred","p25"),
                 n.iter=20000,progress.bar="none")
```

Then, we have

```
> mean(res[[1]][,"theta"] > 0.6)
[1] 0.578
```

(vi) As in point (iii), to answer this question we can simply do

```
> mean(res[[1]][,"ypred"]>=25)
[1] 0.5691
```

or, alternatively

```
> mean(res[[1]][,"p25"])
[1] 0.5691
```

(vii) This is similar to point (iv) but now using the mixture prior

```
model_string <- "model{
y~dbinom(theta,n)
theta~dbeta(a[pick], b[pick])
pick~dcat(p[1:2])
p15=step(y-15)
}"

n=20; a=c(9.2,12); b=c(13.8,3); p=c(0.95,0.05)
model=jags.model(textConnection(model_string),data = list(n=n,a=a,b=b,p=p),n.chains=1)
update(model,1000,progress.bar="none")
res=coda.samples(model,variable.names=c("y","p15"),n.iter=20000,progress.bar="none")

> summary(res)

Iterations = 1001:21000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 20000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
```

```
      Mean     SD Naive SE Time-series SE
p15 0.0517 0.2214 0.001566       0.001592
y   8.4008 3.3972 0.024022       0.024022

2. Quantiles for each variable:

    2.5% 25% 50% 75% 97.5%
p15    0   0   0   0     1
y      3   6   8  10    17
```

The asked probability is then 0.0517. Alternatively, the same value can be obtained by doing

```
> mean(res[[1]][,"y"]>=15)
[1] 0.0517
```