R Supplemental File

This document contains all of the Bayesian calculations described in the paper *Bayesian Computing in the Undergraduate Statistics Curriculum*.

Required Software and Packages

The JAGS software is installed from http://mcmc-jags.sourceforge.net/

Also the following R packages need to be installed from CRAN (https://cran.r-project.org/):

- ProbBayes
- runjags
- ggplot2

```
library(ProbBayes)
library(runjags)
library(ggplot2)
```

Section 2.1 - Discrete Bayes

Visits to an Emergency Department. Observe number of arrivals $y_1, ..., y_n$ that are independent Poisson with mean λ. In the example, $n = 10$ and $s = \sum y_i = 31$.

Assume a discrete prior on λ. The Bayes' rule calculations are illustrated below.

```
lambda <- c(3, 3.5, 4, 4.5, 5)
prior <- c(1, 2, 4, 2, 1) / 10
s <- 31
n <- 10

likelihood <- exp(- n * lambda) * lambda ^ s
product <- prior * likelihood
posterior <- product / sum(product)

(df <- data.frame(lambda, Prior = prior,
                 Likelihood = round(likelihood, 1),
                 Product = round(product, 2),
                 Posterior = round(posterior, 3)))
```

```
##   lambda Prior Likelihood Product Posterior
## 1    3.0   0.1       57.8    5.78     0.241
## 2    3.5   0.2       46.3    9.26     0.386
## 3    4.0   0.4       19.6    7.84     0.327
## 4    4.5   0.2        5.1    1.02     0.042
## 5    5.0   0.1        0.9    0.09     0.004
```

Section 2.2 - Conjugate Analyses

Visits to an Emergency Department. Observe number of arrivals $y_1, ..., y_n$ that are independent Poisson with mean λ. In the example, $n = 10$ and $s = \sum y_i = 31$.

Assume that λ has a gamma prior with shape parameter $\alpha = 80$ and rate parameter $\beta = 20$. Below code illustrates the construction of a 90% Bayesian interval estimate for λ.

```
alpha <- 80
beta <- 20
s <- 31
n <- 10
(alpha_new <- alpha + s)
```

```
## [1] 111
```

```
(beta_new <- beta + n)
```

```
## [1] 30
```

```
qgamma(c(0.05, 0.95), alpha_new, beta_new)
```

```
## [1] 3.141908 4.295974
```

Section 2.3 - Normal Approximation

Facebook example Survey is given to college students. Let p_W and p_M denote the proportions of women and men who are frequent users of Facebook. Fitting a logistic model

$$\log \frac{p_M}{1 - p_M} = \beta_0$$

$$\log \frac{p_F}{1 - p_F} = \beta_0 + \beta_1$$

Apriori assume that β_0 and β_1 are independent where β_0 is Normal with location 0 and standard deviation 0.5 and β_1 is Normal with mean 0 and standard deviation 100.

Function `logistic_posterior()` computes the log posterior density of (β_0, β_1).

```
logistic_posterior <- function(theta, df){
  beta0 <- theta[1]
  beta1 <- theta[2]
  lp <- beta0 + beta1 * df$female
  p <- exp(lp) / (1 + exp(lp))
  sum(df$s * log(p) + df$f * log(1 - p)) +
    dcauchy(beta1, 0, 0.5, log = TRUE) +
    dnorm(beta0, 0, sqrt(1 / 0.0001), log = TRUE)
}
```

Among the 30 women, 15 are frequent Facebook users and among the 30 men, 8 are frequent Facebook users. Place the data into a data frame with variables s, number of FB users, f, number of not-FB users, n sample size, and $female$, indicator of female group.

```
ldata <- data.frame(s = c(15, 8),
                 f = c(30 - 15, 30 - 8),
                 n = c(30, 30),
                 female = c(1, 0))
```

Use the function `laplace()` in the `LearnBayes` package to find a normal approximation to the posterior. Output the mean and variance-covariance matrix of the approximation.

```
library(LearnBayes)
fit <- LearnBayes::laplace(logistic_posterior,
                           c(0, 0), ldata)
fit$mode
```

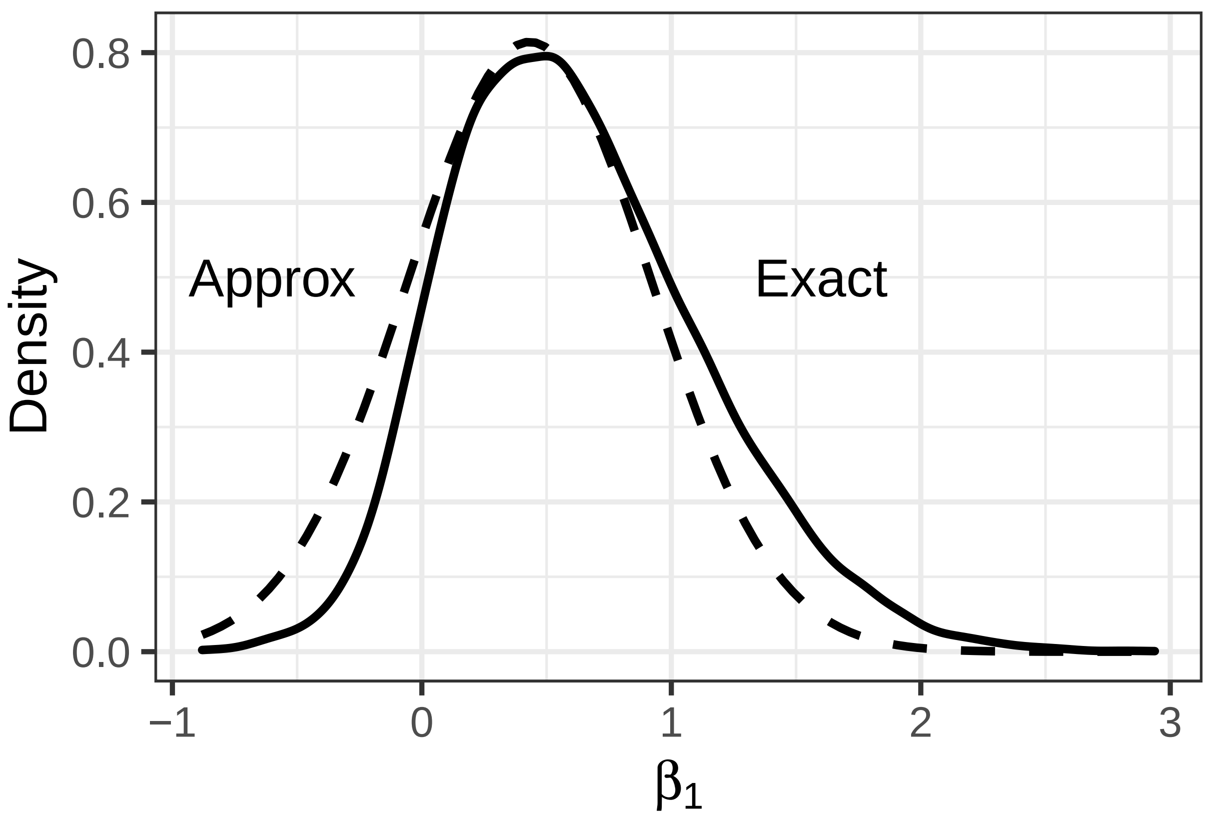
```
## [1] -0.6968111  0.4316592
```

```
fit$var
```

```
##            [,1]       [,2]
## [1,]  0.1375169 -0.1260742
## [2,] -0.1260742  0.2399836
```

Compare exact posterior of β_1 (computed using simulation from a grid approach) with normal approximation.

```
library(latex2exp)
out <- simcontour(logistic_posterior,
                  c(-3, 1, -1, 3), ldata,
                  10000)
exact <- data.frame(beta1 = out$y)
ggplot(exact, aes(beta1)) +
  geom_density(size = 1.5) +
  stat_function(fun = dnorm,
                size = 1.5,
                linetype = "dashed",
                args = list(mean = fit$mode[2],
                     sd = sqrt(fit$var[2, 2]))) +
  xlab(TeX('$\\beta_1')) +
  ylab("Density") +
  annotate(geom = "text", x = 1.6, y = 0.5,
           label = "Exact", size = 7) +
  annotate(geom = "text", x = -0.6, y = 0.5,
           label = "Approx", size = 7) +
  increasefont() +
  theme_bw(base_size = 20)
```



Section 3.2 - Example: ED Visits

Visits to an Emergency Department Observe number of arrivals $y_1, ..., y_n$ that are independent Poisson with mean λ. In the example, $n = 10$ and $s = \sum y_i = 31$. Assume that λ has a gamma prior with shape parameter $\alpha = 80$ and rate parameter $\beta = 20$.

Find the parameters of the gamma posterior distribution.

```
alpha <- 80
beta <- 20
s <- 31
n <- 10
alpha_new <- alpha + s
beta_new <- beta + n
```

Simulate 1000 draws from the gamma posterior.

```
set.seed(123)
S <- 1000
lambda_draws <- rgamma(S, shape = alpha_new,
                       rate = beta_new)
```

Find a 90% interval estimate for λ by finding the 5th and 95th percentiles of the simulated draws.

```
quantile(lambda_draws, c(0.05, 0.95))
```

```
##       5%      95%
## 3.136647 4.248430
```

Interested in estimating

$$h(\lambda) = P(y \leq 2|\lambda) = \exp(-\lambda)(1 + \lambda + \lambda^2/2)$$

. Simulate posterior of $h(\lambda)$ and find its posterior mean.

```
h_lambda <- exp(-lambda_draws) * (1 + lambda_draws + lambda_draws^2/2)
mean(h_lambda)
```

```
## [1] 0.2928842
```

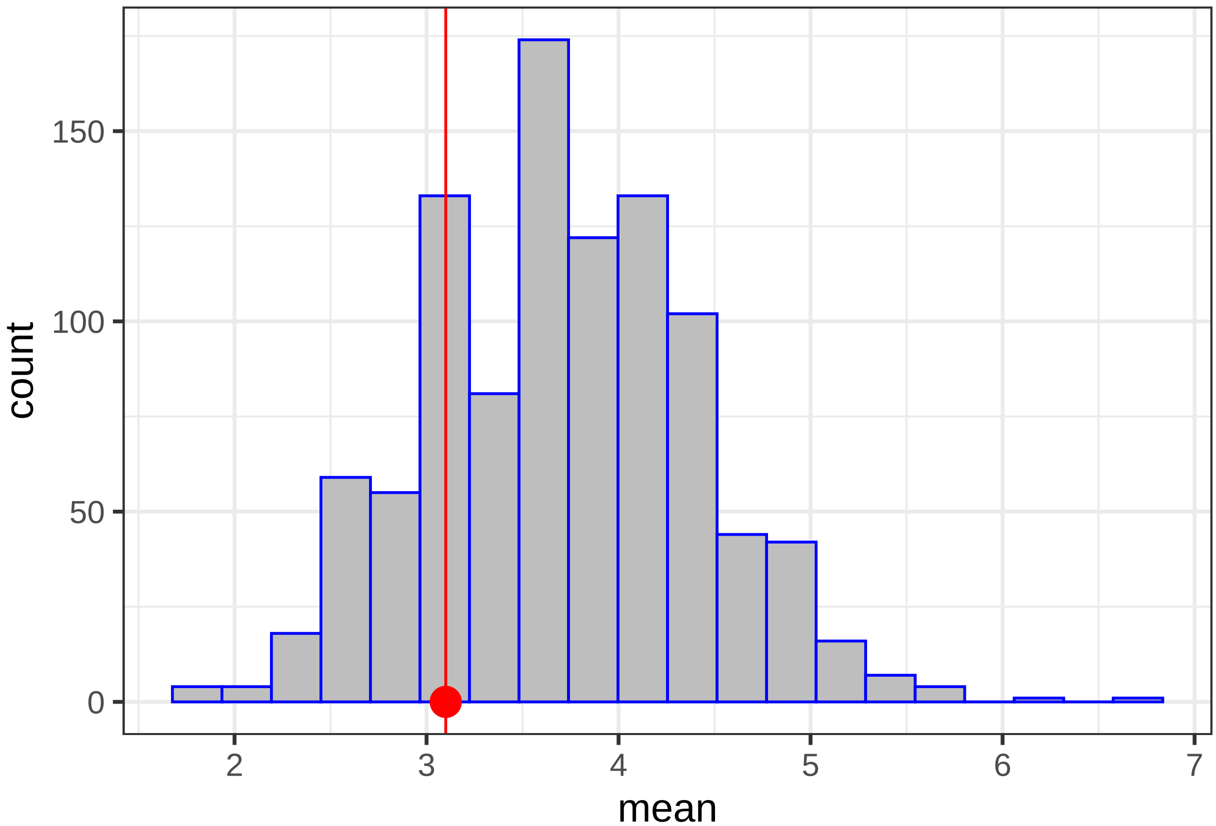
Suppose we take a future sample of size 10 of ED counts. Simulate predictive distribution of $\bar{y}$ by (1) simulating λ from the posterior distribution and (2) simulating a sample of 10 from a Poisson distribution with mean λ and computing $\bar{y}$. Repeat this process 1000 times to get a sample from predictive distribution of $\bar{y}$.

```
set.seed(123)
one_pp_sim <- function(n){
  lambda_draw <- rgamma(1, shape = 111, rate = 30)
  y_pred <- rpois(n, lambda_draw)
  mean(y_pred)
}
sample_means <- replicate(1000, one_pp_sim(10))
```

Display the simulated posterior predictive distribution of $\bar{y}$ by a histogram and overlay the actual value of $\bar{y} = 31/10 = 3.1$.

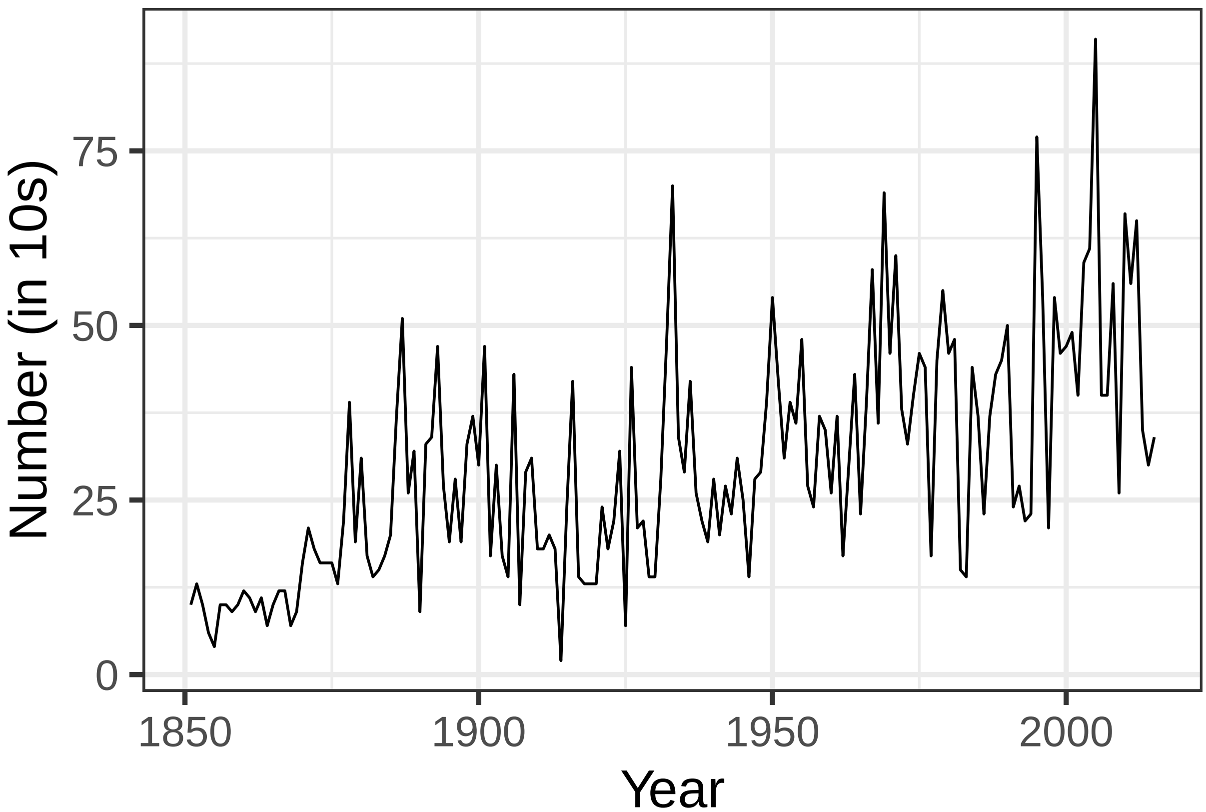
```
require(ggplot2)
df <- data.frame(mean = sample_means)
ggplot(df, aes(mean)) +
  geom_histogram(bins = 20,
                 color = "blue",
                 fill = "grey") +
```

```
  theme_bw(base_size = 15, base_family = "") +
  geom_vline(xintercept = 31/n, colour = "red") +
annotate("point", x = 31/n, y = 0,
         colour = "red", size = 5)
```



Section 4.2 - A Change Point Example of Named Storms

```
Storms <- read.csv("atlantic.csv")

Storms$Year <- floor(Storms$Date / 10000)

NumYears <- length(unique(Storms$Year))

YearCounts <- data.frame(matrix(NA, nrow = NumYears, ncol = 2))
names(YearCounts) <- c("Year", "Number")
YearCounts$Year <- seq(1851, 2015, 1)

for (y in 1:NumYears){
  YearCounts$Number[y] <- round(sum(Storms$Year == YearCounts$Year[y]) / 10)
}

ggplot(YearCounts, aes(x = Year, y = Number)) +
  geom_line() +
  theme_bw(base_size = 20) +
  ylab("Number (in 10s)")
```



Section 4.3 - The Gibbs Sampler

The following script will implement a Gibbs sampling algorithm.

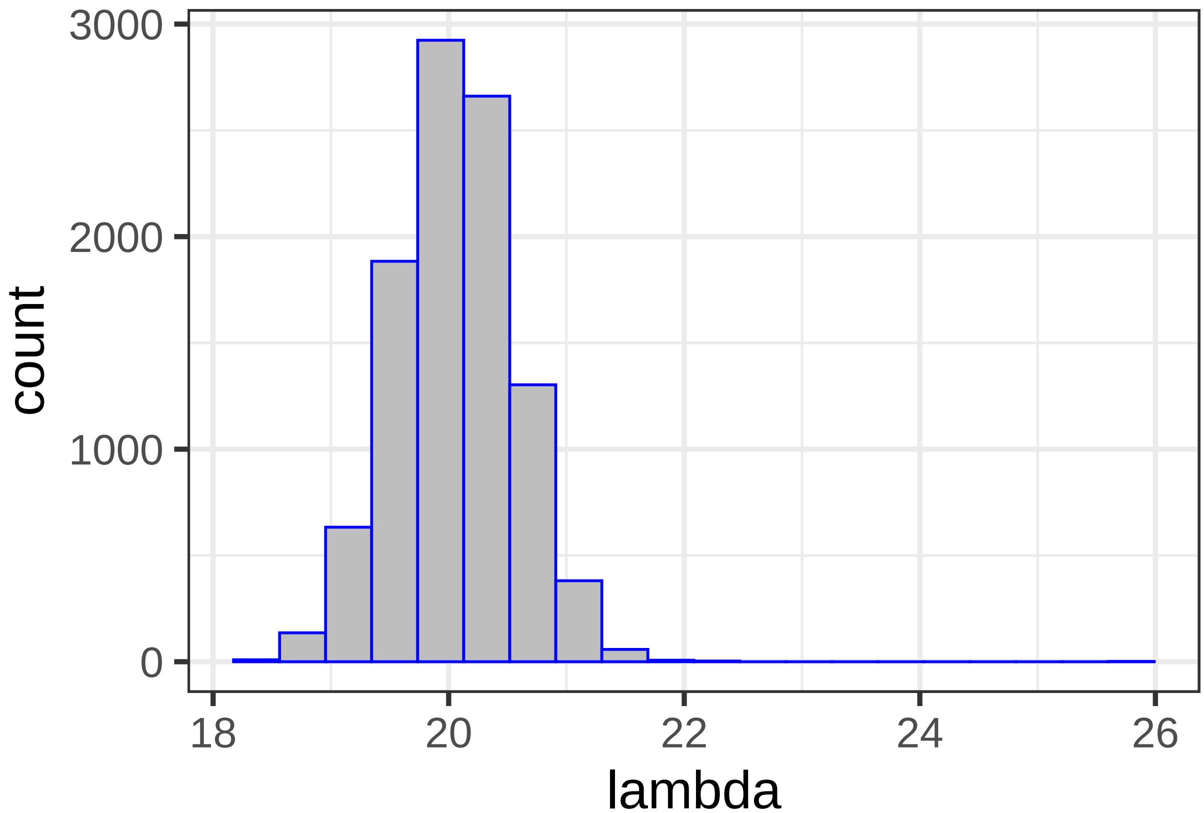
The simulated draws from $\{\lambda_1, \lambda_2, M\}$ are stored in the vector `post_draws_Gibbs`.

```
alpha_1 <- 5
beta_1 <- 0.5
alpha_2 <- 5
beta_2 <- 0.5
y <- YearCounts$Number
n <- length(y)
set.seed(12)

iter <- 10000
M <- which(rmultinom(1, 1, rep(1 / (n - 1), n - 1)) == 1)
post_draws_Gibbs <- matrix(NA, nrow = iter, ncol = 3)
for (i in 1:iter){
  ## draw lambda_1
  lambda_1 <- rgamma(1, sum(y[1:M]) + alpha_1, M + beta_1)
  ## draw lambda_2
  lambda_2 <- rgamma(1, sum(y[(M+1):n]) + alpha_2, n - M + beta_2)
  ## draw M
  term_m_log <- rep(NA, n - 1)
  subtract_term <- (log(lambda_1) + log(lambda_2)) * sum(y) / 2 +
    (lambda_2 - lambda_1) * n / 2
  for (m in 1:(n-1)){
    term_m_log[m] <- log(lambda_1) * sum(y[1:m]) +
      log(lambda_2) * sum(y[(m+1):n]) +
```

```
      (lambda_2 - lambda_1) * m - subtract_term
  }
  term_m <- exp(term_m_log)
  normalized_probs <- term_m / sum(term_m)
  M <- which(rmultinom(1, 1, normalized_probs) == 1)

  post_draws_Gibbs[i, ] <- c(lambda_1, lambda_2, M)
}
```

Here is a histogram showing the posterior of λ_1.

```
ggplot(data.frame(lambda = post_draws_Gibbs[,1]), aes(lambda)) +
  geom_histogram(fill = "grey",
                 color = "blue",
                 bins = 20) +
  theme_bw(base_size = 20)
```



Section 4.4 - The Metropolis Algorithm

Section 4.4.2 - Metropolis Within Gibbs Sampling

First write a function that computes the log posterior of λ_1.

```
logpost <- function(lambda_1){
  log(lambda_1) * sum(y[1:M]) - M * lambda_1 -
    (lambda_1 - mu) ^ 2 / (2 * sigma ^ 2)
}
```

The following script will implement a Metropolis within Gibbs sampling algorithm.

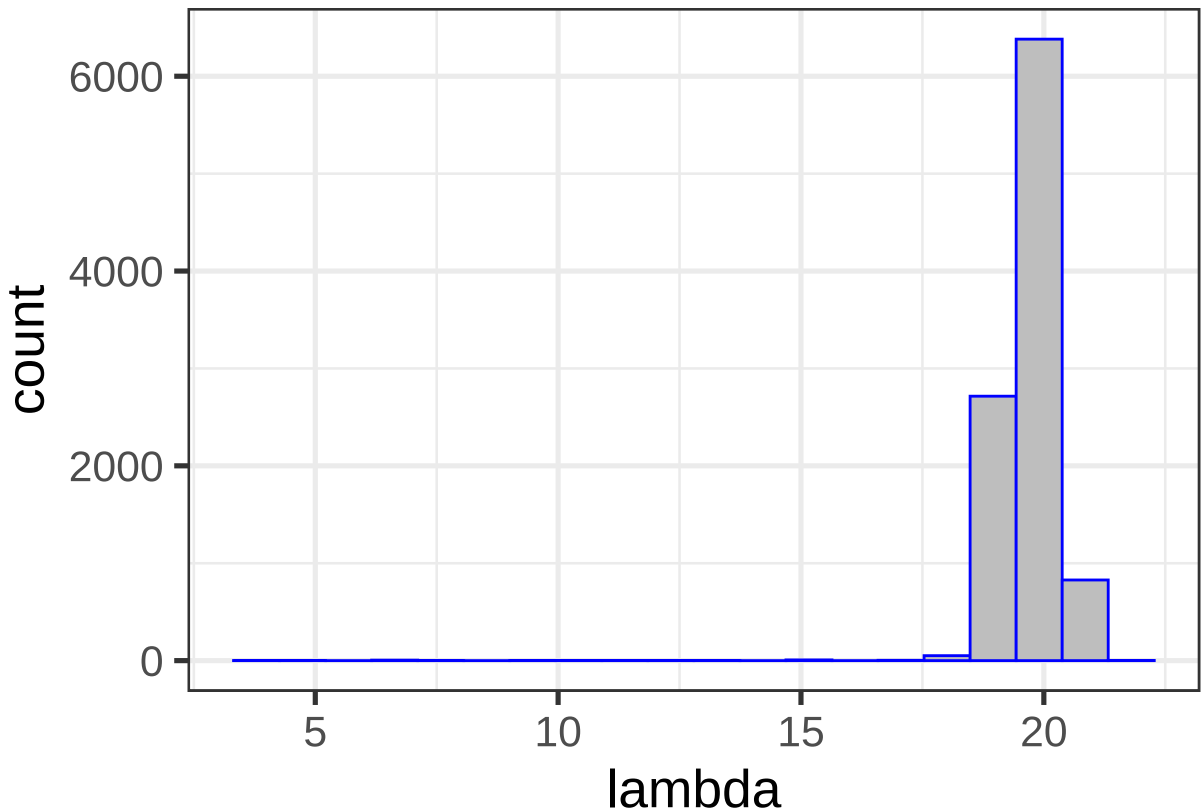
The simulated draws from $\{\lambda_1, \lambda_2, M\}$ are stored in the vector `post_draws_Metropolis`.

```
mu <- 5
sigma <- 3
alpha_2 <- 5
beta_2 <- 0.5
y <- YearCounts$Number
n <- length(y)
C <- 2
set.seed(123)

lambda_1_c <- rnorm(1, mu, sigma)
post_draws_Metropolis <- matrix(NA, nrow = iter, ncol = 3)
accept_vector <- rep(NA, iter)
for (i in 1:iter){
  ## draw lambda_1
  lambda_1_p <- runif(1, min = lambda_1_c - C,  max = lambda_1_c + C)
  R <- exp(logpost(lambda_1_p) - logpost(lambda_1_c))
  accept <- ifelse(runif(1) < R, "yes", "no")
  lambda_1_c <- ifelse(accept == "yes", lambda_1_p, lambda_1_c)
  ## draw lambda_2
  lambda_2 <- rgamma(1, sum(y[(M+1):n]) + alpha_2, n - M + beta_2)
  ## draw M
  term_m_log <- rep(NA, n - 1)
  subtract_term <- (log(lambda_1) + log(lambda_2)) * sum(y) / 2 +
    (lambda_2 - lambda_1) * n / 2
  for (m in 1:(n-1)){
    term_m_log[m] <- log(lambda_1) * sum(y[1:m]) +
      log(lambda_2) * sum(y[(m+1):n]) +
      (lambda_2 - lambda_1) * m - subtract_term
  }
  term_m <- exp(term_m_log)
  probs <- term_m / sum(term_m)
  M <- which(rmultinom(1, 1, probs) == 1)

  post_draws_Metropolis[i, ] <- c(lambda_1_c, lambda_2, M)
  accept_vector[i] <- accept
}

sum(accept_vector == "yes") / iter
```

```
## [1] 0.3824
```

Here is a histogram showing the posterior of λ_1.

```
ggplot(data.frame(lambda = post_draws_Metropolis[,1]), aes(lambda)) +
  geom_histogram(fill = "grey",
                 color = "blue",
                 bins = 20) +
  theme_bw(base_size = 20)
```



Section 4.4 - Coding an MCMC Sampler Using JAGS

The same change point model for named storms is fit using JAGS.

JAGS is installed from http://mcmc-jags.sourceforge.net/

The `runjags` package provides an R interface to JAGS.

The Bayesian logistic model is represented by means of a JAGS character script and stored in the variable `modelString`.

```
library(runjags)
modelString <-"
model {
for (i in 1:n){
lambda[i] = ifelse(i < M, lambda1, lambda2)
y[i] ~ dpois(lambda[i])
}
lambda1 ~ dgamma(alpha1, beta1)
lambda2 ~ dgamma(alpha2, beta2)
M ~ dunif(1, n - 1)
}"
```

The data is represented in JAGS by a list in `the_data`. The `run.jags()` function runs the JAGS sampler with the following inputs:

- modelString is the JAGS script defining the model
- n.chains is the number of chains of the simulation
- data contains the data stored as a list
- monitor indicates which simulated variables to collect

- adapt is the number of iterations for the adaption phase of the sampling
- burnin is the number of iterations for the burn-in phase
- sample is the number of iterations for the collection phase
- thin is the number of thinning iterations for the collection phase

The output variable "'posterior''', contains all of the simulated draws.

By use of the `summary()` function, we output summaries of the marginal posterior distributions of β_0 and β_1.

```
the_data <- list(y = y, n = n,
                 alpha1 = 5, alpha2 = 5,
                 beta1 = 0.5, beta2 = 0.5)
posterior <- run.jags(modelString,
                      n.chains = 1,
                      data = the_data,
                      monitor = c("lambda1", "lambda2", "M"),
                      adapt = 1000,
                      burnin = 5000,
                      sample = 500,
                      thin = 10)
```

```
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Adapting the model for 1000 iterations...
## Burning in the model for 5000 iterations...
## Running the model for 5000 iterations...
## Simulation complete
## Calculating summary statistics...
## Finished running the simulation
```

```
summary(posterior)
```

```
##          Lower95   Median  Upper95     Mean        SD Mode      MCerr MC%ofSD
## lambda1 19.11246 20.03702 21.00233 20.03634 0.4847344   NA 0.02167798     4.5
## lambda2 37.57541 38.81701 40.09756 38.82184 0.6690473   NA 0.02992071     4.5
## M       80.13913 81.08834 81.99166 81.05935 0.5711638   NA 0.02554322     4.5
##         SSeff     AC.100 psrf
## lambda1   500 0.03577778   NA
## lambda2   500 0.03657318   NA
## M         500 0.02062072   NA
```