HW7_bs2996

Bangda Sun

November 9, 2016

Part 1

(i) Write function poisLoglik

```
poisLoglik <- function(lambda, data){
  # input the lambda and x, return the log-likelihood
  x <- data
  Loglik <- sum(log((lambda^x * exp(-lambda)) / factorial(x)))
  return(Loglik)
}
# the value of (1, 0, 0, 1, 1)
data <- c(1, 0, 0, 1, 1)
poisLoglik(lambda = 1, data = data)
```

```
## [1] -5
```

Therefore the log-likelihood of the data when $\lambda = 1$ is -5.

(ii) Write function count_new_genres

```
setwd("C://Users//Bangda//Desktop//GR5206 Materials//Hwk7")
moretti <- read.csv("moretti.csv", header = TRUE)
count_new_genres <- function(year){
  # input one year and return the number of new genres at that year
  num_new_genres <- sum(moretti$Begin == year)
  return(num_new_genres)
}
# the value of 1803 and 1850
count_new_genres(1803)
```

```
## [1] 0
```

```
count_new_genres(1850)
```

```
## [1] 3
```

Therefore the value for 1803 is 0 and the value for 1850 is 3.

(iii) Create a vector: new_genres

```
# create a year vector corrrespond to the number of new genres
year <- 1740:1900
new_genres <- rep(NA, length(year))
for (i in 1:length(new_genres)){
  # count for every year
  new_genres[i] <- count_new_genres(year[i])
}
# position of year 1803 in the vector
which(year == 1803)
```

```
## [1] 64
```

```
# position of year 1850 in the vector
which(year == 1850)
```

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

```
# value of year 1803
new_genres[which(year == 1803)]
```

```
## [1] 0
```

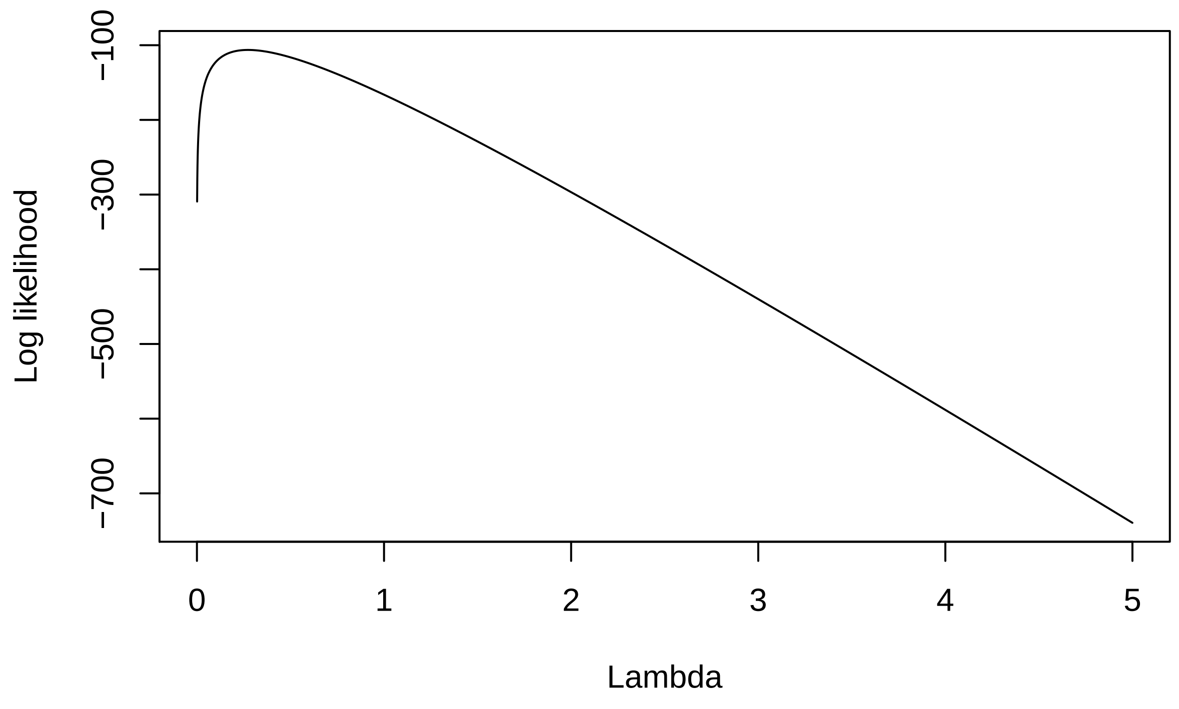
```
new_genres[which(year == 1850)]
```

```
## [1] 3
```

We can see the positions correspond to the year 1803 and 1850 are 64 and 111. The value is 0 and 3. The vector contain all years between 1740 and 1900.

(iv) Plot poisLoglik

```
# set the range of lambda
lambda <- seq(0, 5, by = 0.001)
# initialize the vector p to store the value of log-likelihood
p <- rep(NA, length(lambda))
for (i in 1:length(lambda)){
  p[i] <- poisLoglik(lambda = lambda[i], data = new_genres)
}
plot(lambda, p, type = "l", xlab = "Lambda", ylab = "Log likelihood")
```



```r
# check the maximum point, it should be 0.273
lambda[which.max(p)]
```

```
## [1] 0.273
```

(v) Use nlm() to verify

```r
negpoisLoglik <-function(lambda, data){
  # take oppsite sign to poisLoglik function
  x <- data
  Loglik <- -sum(log((lambda^x * exp(-lambda)) / factorial(x)))
  return(Loglik)
}
nlm(negpoisLoglik, 0.1, data = new_genres)[1:2]
```

```
## $minimum
## [1] 106.3349
##
## $estimate
## [1] 0.2732914
```

We take the oppsite sign of poisLoglik get a new function negpoisLoglik, its minima should equal to the maxima o poisLoglike. And the nlm() function gives that $\lambda = 0.273$.

(vi) Create a vector: intergenre_intervals

```r
appear_year <- moretti$Begin
intergenre_intervals <- diff(appear_year)
# mean of interval
mean_interval <- mean(intergenre_intervals); mean_interval
```

```
## [1] 3.44186
```

```r
# standard deviation of interval
sd_interval <- sd(intergenre_intervals); sd_interval
```

```
## [1] 3.705224
```

```r
# coefficient of variation of interval
cv_interval <- sd_interval / mean_interval; cv_interval
```

```
## [1] 1.076518
```

(vii)

(a) Write function to get intervals

```r
calc_intervals <- function(data){
  # input the number of new genres in each year
  # return the intervals

  # create a index 1 to the length of data, represent the year
  index1 <- 1:length(data)
  # create a empty vector to store the appear year
  index2 <- c()
  for (i in 1:length(data)){
    # if the number of new genre is x in i-th year, the index1 will be repeated x times
    index2 <- c(index2, rep(index1[i], data[i]))
  }
  return(diff(index2))
}
new_intervals <- calc_intervals(data = new_genres)
# check when given new_genres it should return same vector as intergenre_intervals
identical(new_intervals, intergenre_intervals)
```

```
## [1] TRUE
```

(b) Write function to simulate

```r
poiproc <- function(num.year, mean){
  # input the number of years and the mean
  # return the interval and cv

  # simulate poisson process
  # based on the definition of poisson process,
  # the number of appearance in a unit interval has Poi(lambda)
  appearance <- rpois(num.year, lambda = mean)
  # calculate intervals with calc_intervals function
  intervals <- calc_intervals(appearance)
  return(list(intervals = intervals, cv = sd(intervals)/mean(intervals)))
}
# simulation
poiproc(num.year = 161, mean = 0.273)
```

```
## $intervals
##  [1]  1  2  4  2  5  3  1 12  2 13  1 12  3  6  9  9  1  1  0  4  1  4  2
## [24] 11  1  1  2  5  6  2  4  1  8
##
## $cv
## [1] 0.8957268
```

(viii) Simulation

```r
cv <- rep(NA, 100000)
for (i in 1:length(cv)){
  cv[i] <- poiproc(num.year = 161, mean = 0.273)$cv
}
# calculate the fraction of runs have a higher coefficient of variation than moretti's data
mean(cv > cv_interval)
```

```
## [1] 0.23036
```

(ix) Explaination

If the genres doesn't tend to appear together in bursts, it's appear should have poisson distributions. And we can use permutation test to test if the coefficient of variation of the data should be in the distribution of coefficients of variation of simulated data. Where the proportion is 0.23, means the p-value of the test is 0.23, and if we use the common significance level $\alpha = 0.05$ we will fail to reject the null hypothesis, which means the appear of new genres should appear independently and randomly. This is what we know from the above analysis. But here we only use one value of λ, also we use the MLE, it might not be precisely enough.

Part 2

(x) Report the lambda of each decade

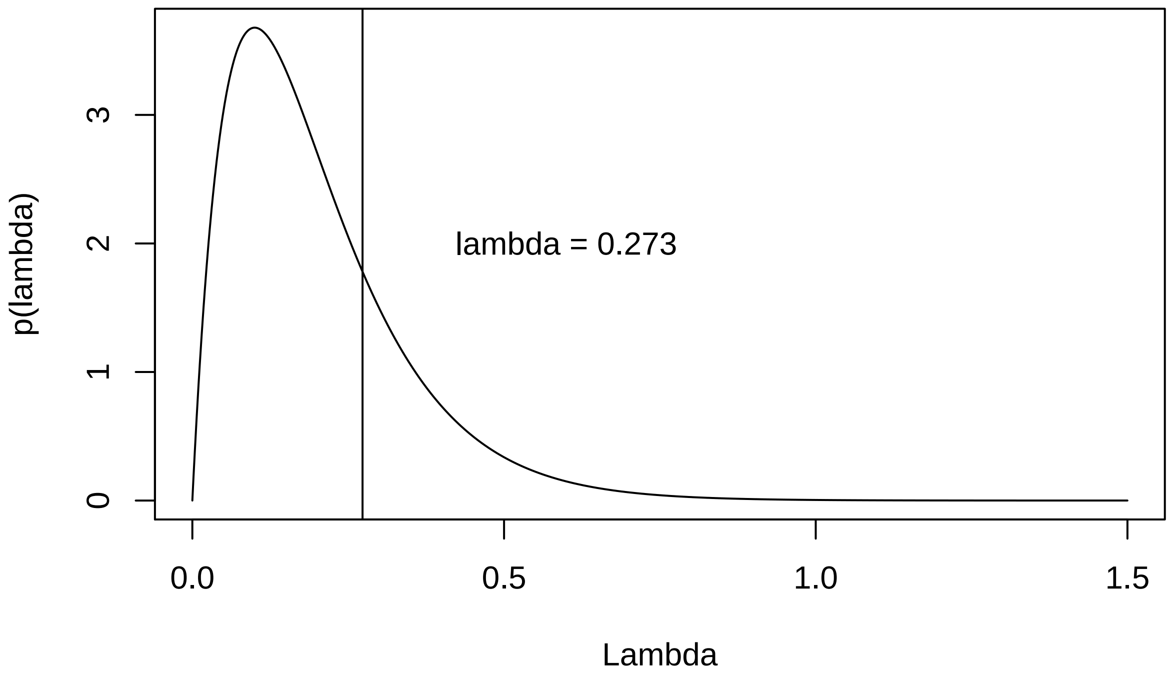
```r
# ignore the 1900
appearance <- data.frame(year = year, num.appear = new_genres)
appearance <- appearance[-161,]
# label with groups
appearance$decade <- rep(1:16, each = 10)
# calculate the MLE
lambdamle <- as.numeric(tapply(appearance$num.appear, appearance$decade, mean)); lambdamle
```

```
##  [1] 0.2 0.1 0.2 0.2 0.1 0.2 0.3 0.1 0.4 0.5 0.5 0.5 0.1 0.5 0.5 0.0
```

(xi)

(a) Plot the density

```r
x <- seq(0, 1.5, by = 0.001)
plot(x, dgamma(x, shape = 2, scale = 0.1), type = "l", xlab = "Lambda", ylab = "p(lambda)")
# check if 0.273 is near to the peak
abline(v = 0.273)
text(0.6, 2, "lambda = 0.273")
```



We can see the most-likely rate in Part 1 is not very close to the peak of prior.

(b) Write function initial

```r
initial <- function(){
  # draw a number from prior
  draw <- rgamma(1, shape = 2, scale = 0.1)
  return(draw)
}
```

(c) Write function proposal

```r
proposal <- function(lambda){
  sum <- lambda + rnorm(1, mean = 0, sd = 0.001)
  return(ifelse(sum > 0, sum, lambda))
}
```

(d) Write function posterior

```r
posterior <- function(lambda, data){
  # take the exponential of the log-likelihood
  likelihood <- exp(poisLoglik(lambda = lambda, data = data))
  # bayes rule
  posterior <- dgamma(lambda, shape = 2, scale = 0.1) * likelihood
  return(posterior)
}
posterior(0.2, new_genres)
```

```
## [1] 2.571445e-47
```

(e) Draw from posterior

```r
new_genres12 <- appearance$num.appear[appearance$decade == 12]
metrostep <- function(x){
  z <- proposal(x)
  u <- runif(1)
  ratio <- posterior(z, new_genres12) / posterior(x, new_genres12)
  if (u < ratio){
    accepted.val <- z
  } else {
    accepted.val <- x
  }
  return(accepted.val)
}
n <- 100000
vals <- vector(length = n)
vals[1] <- initial()
for (t in 2:n){
  vals[t] <- metrostep(vals[t-1])
}
# don't discard first 10000
mean_val1 <- mean(vals); mean_val1
```

```
## [1] 0.2230007
```

```r
sd_val1 <- sd(vals); sd_val1
```

```
## [1] 0.05007386
```

```r
# discard first 10000
vals2 <- vals[-(1:10000)]
mean_val2 <- mean(vals2); mean_val2
```

```
## [1] 0.2299663
```

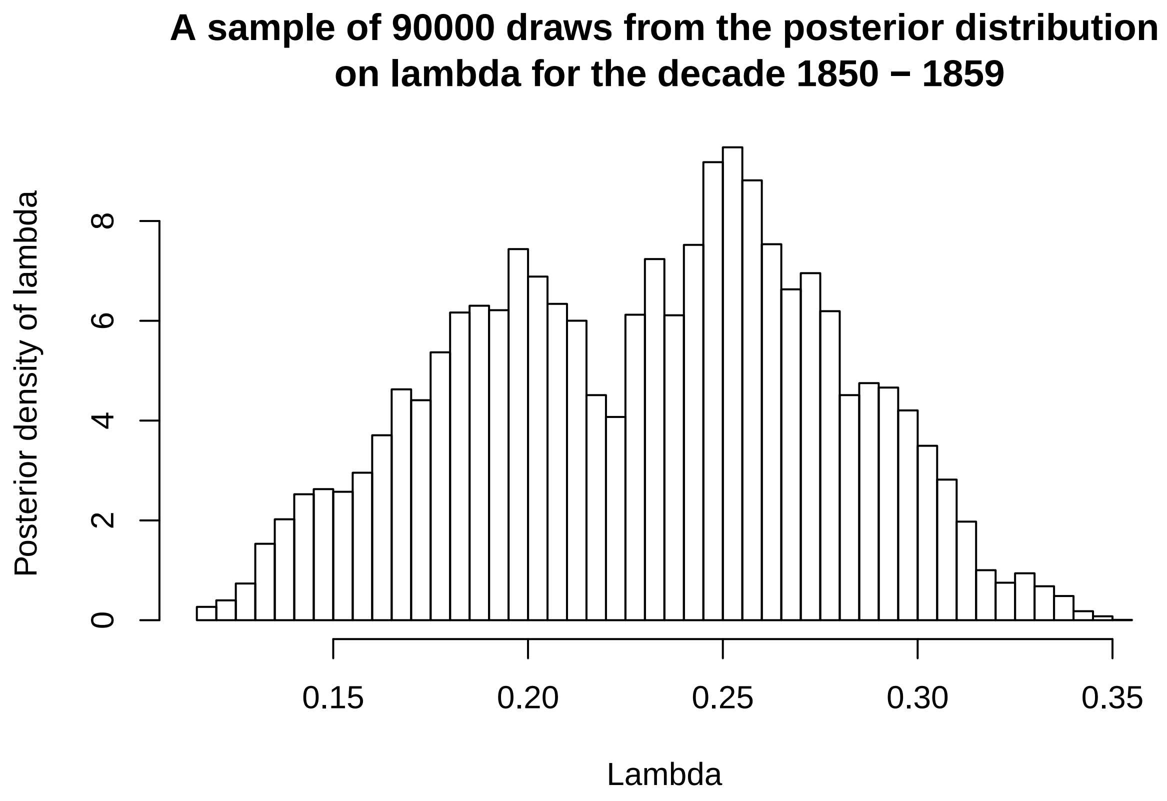
```
sd_val2 <- sd(vals2); sd_val2
```

```
## [1] 0.04721831
```

The difference before discard and after discard is not very big.

(f) Plot histogram

```
hist(vals2, breaks = 60, probability = TRUE,
     xlab = "Lambda",
     ylab = "Posterior density of lambda",
     main = "A sample of 90000 draws from the posterior distribution\n on lambda for the decade 1850 - 
```



Extend Part

(a) Update proposal

```
proposal <- function(lambda){
  stopifnot(length(lambda) == 16)
  sum <- lambda + rnorm(16, mean = 0, sd = 0.001)
  # replace the element which smaller than or equal to zero with lambda
  sum[which(sum <= 0)] <- lambda[which(sum <= 0)]
  return(sum)
}
```

(b) Update posterior

```
posterior <- function(lambda, data){
  stopifnot(length(lambda) == 16)
  # take the exponential of the log-likelihood
  loglik <- rep(NA, 16)
  for (i in 1:16){
    loglik[i] <- sum(log(dpois(data[(i * 10 - 9):(i * 10)], lambda = lambda[i])))
  }
  likelihood <- exp(sum(loglik))
  # bayesian rule
  posterior <- prod(dgamma(lambda, shape = 2, scale = 0.1)) * likelihood
  return(posterior)
}
posterior(rep(0.2, 16), new_genres[1:160])
```

```
## [1] 9.630563e-41
```

(c) Generate sample

```
new_genres <- new_genres[1:160]
initial <- function(){
  return(rgamma(16, shape = 2, scale = 0.1))
}
metrostep <- function(x){
  # x is now a vector of 16 intensity values
  z <- proposal(x)
  u <- runif(1)
  ratio <- posterior(z, new_genres) / posterior(x, new_genres)
  if (u < ratio){
    accepted.val <- z
  } else {
    accepted.val <- x
  }
  return(accepted.val)
}
n <- 100000
vals <- matrix(NA, nrow = n, ncol = 16)
vals[1, ] <- initial()
for (t in 2:n){
  vals[t, ] <- metrostep(vals[t-1, ])
}
# discard first 10000
vals2 <- vals[-(1:10000), ]
# mean of 16 decades
apply(vals2, 2, mean)
```

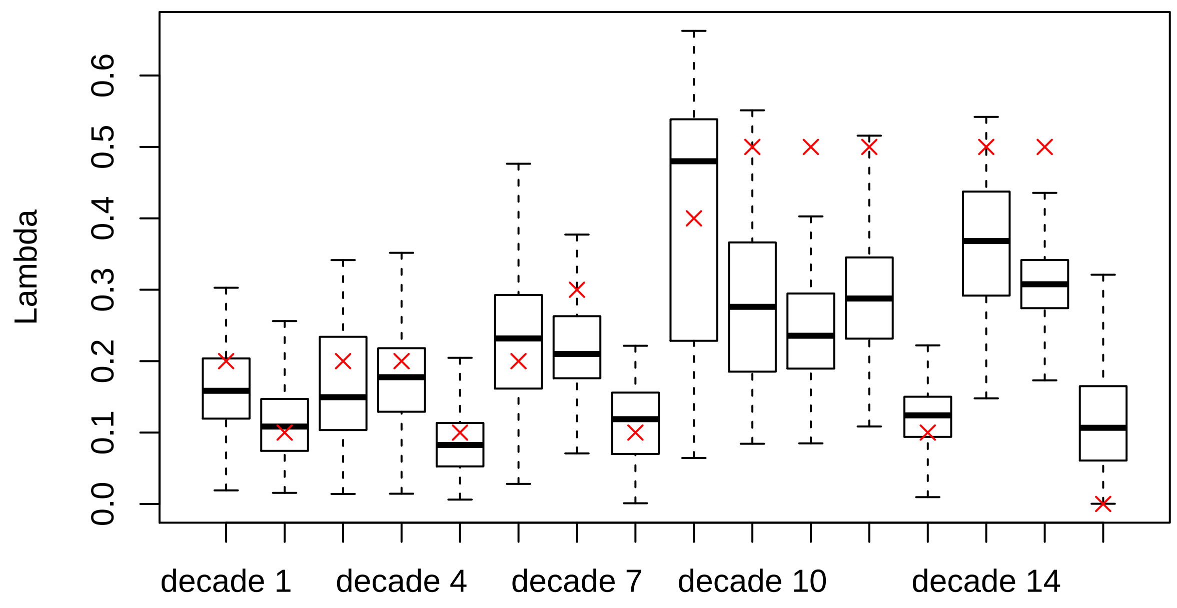
```
##  [1] 0.16060330 0.11265455 0.16422966 0.17858328 0.09052844 0.22937069
##  [7] 0.22051038 0.11421817 0.39108397 0.28776444 0.24390138 0.30032048
## [13] 0.12003913 0.35794220 0.30531293 0.11523564
```

```
# standard deviation of 16 decades
apply(vals2, 2, sd)
```

```
##  [1] 0.05923566 0.04795988 0.07648576 0.08077629 0.05249659 0.10280633
##  [7] 0.06199966 0.05039080 0.18090776 0.10935567 0.06608175 0.09465995
## [13] 0.04209896 0.09573292 0.05157550 0.06503853
```

(d) Plot

```
vals2 <- as.data.frame(vals2)
colnames(vals2) <- paste("decade","1":"16")
boxplot(vals2, outline = FALSE, ylab = "Lambda")
points(1:16, lambdamle, col = "red", pch = 4)
```


Lambda
decade 1
decade 4
decade 7
decade 10
decade 14