STAT GR5206 Homework 6 [100 pts]
Due 8:00pm Monday, November 14 on Canvas

Your homework should be submitted on Canvas as an `R` Markdown file. **Please submit the knitted .pdf file** along with the .Rmd file. We will not accept any other formats. Please clearly label the questions in your responses and support your answers by textual explanations and the code you use to produce the result. Note that you cannot answer the questions by observing the data in the "Environment" section of `R`Studio or in Excel – you must use coded commands. We will print out your homeworks. Please do not waste paper by printing the dataset or any vector over, say, length 20.

Goals: More practice with simulations. Summarizing data using distributions and estimating parameters.

The file `moretti.csv` contains data compiled by the literary scholar Franco Moretti on the history of genres of novels in Britain between 1740 and 1900 (Gothic romances, mystery stories, stories, science fiction, etc.). Each record shows the name of the genre, the year it first appeared, and the year it died out.

It has been conjectured that that genres tend to appear together in bursts, bunches, or clusters. We want to know if this is right. We will simulate what we would expect to see if genres really did appear randomly, at a constant rate – a Poisson process. Under the assumption, the number of genres which appear in a given year should follow a Poisson distribution with some mean λ, and every year should be independent of every other.

Part 1

i. Assume the variables $x_1, x_2, \ldots, x_n$ are independent and Poisson-distributed with mean λ then the log likelihood function is given by the following:

$$\ell(\lambda) = \sum_{i=1}^{n} \log\left(\frac{\lambda^{x_i} e^{-\lambda}}{(x_i)!}\right).$$

Write a function `poisLoglik`, which takes as inputs a single number λ and a vector `data` and returns the *log*-likelihood of that parameter value on that data. What should the value be when `data = c(1, 0, 0, 1, 1)` and $\lambda = 1$?

ii. Write a function `count_new_genres` which takes in a year, and returns the number of new genres which appeared in that year: 0 if there were no new genres that year, 1 if there was one, 3 if there were three, etc. What should the values be for 1803 and 1850?

iii. Create a vector, `new_genres`, which counts the number of new genres which appeared in each year of the data, from 1740 to 1900. What positions in the vector correspond to the years 1803 and 1850? What should those values be? Is that what your vector `new_ genres` has for those years?

iv. Plot `poisLoglik` as a function of λ on the `new_ genres` data. (If the maximum is not at $\lambda = 0.273$, you're doing something wrong.)

v. Use `nlm()` to maximize the log likelihood to check the $\lambda = 0.273$ value suggested in the previous question. Hint: you may need to rewrite your function from (i.) with some slight alterations.

vi. To investigate whether genres appear in bunches or randomly, we look at the spacing between genre births. Create a vector, `intergenre_intervals`, which shows how many years elapsed between new genres appearing. (If two genres appear in the same year, there should be a 0 in your vector, if three genres appear in the same year your vector should have two zeros, and so on. For example if the years that new genres appear are $1835, 1837, 1838, 1838, 1838$ your vector should be $2, 1, 0, 0$.) What is the mean of the time intervals between genre appearances? The standard deviation? The ratio of the standard deviation to the mean, called the **coefficient of variation**? Hint: The `diff()` function might help you here. Check out `?diff`.

vii. For a Poisson process, the coefficient of variation is expected to be around 1. However, that calculation doesn't account for the way Moretti's dates are rounded to the nearest year, or tell us how much the coefficient of variation might fluctuate. We will handle both of these by simulation.

a. Write a function which takes a vector of numbers, representing how many new genres appear in each year, and returns the vector of the intervals between appearances. Check that your function works by seeing that when it is given `new_genres`, it returns `intergenre_intervals`.

b. Write a function to simulate a Poisson process and calculate the coefficient of variation of its inter-appearance intervals. It should take as arguments the number of years to simulate and the mean number of genres per year. It should return a list, one component of which is the vector of inter-appearance intervals, and the other their coefficient of variation. Run it with 141 years and a mean of 0.273; the mean of the intervals should generally be between 3 and 4.

viii. Run your simulation 100,000 times, taking the coefficient of variation (only) from each. (This should take less than two minutes to run.) What fraction of simulations runs have a higher coefficient of variation than Moretti's data?

ix. Explain what this does and does not tell you about the conjecture that genres tend to appear together in burst?

Part 2

More subtle tests than the one used above indicate that there is some evidence of bursts of genre formation. To further investigate this, we look at more complex models. In an inhomogeneous Poisson process, the number of new events during year t follows a Poisson distribution with mean λ_t, where λ_t is called the intensity during that year.

To keep things simple, we will assume that the intensity is constant over each decade of our data, but can switch from decade to decade. (That is, all years from 1740 to 1749 have one intensity, 1750 to 1759 another, etc.) There are thus 16 decades covered by the data. (For these purposes we ignore 1900).

x. We first find the maximum likelihood estimate of the 16 decades' intensities. Note that the log likelihood in this case is given by the following:

$$\ell(\lambda_1, \lambda_2, \ldots, \lambda_{16}) = \sum_{d=1}^{16} \sum_{i=1}^{10} \log\left(\frac{\lambda^{x_{d,i}} e^{-\lambda}}{(x_{d,i})!}\right),$$

where $x_{d,1}, x_{d,2}, \ldots, x_{d,10}$ are independent and Poisson-distributed with mean λ_d for $d = 1, 2, \ldots, 16$. Note then that the MLE estimators of $\lambda_1, \lambda_2, \ldots, \lambda_{16}$ are given by

$$\hat{\lambda}_1^{MLE} = \frac{1}{10}\sum_{i=1}^{10} x_{1,i}, \quad \hat{\lambda}_2^{MLE} = \frac{1}{10}\sum_{i=1}^{10} x_{2,i}, \quad \ldots, \quad \hat{\lambda}_{16}^{MLE} = \frac{1}{10}\sum_{i=1}^{10} x_{16,i},$$

or in other words the MLE estimators in each decade are given by the average of the data in that decade. Some easy calculus should be able to show you this is true. Using the above, report the MLE of λ for each decade, and the code you used to find it.

xi. Even if intensities vary over decades, it seems implausible that they would all be very wildly different from each other. We use a **prior distribution** $p(\lambda)$ to express our sense of what we think the intensities ought to be like, and then use Bayes' rule to update this. To sample from the resulting posterior distribution we will use the Metropolis algorithm, which is close to, but not quite, the rejection method.

   a. Our prior distribution for λ is a gamma distribution with a shape of 2 and scale 0.1. Plot the density, $p(\lambda)$. Is the most-likely uniform rate you found in Part 1 near the peak of the prior?

   b. Write a function, `initial`, which takes no arguments, and returns a single draw from the prior distribution.

   c. Next we will want to write an `proposal` function. This should take as its argument a value for λ, and return a small random perturbation to it. Write this function, with a single input λ and a single output value. The output value should be λ plus Gaussian noise (with mean

0 and standard deviation .001) if this sum remains greater than zero, otherwise the return value should be the input λ.

d. Write a function, `posterior`, to calculate the product of the prior density for λ and the likelihood (not the log-likelihood) of the data using that value of λ. It should take two arguments, the value of λ and the vector of data, and return a single number. For example,

```
posterior(0.2, new_genres)
[1] 3.987883e+22
```

e. Use the following code, your `initial` function, your `proposal` function, and your `posterior` function, to generate a sample of 100,000 draws from the posterior distribution for λ for the decade 1850 – 1859. You will notice that the Metropolis algorithm is very like the rejection method for generating random variables, but not quite. In particular, in each step of the algorithm we consider a new proposed value and we either accept or reject that proposal. In this method we accept the proposal if the proposal has a larger posterior density than the previous estimate and we sometimes accept proposals if the proposal has a smaller posterior density than the previous estimate – it's very likely we accept it if the posterior density of the proposal is only a little below the posterior density of the previous estimate, and we never accept it if the posterior density is 0.

First create a new vector `new_genres12` (we're considering the 12^{th} decade) which counts the number of new genres which appeared in each year from 1850 to 1859 and then run the algorithm with the following code.

```
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])
}
```

Discard the first 10,000 values from `vals` as "burn in", and report the mean and standard deviation of the last 90,000. How different are the mean and standard deviation if you don't discard the first 10,000?

f. Plot a histogram of the retained output from the Metropolis method, which is a sample of 90,000 draws from the posterior distribution on λ for the decade 1850 – 1859. Label the plot appropriately.

1. We now extend this method from one decade to sixteen.

   a. Update your `proposal` function to now take as input, *lambda*, which is a vector of length 16. Again the function should return a small perturbation to λ, this time by adding a vector of Gaussian noise (with mean 0 and standard deviation .001). The output should have no elements taking values below zero (for those elements, return the input *lambda* value).

   b. Update your `posterior` function to calculate the product of the prior density for a vector of λ values and the likelihood (not the log-likelihood) of the data using the vector of λ values. It should take two arguments, the vector λ and the vector of data, and return a single number. Note that we assume each element in the λ vector has a prior which is a gamma distribution with a shape of 2 and scale 0.1 and that data is a vector containing the number of new genres in each year of the 16 decades (in order). For example,

```
posterior(rep(0.2, 16), new_genres[1:160])
[1] 9.630563e-41
```

c. Now we generate a sample from the posterior distribution of intensity *vectors*, i.e., our result is a matrix of n rows and 16 columns, each column being a sample for the intensity for a different decade. Again you should generate $100,000$ draws with the following code, discard the first $10,000$ as burn-in, and report the mean and standard deviation for each decade's intensity.

```
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, ])
  }
```

d. Use `boxplot()` to make a summary display of the posterior distributions for each decade. Compare them to the maximum likelihood intensity estimates by adding the MLE estimates to the plot using `points()`. Color the MLE estimates red. Hint: you will find the graph easier to read with `outline=FALSE`.