Lab02 - HCMV

Jonathan Stuart

3/14/2018

Spacings and the Expoential and the Gamma
Gamma(2, 3, 4, λ) will probably be more accurate at finding clusters than the exponential(λ) because of dependence issues. Summing 2, 3, or 4 intervals will probably bring dependence; they'll be large or small together. And in this case, the dependence might be helpful.

Other Notes
The plots and figures used in the text are good ideas; it's ok to use the ideas contained them, but they can't be explicityly reproduced for the report.

Definitely do some CI work; and inside the $I(\lambda)$ expression will be $\hat{\lambda}$, which is normal when producing confidence intervals; we usually estimate the parameter.

Picking the right size for the intervals is definitely a big part of the lab. He says that the Poisson is actually not applicable, or that for certain sizes it's not applicable. He said it shouldn't be hard to find a size for which the Poisson is not applicable. And when it is not, the $chi-square$ and other tests will indicate that. He also said that it would kind of be cheating to find just the right interval size and just the right parameter to make the Poisson work. His advice was to write a function where the interval size would be an input, and have that function run all the tests and give the results for that choice of interval size.

He said that the Max Number of Hits section is "one way to do it," it's one tool, but then so is the exponential/gamma discussion, and other things. Also, so is just the counts of occurences. There are many ways to make an arguement here, but the idea is to make a convincing argument to our boss about how they should proceed with the study.

Background
- Biologists conjecture that clusters of palindromes in HCMV may serve the same role as single long palindromes in Herpes simplex, i.e. indicators of origins of replication.
- HCMV DNA is 229, 354 base pairs long, with 296 total palindromes
- Infection rates, vulnerable populations, pathology of the virus. Genomics and other modern advances, application of knot theory to 3D stucture and molecular dynamics of DNA.

Investigations
1. How do we find clusters of palindromes?
2. How do we determine whether a cluster is a chance occurrence or an indication of a replication site?
- Use the answers to these questions to advise a scientist about to start a search for an origin of replication
- Pg. 82 - Discussion of interval size is key; does the interval with the largest number of palindromes indicate a replication site? How do you know if your interval size is accurate or leads to indicative results? What happens if it's too big? Too small?

Theory
- There are many properties of the homogeneous Poisson process that can be used to check how well this reference model first the DNA data. So, we're essentially asking of the Poisson process is an accurate distribtuion to apply to the data.
- You can compare the observed counts to the expected number of counts per region. This leads to the *chi − square Goodness-of-Fit* for this probability distribution (this is an hypothesis test). - We could potentially seek to optimize the interval size through goodness of fit testing. - Another option would be to use a residual plot to measure goodness-of-fit.
- Another option is to compare the actual locations of palindromes with the expected locations of palindromes from a uniform distribution. - Does the distance between successive hits follow the expoential distribution, as they should? Does the distance between, 2, 3, 4, etc hits follow a gamma distribution, as they should?

- We can use the distribution to find the projected maximum number of hits (this an hypothesis test.) - We can have a discussion of using the method of moments and the method of maximum likelihood to get the estimate of λ, $\hat{\lambda}$. - From the likelihood estimates we can compute the information, and use this to construct confidence intervals. - We can conduct a final hypothesis test with a null hypothesis and an alternative hypothesis test, using a *z statistic*

Extensions This is where you'll find the basis of the winning approach. It's moving a window of some size along the DNA, one base pair at a time, and finding the location of the window with the largest palindrome count.

Potential Plots (4)
Pg 81 - Random plots of 296 locations between 1 and 229, 354
Pg 82 - Compare spacings between palindromes, sums of pairs of spacings, triplets to what you would expect from a random scatter

Potential Tables(2)
-Pg 82 - Counts of palindromes in nonoverlapping regions compared to what you expect from a uniform random scatter. *Show* that the counts for short regions will be more variable than the counts for larger regions.
-

**To begin, pursue the point of view that structure in the data is indicated by departures from a univorm scatter of palindroms across the DNA.

Let's mount the necessary packages

```
library('dplyr')
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Let's read in the data

```
data <- read.csv("hcmv.csv")
```

Let's create 100 instances of random scatters of palindromes.

```
set.seed(3)
random_palindromes <- matrix(NA , 296, 100)
for (i in 1:100) {
  random_palindromes[ , i] <- sort(sample(1:229354, 296, replace = FALSE))
}
```

functions to produce counts with variable interval size and data set

```
#funtion to calculate the true length of the vector of palindrome counts
true_length <- function(x) {
  floor(229354/x)
}

#function that counts the number of palindromes and takes input of interval size
palindrome_count <- function(x) {
  counts <- 0
```

```r
  iterator <- seq(from = 1, to = 229354, by = x)
  counts[1] <- nrow(filter(data, data[, 1] < x))
  for (i in 1:length(iterator)) {
    counts[i + 1] <- nrow(filter(data, data[, 1] > (iterator[i + 1] - iterator[1]) &
                                 data[, 1] < iterator[i + 2] - iterator[1]))
  }
  counts <- counts[1:true_length(x)]
  return(counts)
}

#function that counts the number of palindromes and takes input of interval size
#and data set
super_palindrome_count <- function(data, x) {
  counts <- 0
  iterator <- seq(from = 1, to = 229354, by = x)
  counts[1] <- nrow(filter(data, data[, 1] < x))
  for (i in 1:length(iterator)) {
    counts[i + 1] <- nrow(filter(data, data[, 1] > (iterator[i + 1] - iterator[1]) &
                                 data[, 1] < iterator[i + 2] - iterator[1]))
  }
  counts <- counts[1:true_length(x)]
  return(counts)
}
```

Goodness of Fit for Poisson with interval size 500

```r
#goodness of fit code

#counting number of palindromes in each interval
counts <- palindrome_count(500)
counts
```

```
##   [1] 1 0 3 0 0 0 3 0 0 0 0 0 0 0 1 0 0 0 3 0 0 1 0 1 0 1 0 0 1 1 0 0 2 2 0
##  [36] 0 1 0 3 0 1 1 0 0 1 2 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 0 1 0 1 0 0 3 1
##  [71] 0 0 0 2 0 0 0 1 0 0 0 1 2 0 1 0 1 1 0 0 1 0 0 0 0 1 2 1 0 0 0 0 2 0 1
## [106] 1 1 1 3 0 1 0 0 1 1 0 0 0 0 0 2 1 1 0 0 1 2 2 0 1 0 3 1 0 0 1 1 0 0 1
## [141] 0 1 1 0 1 1 0 0 2 1 0 4 2 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 3 2 0 0
## [176] 1 0 1 0 1 1 1 1 2 0 8 2 2 1 0 0 1 0 0 1 0 1 1 0 1 0 1 0 0 2 1 0 0 1 1
## [211] 0 1 0 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 3 0 0 1 0 0 1 0 3 2 0 1 0 0
## [246] 0 0 0 0 1 0 1 0 1 2 1 0 1 1 1 0 0 1 1 0 0 1 0 1 0 1 0 2 2 1 2 1 0 1 0
## [281] 0 1 1 1 1 1 1 3 0 0 0 0 0 1 0 3 0 2 0 0 1 0 1 1 3 0 0 0 1 0 1 1 0 0 0
## [316] 1 0 0 0 0 0 0 2 0 0 3 0 2 1 0 1 3 1 0 0 0 1 2 0 0 1 2 0 1 0 0 0 1 4 0
## [351] 0 0 0 0 0 2 0 1 0 0 3 0 0 0 1 0 0 0 0 0 0 0 3 0 0 1 2 0 1 1 0 3 1 0 0
## [386] 1 1 1 1 0 6 1 0 1 2 0 1 1 0 0 0 0 2 0 1 0 0 0 0 1 0 1 0 0 0 3 0 1 0 1
## [421] 1 0 0 0 0 0 0 0 0 0 0 1 2 1 1 0 0 0 0 0 0 1 0 2 1 2 1 1 0 1 0 1 0 1 3
## [456] 0 1 1
```

```r
#estimating lambda_hat as x_bar
lambda <- sum(counts) / length(counts)
lambda
```

```
## [1] 0.6441048
```

```r
#calculating the observed values (this doesn't account for intervals with zero palindromes)
count_table <- table(counts)
count_table
```

```
## counts
##   0   1   2   3   4   6   8
## 255 145  34  20   2   1   1
```

```r
observed <- c(sum(count_table[1]), count_table[2], count_table[3], sum(count_table[4:7]))

observed <- unname(observed)
observed
```

```
## [1] 255 145  34  24
```

```r
expected <- 0
expected
```

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

```r
#calculating expected values
expected[1] <- length(counts) * dpois(0, lambda)
expected[2] <- length(counts) * dpois(1, lambda)
expected[3] <- length(counts) * dpois(2, lambda)
expected[4] <- length(counts) * sum(dpois(3:8, lambda))

#calculating the chi-square test statistic
chi_test_statistic <- sum((observed - expected)^2 / expected)

#calculating the degrees of freedom
degrees_of_freedom <- length(expected) - 2

1 - dchisq(chi_test_statistic, degrees_of_freedom)
```

```
## [1] 0.9998795
```

```r
#calculating risiduals
  (observed - expected) / sqrt(expected)
```

```
## [1]  0.9342900 -0.7965378 -2.2497146  3.1770151
```

```r
#code for the maximum count
1 - (sum(dpois(0:7, lambda)))^500
```

```
## [1] 0.0002076464
```

Goodness of Fit for Poisson with interval size 2500

```r
#goodness of fit code

#counting number of palindromes in each interval
counts <- palindrome_count(2500)
counts
```

```
##  [1]  4  3  1  3  2  3  4  4  3  3  1  1  3  5  2  1  4  2  1  4  3  6  3
## [24]  0  4  6  4  3  3  4  6  2  0  1  6  3  5 13  2  3  3  3  2  2  1  2
## [47]  4  4  3  1  4  4  2  2  6  4  4  5  1  5  6  1  2  1  2  6  5  3  4
## [70]  5  0  3  4  0  3  5  4  4 10  2  3  1  2  5  1  0  5  0  4  5  5
```

```r
#estimating lambda_hat as x_bar
lambda <- sum(counts) / length(counts)
lambda
```

```
## [1] 3.230769
```

```
#calculating the observed values (this doesn't account for intervals with zero palindromes)
count_table <- table(counts)
count_table
```

```
## counts
##  0  1  2  3  4  5  6 10 13
##  6 13 14 19 19 11  7  1  1
```

```
observed <- c(count_table[1], count_table[2], count_table[3], count_table[4], count_table[5], count_tab

observed <- unname(observed)
observed
```

```
## [1]  6 13 14 19 19 11  9
```

```
expected <- 0

#calculating expected values
expected[1] <- length(counts) * dpois(0, lambda)
expected[2] <- length(counts) * dpois(1, lambda)
expected[3] <- length(counts) * dpois(2, lambda)
expected[4] <- length(counts) * dpois(3, lambda)
expected[5] <- length(counts) * dpois(4, lambda)
expected[6] <- length(counts) * dpois(5, lambda)
expected[7] <- length(counts) * sum(dpois(6:13, lambda))


#calculating the chi-square test statistic
chi_test_statistic <- sum((observed - expected)^2 / expected)

#calculating the degrees of freedom
degrees_of_freedom <- length(expected) - 2

1 - dchisq(chi_test_statistic, degrees_of_freedom)
```

```
## [1] 0.8498018
```

```
#calculating risiduals
  (observed - expected) / sqrt(expected)
```

```
## [1]  1.2670453  0.4045342 -1.1014653 -0.2705245  0.6610989  0.1382981
## [7] -0.2900789
```

```
#code for the maximum count
1 - (sum(dpois(0:12, lambda)))^2500
```

```
## [1] 0.08214843
```

Goodness of Fit for Poisson with interval size 4000

```
  #counting number of palindromes in each interval
  counts <- super_palindrome_count(data, 4000)
  counts
```

```
##  [1]  7  1  5  3  8  6  1  4  5  3  6  2  5  8  2  9  6  4  9  4  1  7  7
## [24] 14  4  4  4  3  5  5  3  6  5  3  9  9  4  5  6  1  7  6  7  5  3  4
## [47]  4  8 11  5  3  6  3  1  4  8  6
```

```r
#estimating lambda_hat as x_bar
lambda <- sum(counts) / length(counts)
lambda
```

```
## [1] 5.157895
```

```r
#calculating the observed values (this doesn't account for intervals with zero palindromes)
count_table <- table(counts)
count_table
```

```
## counts
##  1  2  3  4  5  6  7  8  9 11 14
##  5  2  8 10  9  8  5  4  4  1  1
```

```r
observed <- c(sum(count_table[1:2]), count_table[3], count_table[4],
              count_table[5], count_table[6], count_table[7],
              count_table[8], sum(count_table[9:length(count_table)]))

observed <- unname(observed)
observed
```

```
## [1]  7  8 10  9  8  5  4  6
```

```r
expected <- 0

#calculating expected values
expected[1] <- length(counts) * sum(dpois(0:2, lambda))
expected[2] <- length(counts) * dpois(3, lambda)
expected[3] <- length(counts) * dpois(4, lambda)
expected[4] <- length(counts) * dpois(5, lambda)
expected[5] <- length(counts) * dpois(6, lambda)
expected[6] <- length(counts) * dpois(7, lambda)
expected[7] <- length(counts) * dpois(8, lambda)
expected[8] <- length(counts) * sum(dpois(9:14, lambda))

#calculating the chi-square test statistic
chi_test_statistic <- sum((observed - expected)^2 / expected)

#calculating the degrees of freedom
degrees_of_freedom <- length(expected) - 2

1 - dchisq(chi_test_statistic, degrees_of_freedom)
```

```
## [1] 0.960252
```

```r
#calculating risiduals
(observed - expected) / sqrt(expected)
```

```
## [1]  0.24455726  0.18234910  0.10552502 -0.30938506 -0.19699668 -0.52501255
## [7] -0.03697505  0.71871337
```

```r
  #code for the maximum count
1 - (sum(dpois(0:13, lambda)))^4000
```

```
## [1] 0.976261
```

Goodness of Fit for Poisson with interval size 7000

```
#goodness of fit code

#counting number of palindromes in each interval
counts <- palindrome_count(7000)
counts
```

```
##  [1]  7  7 12  5  9  6  8 10  7 12 13  3 11 18  9  5  7  8  9 12 10 11  4
## [24] 13 12  6  7 17  8  8  5  9
```

```
#estimating lambda_hat as x_bar
lambda <- sum(counts) / length(counts)
lambda
```

```
## [1] 9
```

```
#calculating the observed values (this doesn't account for intervals with zero palindromes)
count_table <- table(counts)
count_table
```

```
## counts
##  3  4  5  6  7  8  9 10 11 12 13 17 18
##  1  1  3  2  5  4  4  2  2  4  2  1  1
```

```
observed <- c(sum(count_table[1:2]), count_table[3], count_table[4], count_table[5], count_table[6], co

observed <- unname(observed)
observed
```

```
##  [1] 2 3 2 5 4 4 2 2 4 2 2
```

```
expected <- 0

#calculating expected values
expected[1] <- length(counts) * sum(dpois(0:4, lambda))
expected[2] <- length(counts) * dpois(5, lambda)
expected[3] <- length(counts) * dpois(6, lambda)
expected[4] <- length(counts) * dpois(7, lambda)
expected[5] <- length(counts) * dpois(8, lambda)
expected[6] <- length(counts) * dpois(9, lambda)
expected[7] <- length(counts) * dpois(10, lambda)
expected[8] <- length(counts) * dpois(11, lambda)
expected[9] <- length(counts) * dpois(12, lambda)
expected[10] <- length(counts) * dpois(13, lambda)
expected[11] <- length(counts) * sum(dpois(14:18, lambda))

#calculating the chi-square test statistic
chi_test_statistic <- sum((observed - expected)^2 / expected)

#calculating the degrees of freedom
degrees_of_freedom <- length(expected) - 2

1 - dchisq(chi_test_statistic, degrees_of_freedom)
```

```
## [1] 0.9365195
```

```
#calculating risiduals
  (observed - expected) / sqrt(expected)
```

```
##  [1]  0.1818439  0.7580583 -0.5358681  0.6468737 -0.1052827 -0.1052827
##  [7] -0.9212505 -0.6269253  1.0954052  0.3055745 -0.1888988
```

```r
#code for the maximum count
1 - (sum(dpois(0:17, lambda)))^7000
```

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

Code for lambda hat; discuss in the context of method of moments and maximum likelihood

```r
lambda_hat <- mean(counts)
lambda_hat
```

```
## [1] 9
```

```r
#information and confidence interval code
#exponential and gamma for distance between hits code
#maximum count code
#z-test code
#sliding window code
```

Background on the problem. HCVM family, volatility, desire to find replication sites. Replication sites appear as clusters of palindromes.

Whether or not the poisson fits the data. Does it fit the data better than random scatter?

How do we know that size interval to use?

Conducted multiple hypothesis tests to check whether a poisson distr could be effectively fit to the data.

Yes, poisson is applicable, which allows us to see where we might want to focus for clustering. We can look to the max counts data.