

Alex Hsieh - A11651288 - 4th Year Math and Economics Major
Atharva Fulay - A13140630 - 4th year Applied Math Major
Peter Yao - A12286133 - 3rd year Math-Computer Science Major
Justin Glommen - A13045994 - 2nd year Math-Computer Science Major

Case Study 3

Abstract:

Our global analysis revealed that the origin of replication should statistically be between 92,000 and 94,300 base pairs. Through our data analysis, our team believes the best way to experimentally search for the origin of replication in the human cytomegalovirus is the Pearson's Chi-Squared Test with respect to a homogeneous Poisson Distribution.

Introduction:

The human cytomegalovirus (CMV) is a disease which can be life threatening for those with weak or deficient immune systems. To combat this, scientists are searching for a place in the virus's DNA that contains instructions, represented in a coded message of letters A, C, G and T (which correspond to adenine, cytosine, guanine, and thymine respectively), for its replication. Because the origin of replication for CMV is the same as that of herpes and both are in the same family, their replication messages both represent a complementary palindrome. The palindromes for CMV can vary in range from 10 to 18 base pairs. The entire DNA sequence of CMV consists of 229,354 letters, which can be time consuming to look through if we go through the message letter by letter, so we will look for unusual clusters of palindromes in the sequence to reduce the amount of searching and testing needed to find the origin of replication.

Other similar studies done, including one by Marie Masse from Stanford University (<http://www.pnas.org/content/89/12/5246.full.pdf>), seem to reflect our sentiments: that the part of the genome with the majority of repeated sequences was in between 92,100 and 93,500 base pairs, which has been given the name of oriLyt, is the lytic origin of replication.

The different methods of determining the best way to identify clusters of palindromes include the Pearson's chi-squared test with respect to the homogeneous Poisson Distribution, a Monte Carlo simulation, cumulative distribution function for a Poisson Distribution, and the distributions, spacings, clusters, and residuals for palindrome locations in general. We use the aforementioned methods of statistical analysis in order to determine if the clusters that we observed have occurred simply due to chance, or because they represent points in the DNA where the origin of replication is.

Data:

The available data given to us consists of 296 palindrome locations that were spread throughout the 229,354 possible palindrome sites along the DNA sequence. We know that palindromes should be occurring following a uniform distribution, so we can use this knowledge

to guide us in reviewing the actualized data. Our data is used to further examine many properties, such as the general distribution of the palindromes, how many palindromes occurred over given intervals, etc., comparing those results to those of the theoretical distribution in order to check for inconsistencies, helping identify our palindrome replication site.

Random Scatter:

We know that a strand of DNA would normally have a uniform scatter of palindromes across the message. To find the origin of replication, we want to find where the message strays from a uniform scatter of palindromes to identify whether the DNA of CMV truly has clusters of palindromes. Therefore, we will perform two chi-squared tests to find unusual cluster of palindromes, one on the entire message of CMV's DNA and another on a simulated DNA message that has a uniformly scatter of palindromes.

For the simulated data, we wanted to imitate what the DNA would look like with uniformly scattered palindromes, so we generated 296 palindrome locations in a message of length 229,354, the same amount of letters in CMV's DNA. Figure 2 shows the locations of every palindrome in the simulated message. We then separate the message into 50 non-overlapping bins and perform a chi-squared test. After combining bins that have fewer than 5 palindromes in them, we get that the probability of our simulated data having a test statistic at least as large as ours under the random scatter model is 0.7828 under 5 degrees of freedom, which is reasonably likely.

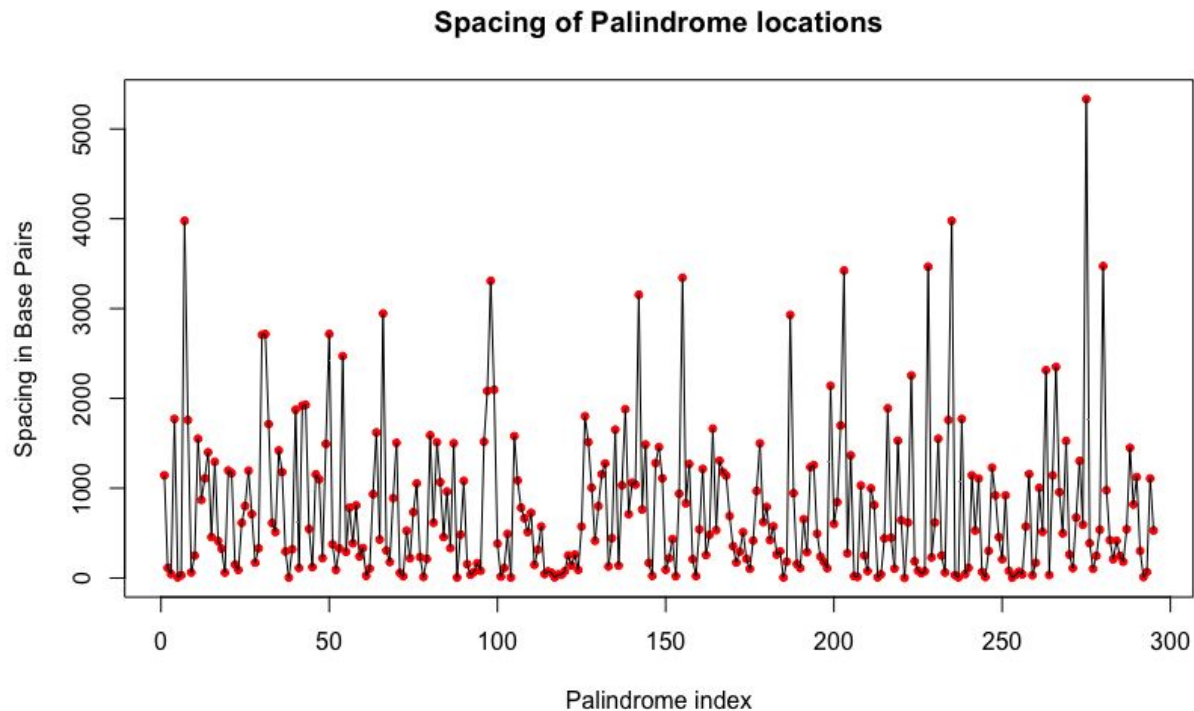
For the CMV data, we will also separate the DNA message into 50 non-overlapping bins and perform a chi-squared test. Figure 3 shows the locations of every palindrome in the CMV message. After combining bins with fewer than 5 palindromes in them, we get that the probability of our simulated data having a test statistic at least as large as ours under the random scatter model is .2219, which is much less likely than that of our simulated data. This is sufficient evidence to suspect that the CMV data does not follow a uniform random distribution and we should further investigate.

Locations and Spacings:

As briefly mentioned in the introduction, the study conducted by Marie Masse from Stanford University concludes that the replication zone lies roughly between 92,100 and 93,500 with high probability. We want to take a look at our data and try to mirror it to this study for comparison. **Figure 4** reflects our data using a histogram, breaking the site locations into intervals and evaluating the amount of clusters within each. Similarly, **Figure 5** represents a residual plot of our data. As foreshadowed through the Stanford study, our dataset's largest cluster of palindromes occurs within nearly the same region.

The spacing of the consecutive palindromes is tested using Monte Carlo simulation. Generating several uniformly distributed sample sets, we want to contrast the average amount of consecutive palindromes within the simulation to that of the dataset to see if the more

palindromes are concentrated in one over the other, i.e. the spacing of the palindromes. The spacing however appears statistically to be similar and not abnormal, since the average from the theoretical set comes out to about 1.3 while the highest occurrence in the dataset overall is only 2. Therefore the spacing does not seem to have high weight on the indication of the replication site.



When comparing the spacing between the given data and a random scatter, the mean of the given data was 775 base pairs apart, while the mean of a random scatter was 769 base pairs apart. Medians for the given data and the random scatter are 512 and 515 respectively. The maximums for the given data and the random scatter are 5333 and 4667 respectively.

Additionally, the base pair location of the given data was on average 436 base pair larger than the random scatter data base pair location. See **Figure 6** in the Appendix.

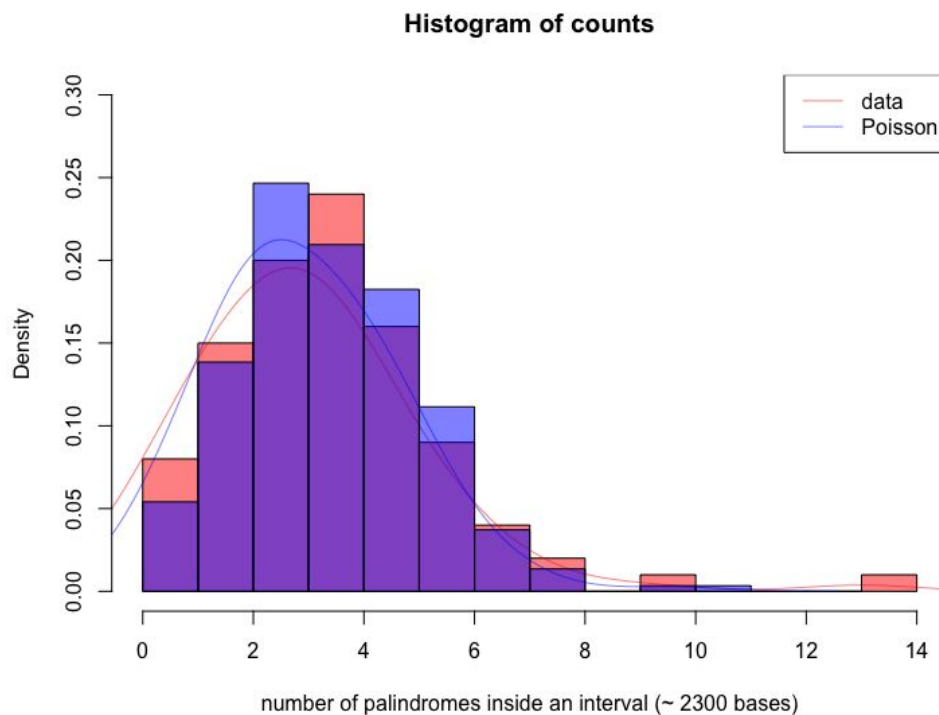
Counts:

The CMV dataset gave the location of the bases of where palindromes longer than 10 occur. The locations were categorized into intervals based on their location. From there, the count of how many intervals is displayed in red. For example, the first red bar is how many intervals contain zero palindromes, the second bar is how many intervals contain one palindrome and so on. Overlaid in blue are the theoretical values of a homogeneous Poisson distribution with λ as the mean of $n = 296$ counts. The bars that are purple are the overlapping regions. Density lines are shown as well to show the similarities in graph shape. The only major outlier is

at the very right, which is for one interval which contained 13 palindromes. This counts histogram shows that the data is very similar to that of a homogeneous Poisson distribution.

Additionally, the skewness of the sample is 1.47456, indicating that the graph is almost symmetrical but has a longer tail to the right. This is significant since we are looking for intervals with large numbers of palindromes. Additionally, the skewness of a Poisson distribution is 0.731441, respectively. This means the data we are using is less symmetrical than a Poisson distribution and more skewed to the right, further indicating there are intervals with larger numbers of palindromes.

For the goodness of fit test, the chi-squared value for counts, the number of palindromes in the intervals, is 137.78 with 99 degrees of freedom and a p-value of 0.006083. This means that we have 0.61% of finding a difference between the count and Poisson distributions that are larger than this one.



(Figure 1)

The Biggest Cluster:

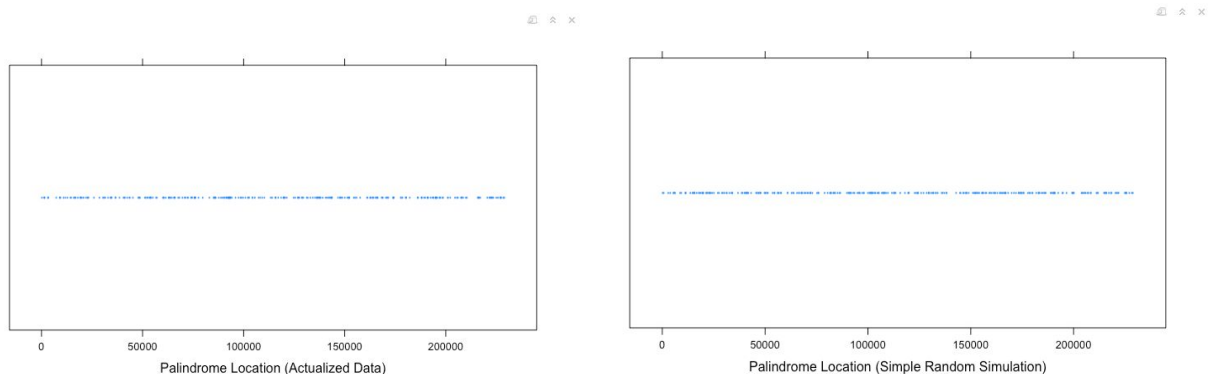
We know the the distances between successive palindromes follow the exponential distribution. Using counts from above, we have 100 non-overlapping intervals, each with a certain number of palindromes within that interval. The numbers of palindromes in the intervals are very close to Poisson distribution. Thus, the largest amount of palindromes in an interval will, “[behave] as the maximum of independent Poisson random variables” (Slide 32).

If we let $m = 100$, the probability that the maximum count over m intervals is greater than or equal to k , where k is some large arbitrary value of number of palindromes in an interval. This is equivalent to 1 minus the probability that the maximum is less than k . If the maximum is less than k , then all count values must be less than k . But the value of the first interval count should be the same as any other interval count. So we have 1 minus the probability of the first interval count is less than k , this entire quantity raised to the m th power.

Finally, this can be simplified to $1 - [\lambda^0 e^{-\lambda} + \dots + (\lambda^{k-1}) (e^{-\lambda}) / (k-1)!]^m$. But we don't know λ , so we estimate λ_{hat} . By the method of moments, we get $\lambda_{\text{hat}} = x_{\text{bar}} = 296/100 = 2.96$. Using all this information and the *qpois* built in R function, we plugged in .9995 and 2.96. According to the R documentation, "The quantile is right continuous: *qpois*(p , λ) is the smallest integer x such that $P(X \leq x) \geq p$." The integer returned was 10. This means that there is only a .0005 chance that there is a value of 10 or larger. However, one of values in count was 13 palindromes in an interval. This is a very rare and unusually high amount of palindromes in one interval. Thus, we found a cluster that had less than a .0005 chance of existing.

Analysis and Methods:

The first step to performing analysis on which site is potentially the replication site is ensuring that the data is clean and relevant. After doing so, plotting the data with strip plots, histograms, and residual plots presents us an overarching visualization of the distribution of palindromes over the entire population size. Similarly, a simple random sample of 296 from a uniform distribution mimicking that of our data is plotted and compared to the dataset in order to confirm the inference that palindromes should be uniformly distributed. **(Figures 2 and 3)**



Our next goal is to try to identify long strings of consecutive palindrome doubles, triplets, etc. in order to help lead us towards the replication site. We performed Monte Carlo simulation of uniformly generated variables from a clone distribution in order to form a baseline for what is abnormal in terms of the amount of consecutive palindromes. After obtaining the results, we perform the same analyses on the original actualized dataset for comparison. The conclusion here is undeterminable, as the amount of consecutive palindromes in our data, compared to the

theoretical distribution, does not appear abnormal. That said, we need to resort to the Poisson Process as we now want to test for abnormal occurrences of larger clusters of palindromes.

Breaking our dataset into smaller intervals, we can measure the amount of palindromes within each and watch for high amounts of palindromes. To do this, we must form a baseline for what is unusually high clustering, therefore we compare the results of the dataset to those of a poisson distribution mimicking the same rate of occurrence in **Figure 1**. Clearly, the thirteen occurrences appears to be an outlier at first glance. In order to assess this further, we find the 99.95th quantile of a theoretical poisson distribution using sample statistics and determine a threshold for what is considered the maximum likely amount of palindromes in a cluster. The result yields ten, and after finding only one cluster in our dataset with thirteen, we infer this cluster as highly improbable and thus a probable candidate for our replication site.

Conclusion:

Based off of the various tests which we conducted in this experiment, we have determined that the Pearson's Chi-Squared Test with the Homogeneous Poisson Distribution is the most effective way to determine the replication site of the CMV. The Homogeneous Poisson Distribution is an effective point distribution to emulate the uniform random scatter of palindromes on a strand of DNA, and serves as a good base to compare our chi-squared test to, and when compared with a Poisson Point Distribution, we could immediately see where our data departed from a normal distribution. Against the Poisson Poisson, the Monte Carlo simulation lacked in usefulness. While although proving useful realizing the distribution of our palindromes as uniform, it lead to little inference about consecutive palindrome's impact on finding the replication site. Spacing would be the last metric for an analyst to look at, as little can be derived from the information, and spaces between palindromes do not seem to vary much.

Our results show the interval containing 13 palindromes is (92,000, 94,300] base pairs, which agrees with <http://www.pnas.org/content/89/12/5246.full.pdf>, a study done at Stanford which confirmed the origin of replication for the CMV. Therefore, we would recommend for the biologist to search within that region of the DNA site based on our statistical analysis.

Theory:

Pearson's Chi-Squared Test:

Pearson's chi-squared test is usually applied to sets of categorical data to determine if observed differences in the set of data were statistically probable or if they just arose by chance. It is mainly used when analyzing unpaired data from large samples. There is a null hypothesis which is tested, which states that the frequency distribution of certain events which were observed in our sample is consistent with another particular theoretical distribution, in this case, the Homogeneous Poisson Distribution (see below). In order for an event to satisfy the Pearson's Chi-Squared Test, both events must be mutually exclusive, and have a total probability of one.

The Pearson's Chi-Squared Test can be used for three different types of comparison: *goodness of fit*, which indicates whether or not our frequency distribution differs from a theoretical one, *homogeneity*, which compares distribution of counts for different groups using the same categorical variables, or a *test of independence*, which measures whether or not unpaired observations are independent of each other. For the purposes of this case study, we will only be using the Chi-Squared Test for goodness of fit.

The steps for the Pearson's Chi-Squared Test are to first calculate the chi-squared statistic (X^2), which corresponds with the normalized sum of the squared deviations between our own and the theoretical frequencies. We then determine the number of degrees of freedom, which in our case was 99, which is the number of categories reduced by the total number of parameters in the fitted distribution. Afterwards, we test against a certain confidence level, which in our case was .05. Since our X^2 value was much less than the critical value of 124, we can say with 95% confidence that our observed frequency distribution is the same as the theoretical distribution and accept our original null hypothesis.

Standardized Residual Test:

A standardized residual test is a form of regression analysis in which we analyze the quotient obtained when dividing a residual by an estimate of its standard deviation. These typically vary significantly between data points, even if all the errors contain the same standard deviation, which is why it becomes necessary in order to standardize the data. This makes it an important tool for statisticians to detect outliers.

Homogeneous Poisson Distribution

In statistics, a Poisson point process corresponds to a random mathematical object which has random points located in a theoretical space, and is commonly used to model for uniform random scatter. In our case study, if we look at the strand of DNA that we are analyzing as a line, with the location of palindromes as points on the line, then palindromes should be scattered randomly and uniformly across the DNA strand. Therefore, if there is an unusual amount of palindromes at a certain point of the line, this can possibly be enough to suspect that that certain point is the origin for replication for the virus, since the process reflects the fact that any tiny piece of DNA has the same chance of having a palindrome in it as any other small strand of DNA.

The Poisson point process is usually defined on a real line as a stochastic process, and depends on a single mathematical object, in our case, the occurrence of base pair palindromes on the DNA strand. For the homogeneous or stationary Poisson point process, the constant would be known as the rate of intensity, and represents the average density of points within the points that we are observing.

Appendix:

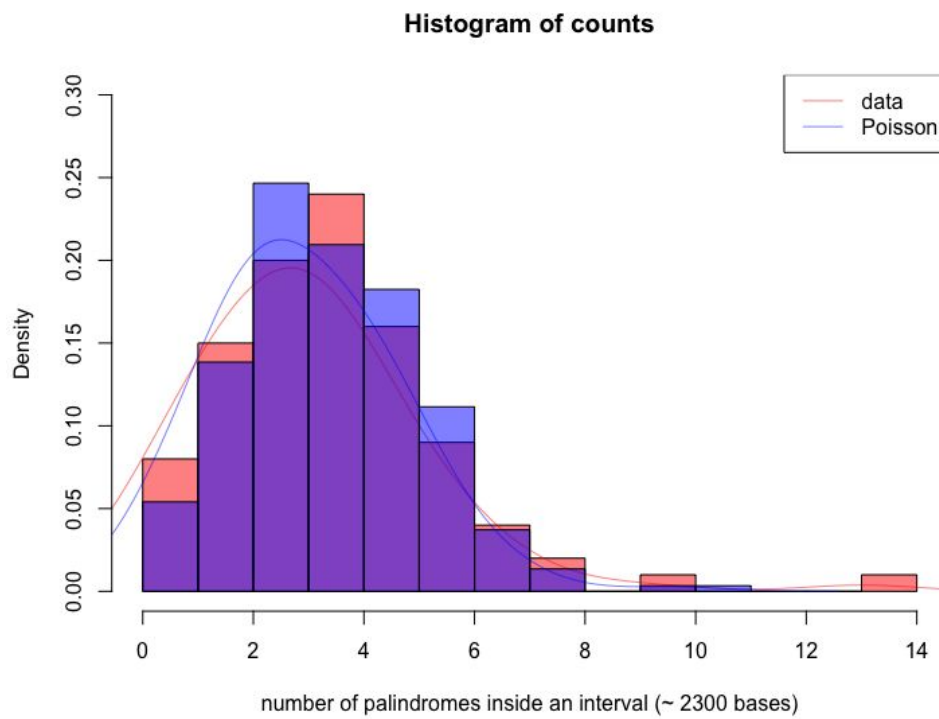


Figure 1: Interval palindrome counts

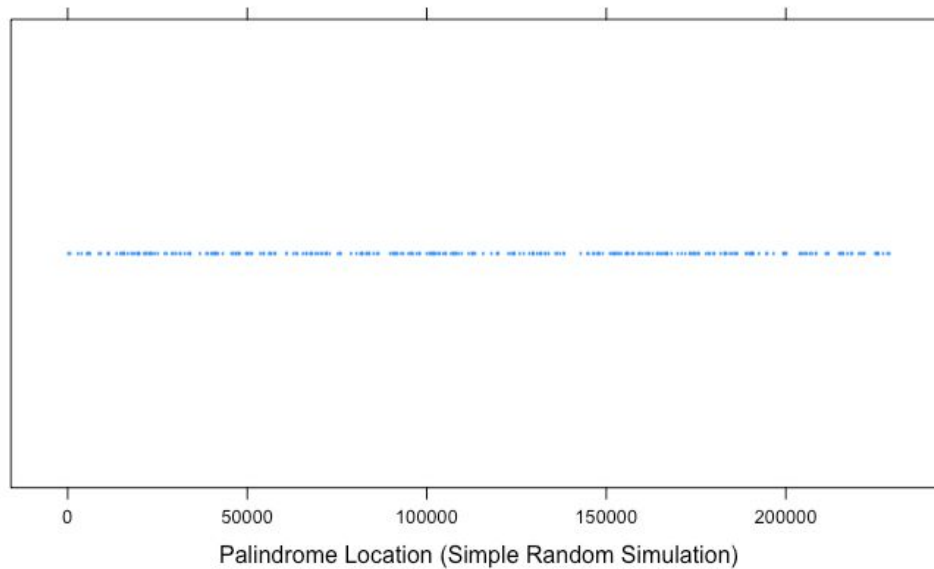


Figure 2. The locations of the palindromes in the simulated genome.

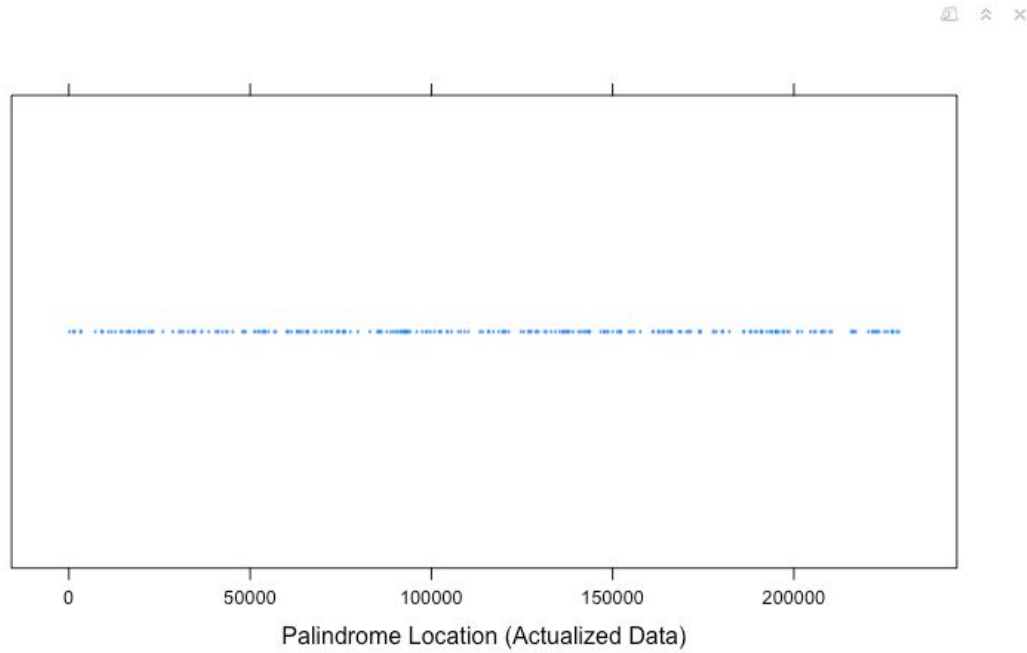


Figure 3. The locations of the palindromes in the CMV genome.

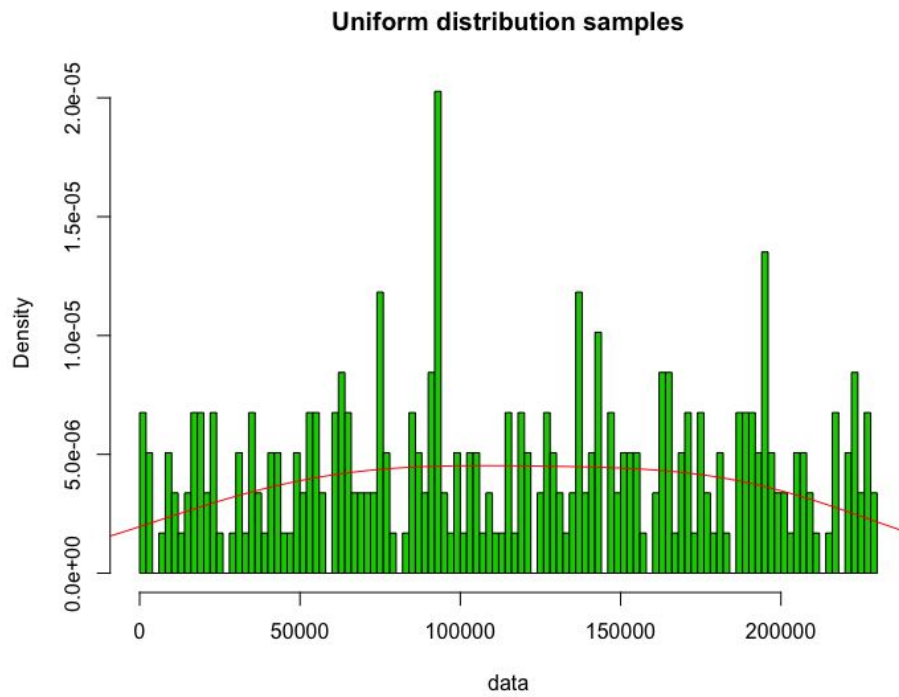


Figure 4: Palindrome occurrences within the DNA site

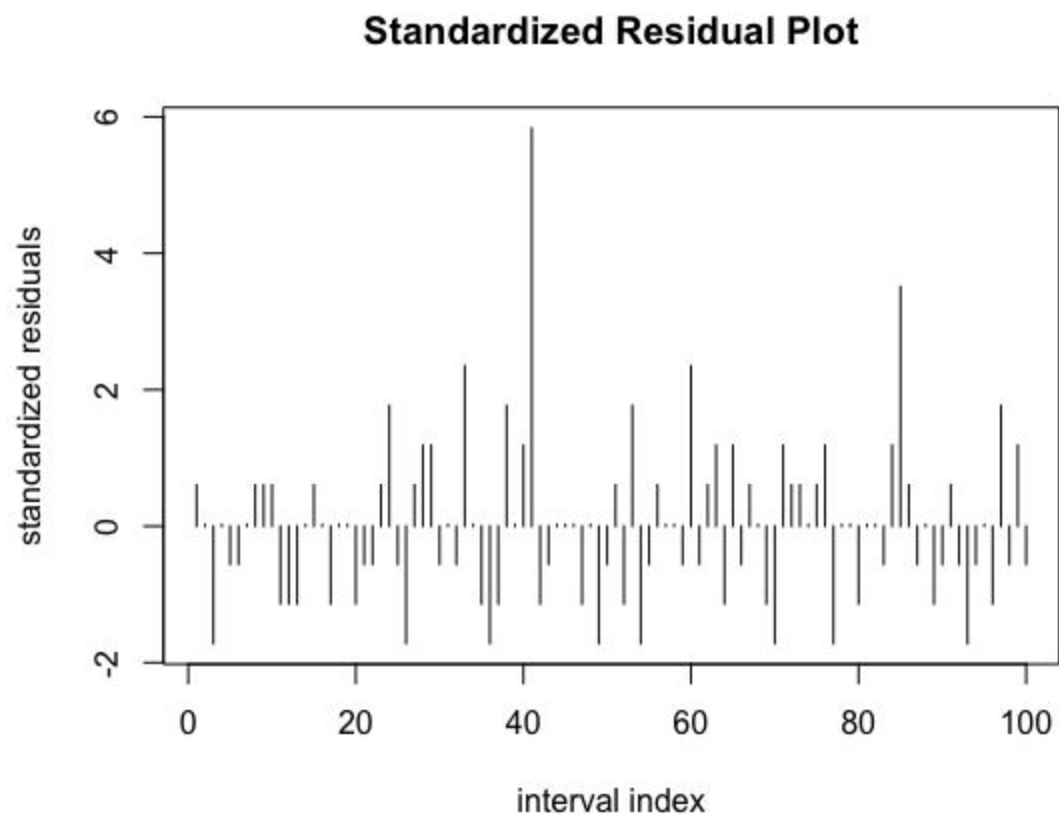


Figure 5: Residual plot of clusters of palindromes

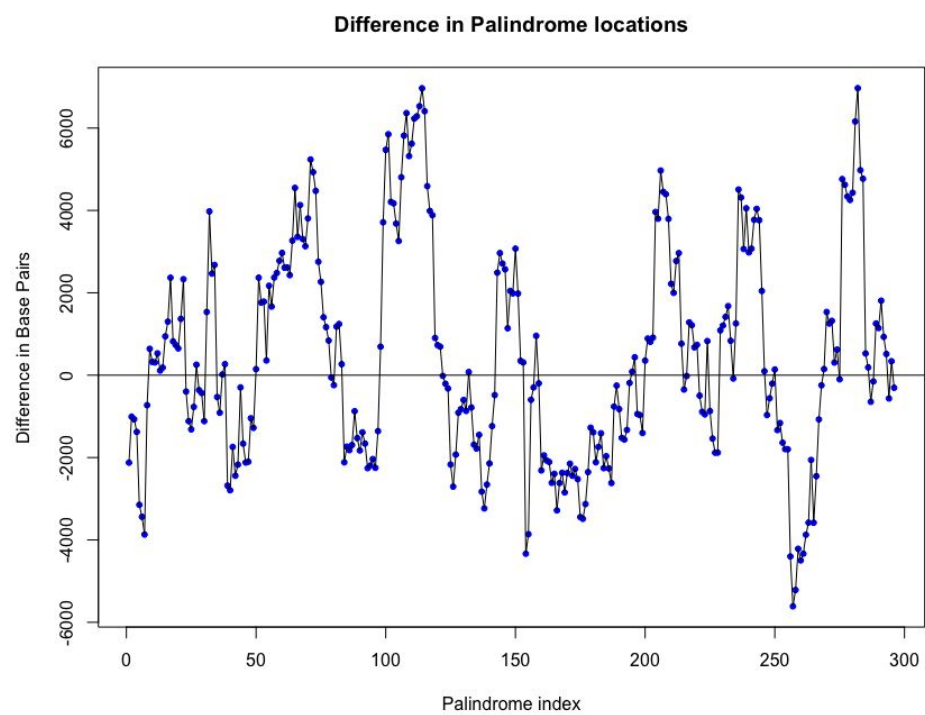


Figure 6: Difference between the given data's location and a random scatter data's location.

Case Study 3

Justin Glommen

Alex Hsieh

Atharva Fulay

Peter Yao

2/23/2017

Setup

Setting up the dataset and the actual data realization information into accessible variables for analysis.

```
# Assuming the data is from the same working directory as this file
gene <- read.table("hcmv.txt", header=TRUE)
data <- gene[,1]
# Actual data
N <- 229354      # Population size
n <- 296         # Sample number of palindromes
editGene <- gene
site.random <- editGene[["location"]]
# Display the data
site.random
```

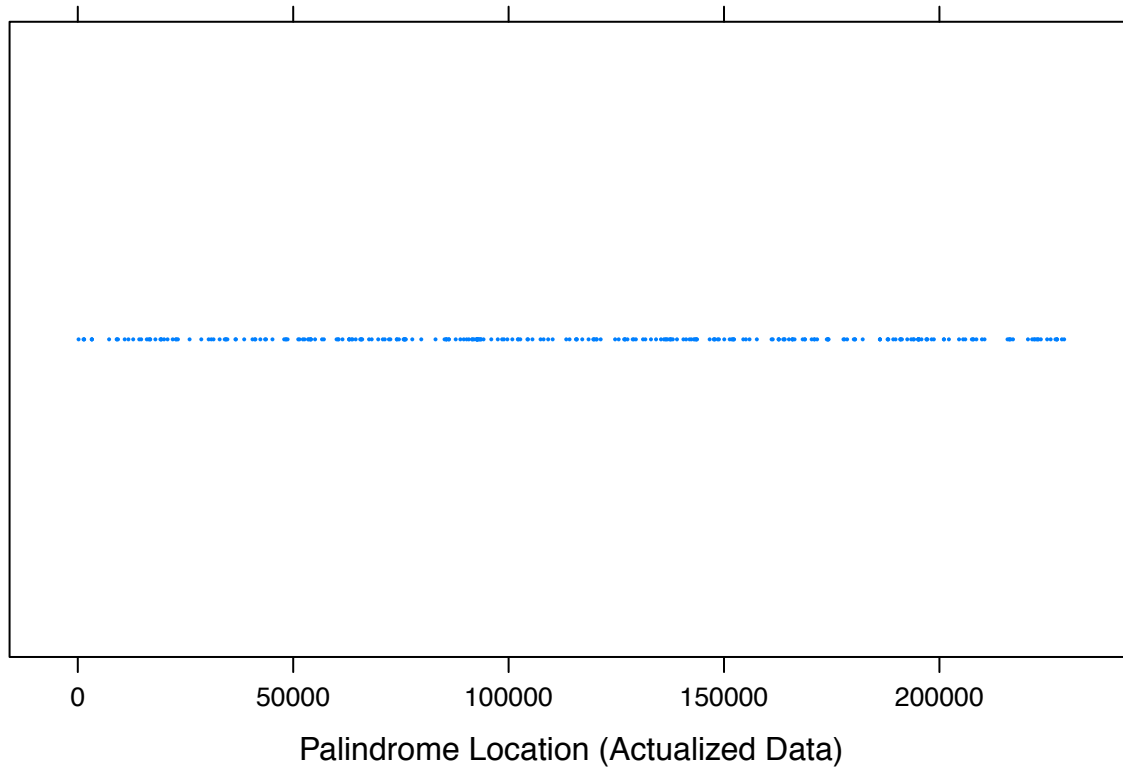
```
## [1] 177 1321 1433 1477 3248 3255 3286 7263 9023 9084
## [11] 9333 10884 11754 12863 14263 14719 16013 16425 16752 16812
## [21] 18009 19176 19325 19415 20030 20832 22027 22739 22910 23241
## [31] 25949 28665 30378 30990 31503 32923 34103 34398 34403 34723
## [41] 36596 36707 38626 40554 41100 41222 42376 43475 43696 45188
## [51] 47905 48279 48370 48699 51170 51461 52243 52629 53439 53678
## [61] 54012 54037 54142 55075 56695 57123 60068 60374 60552 61441
## [71] 62946 63003 63023 63549 63769 64502 65555 65789 65802 66015
## [81] 67605 68221 69733 70800 71257 72220 72553 74053 74059 74541
## [91] 75622 75775 75812 75878 76043 76124 77642 79724 83033 85130
## [101] 85513 85529 85640 86131 86137 87717 88803 89586 90251 90763
## [111] 91490 91637 91953 92526 92570 92643 92701 92709 92747 92783
## [121] 92859 93110 93250 93511 93601 94174 95975 97488 98493 98908
## [131] 99709 100864 102139 102268 102711 104363 104502 105534 107414 108123
## [141] 109185 110224 113378 114141 115627 115794 115818 117097 118555 119665
## [151] 119757 119977 120411 120432 121370 124714 125546 126815 127024 127046
## [161] 127587 128801 129057 129537 131200 131734 133040 134221 135361 136051
## [171] 136405 136578 136870 137380 137593 137695 138111 139080 140579 141201
## [181] 141994 142416 142991 143252 143549 143555 143738 146667 147612 147767
## [191] 147878 148533 148821 150056 151314 151806 152045 152222 152331 154471
## [201] 155073 155918 157617 161041 161316 162682 162703 162715 163745 163995
## [211] 164072 165071 165883 165891 165931 166372 168261 168710 168815 170345
## [221] 170988 170989 171607 173863 174049 174132 174185 174260 177727 177956
## [231] 178574 180125 180374 180435 182195 186172 186203 186210 187981 188025
## [241] 188137 189281 189810 190918 190985 190996 191298 192527 193447 193902
## [251] 194111 195032 195112 195117 195151 195221 195262 195835 196992 197022
## [261] 197191 198195 198709 201023 201056 202198 204548 205503 206000 207527
## [271] 207788 207898 208572 209876 210469 215802 216190 216292 216539 217076
## [281] 220549 221527 221949 222159 222573 222819 223001 223544 224994 225812
```

```
## [291] 226936 227238 227249 227316 228424 228953
```

The Data

Here a strip plot is shown to visualize where palindromes are distributed, by laying out all possible palindrome location sites and displaying binary values representative of palindrome occurrences.

```
library(lattice)
stripplot(site.random, pch=16, cex=0.25, xlab="Palindrome Location (Actualized Data)")
```



Testing

Uniform Random Distribution by Simple Random Sampling

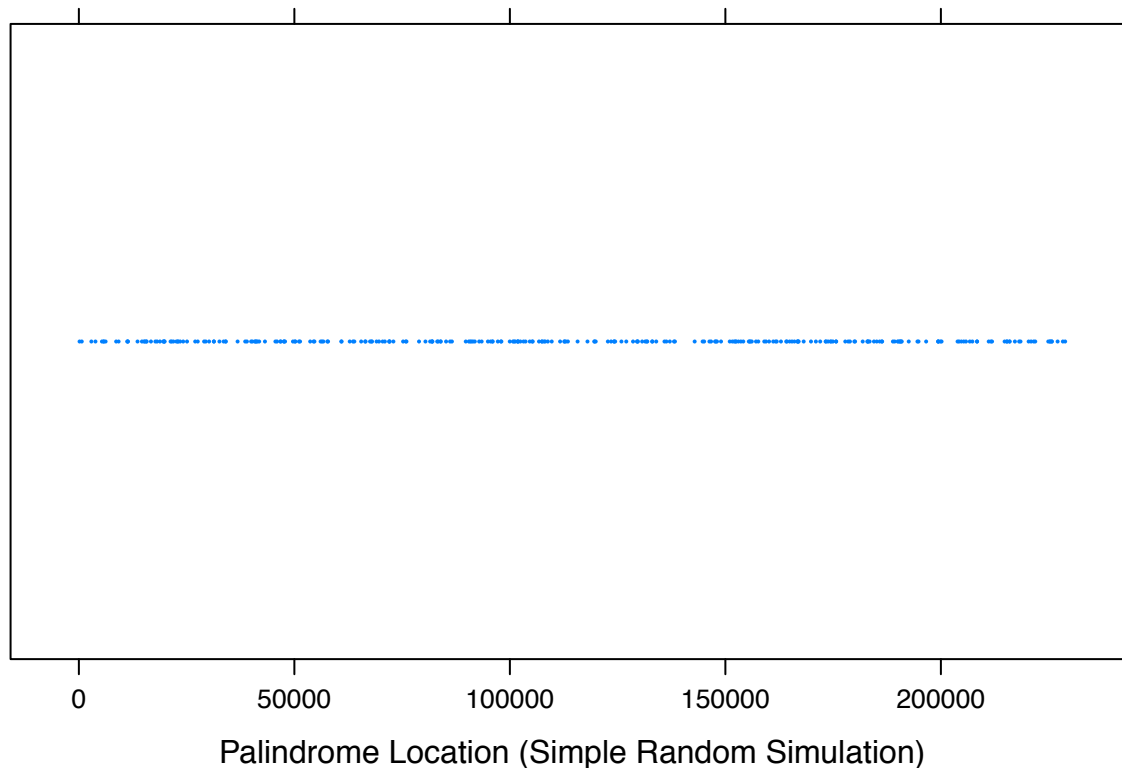
```
# Pseudo data for simulation
set.seed(22217)      # Setting seed for the date at which this analysis was performed
gene <- seq(1, N)
# Produce simple random sample
site.random <- sample.int(N, size=n)
site.random
```

```
## [1] 174949 85198 54658 83367 72973 91865 106758 15113 19546 119591
## [11] 96009 164114 157638 70997 220364 144722 131728 181859 53671 90823
## [21] 130697 124433 72012 190519 183051 95948 123480 224920 204215 225734
## [31] 6201 26994 215341 172002 131893 142875 15731 186370 206710 15320
## [41] 89768 133009 196581 215916 97206 135873 57820 16730 21534 72088
```

```
## [51] 175670 100715 91087 148991 129680 189277 47631 157104 70484 63601
## [61] 105306 41189 63919 124143 67861 225315 204752 2902 208373 225741
## [71] 208412 82017 18803 90473 86555 166688 194563 677 23050 154151
## [81] 199346 151015 112659 67559 221899 188870 91449 136312 118009 112991
## [91] 177821 22628 40832 174562 133071 174368 190177 107483 98033 25099
## [101] 156135 146687 43177 56412 54455 46738 29438 190864 207306 166013
## [111] 24206 23508 83949 161648 199413 15621 60911 225348 101947 153596
## [121] 38596 40275 165459 80415 133956 78921 190555 211874 108806 183468
## [131] 166963 138078 194812 111652 41423 81951 103166 62846 173352 47548
## [141] 185794 217170 56090 34055 38986 92882 147599 51325 32614 19815
## [151] 76009 107916 146298 101794 19782 5773 93454 119658 170997 179921
## [161] 152061 102443 5355 205314 164279 199427 81979 119920 169866 19893
## [171] 50119 127009 30217 128574 125884 184474 11390 113468 131415 145078
## [181] 57659 129479 152350 104617 138299 178905 166946 105186 29061 227086
## [191] 39966 66443 18149 31320 166789 75231 218477 100994 183115 5768
## [201] 214790 148054 17754 27584 205823 36830 215960 152500 153030 159426
## [211] 173815 112666 46851 49676 161107 69637 11329 221523 228302 86128
## [221] 180094 124225 220926 31336 41832 34085 8653 101249 72057 159021
## [231] 33585 160137 190886 147808 45605 162896 50239 47843 211100 60943
## [241] 192546 83934 13614 81505 151623 182934 215324 211606 228835 168165
## [251] 115714 45974 75841 9223 40997 155648 203873 186298 99977 109712
## [261] 94906 97925 103631 161921 65500 199918 129636 3849 101848 11221
## [271] 175712 22927 163 51070 95523 69072 164911 173253 68029 218164
## [281] 189971 107415 21279 185108 178498 155443 137114 14635 56735 200160
## [291] 21943 66517 122735 83140 155824 108215
```

Plotting the uniform random distribution with a strip plot, we can compare it to the actualized data above.

```
library(lattice)
stripplot(site.random, pch=16, cex=0.25, xlab="Palindrome Location (Simple Random Simulation)")
```



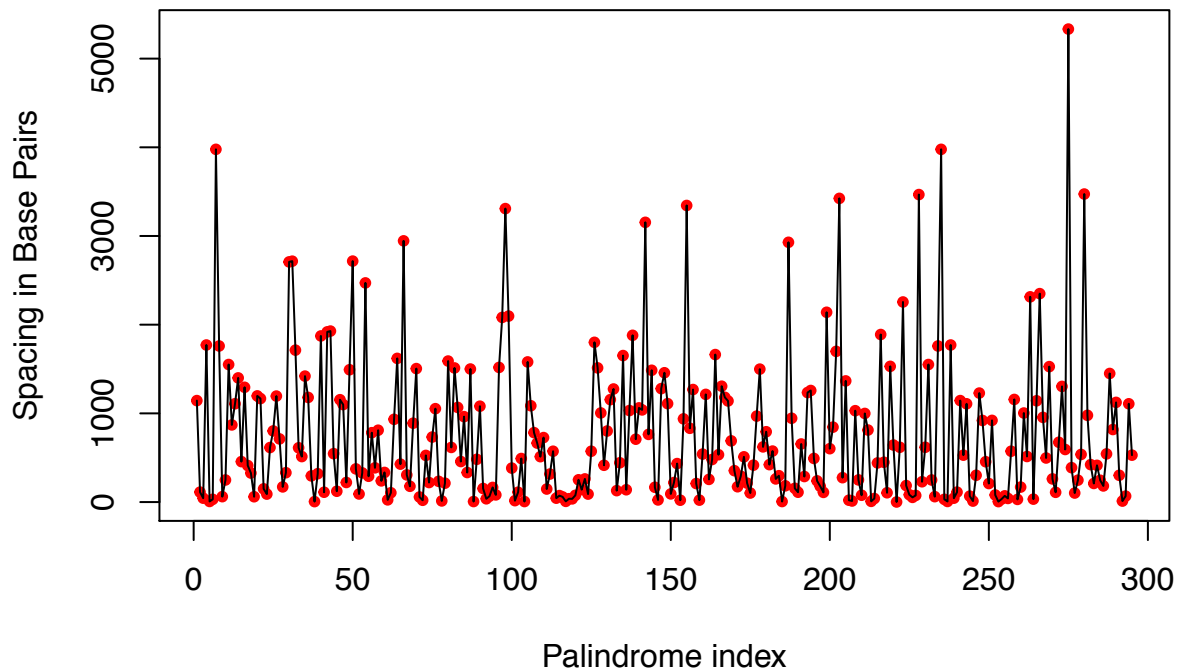
It's easy to notice that while although similar, the actualized data may appear to have a couple more dense clusters than what was generated via the simple sample. We will need to perform more testing to confirm whether these apparent clusters are statistically significant within the actualized data.

Spacing

Here we create a graph to determine the usualness of spacing in the dataset.

```
spacing <- {0}
for (i in 1:length(data)){
  spacing[i] <- data[i+1] - data[i];
}
spacing <- spacing[!is.na(spacing)]
plot(spacing, pch=20,col='red', main="Spacing of Palindrome locations", xlab = "Palindrome index", ylab = "Spacing in Base Pairs", las=1)
lines(spacing, y=NULL)
```

Spacing of Palindrome locations



Monte Carlo Uniform Simulation

Generating the samples

Here we are generating one thousand instances of 296 simple randomly chosen uniformly distributed variables, in order to acquire distributions of desired parameters for testing.

```
B <- 1000 # 1000 bootstrap uniform samples
monteCarloSampleDists <- matrix(data = NA, ncol = n, nrow = B)
monteCarloSampleDists.mean <- vector(mode="logical", length = B)
for (i in 1:B) {
  # Row is overall sample, column is data per sample
```

```

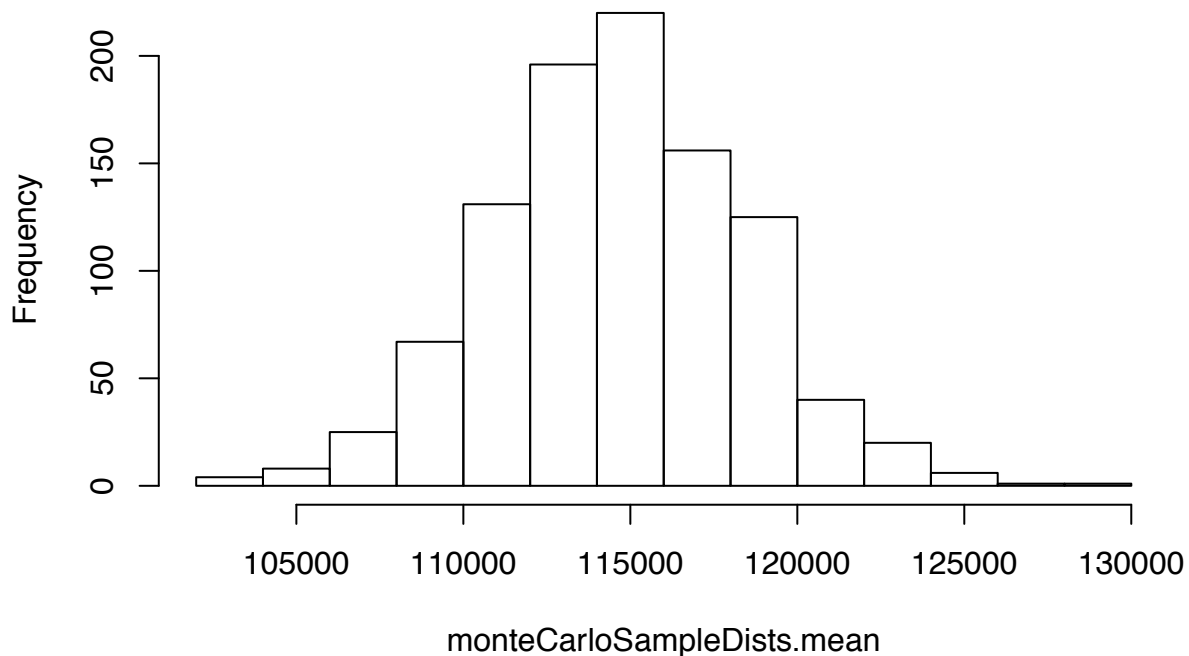
monteCarloSampleDists[i,] <- sample.int(N,n)
# Need to sort them in order to check consecutive palindromes
monteCarloSampleDists[i,] <- monteCarloSampleDists[i,order(monteCarloSampleDists[i,])]
monteCarloSampleDists.mean[i] <- mean(monteCarloSampleDists[i,])
}

#print(monteCarloSampleDists[1,])

hist(monteCarloSampleDists.mean)

```

Histogram of monteCarloSampleDists.mean



Expected Consecutive Palindrome Occurences

Now, we want to build some statistics surrounding the randomly distributed monte carlo simulation. Here we will count the longest string of consecutive palindromes using the sorted data, and more.

```

counts <- matrix(data = NA, nrow = B, ncol = n)
for (i in 1:B) {
  indexHighestCount <- 0
  tempFirstIndex <- 0
  # 0 indicates false, 1 indicates true
  booleanIsConsecutiveNow <- 0
  # Counting the amount of consecutive palindromes
  count <- 1 # Initialized to one since we're counting backwards down below in loop
  highestCount <- 1 # Used to track the highest count overall
  # Inefficient, but stable
  # Starts from 2 to compare to last element
  for (j in 2:n) {
    #monteCarloSampleDists[i,j]
    if (monteCarloSampleDists[ i, (j - 1) ] == ((monteCarloSampleDists[i,j]) - 1)) {

```

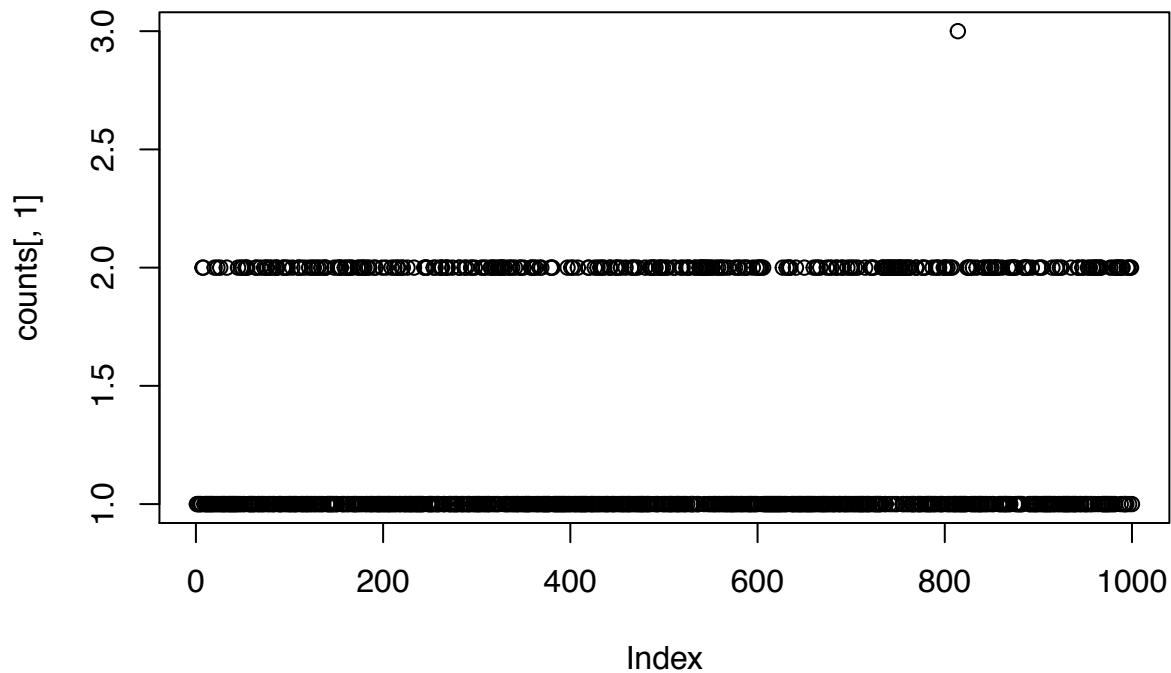


```

    if(booleanIsConsecutiveNow == 0) {
      tempFirstIndex <- (j - 1) # Index at first palindrome in at least 2 consecutive occurrences
    }
    count <- (count + 1)
    booleanIsConsecutiveNow <- 1
  }
  else {
    if (count > highestCount) {
      highestCount <- count
      indexHighestCount <- tempFirstIndex
    }
    count <- 1
    tempFirstIndex <- 0
    booleanIsConsecutiveNow <- 0
  }
}
# Store highest count into the counts array
counts[i,1] <- highestCount
# Store the index at which highestCount occurred into the array also.
counts[i,2] <- indexHighestCount
}

# Clearly, two consecutively is quite normal.
plot(counts[,1])

```



```
summary(counts[,1])
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  1.000   1.000   1.000   1.324   2.000   3.000
```

Now we have to run the same process on the actualized data, to determine if there's an unusually long string of palindromes to help us easily identify the replication site.

```

indexHighestCount <- 0
tempFirstIndex <- 0
# 0 indicates false, 1 indicates true
booleanIsConsecutiveNow <- 0
# Counting the amount of consecutive palindromes
count <- 1 # Initialized to one since we're counting backwards down below in loop
highestCount <- 1 # Used to track the highest count overall
# Inefficient, but stable
# Starts from 2 to compare to last element
for (i in 2:n) {
  if (data[ (i - 1) ] == ((data[i]) - 1)) {
    if(booleanIsConsecutiveNow == 0) {
      tempFirstIndex <- (i - 1) # Index at first palindrome in at least 2 consecutive occurrences
    }
    count <- (count + 1)
    booleanIsConsecutiveNow <- 1
  }
  else {
    if (count > highestCount) {
      highestCount <- count
      indexHighestCount <- tempFirstIndex
    }
    count <- 1
    tempFirstIndex <- 0
    booleanIsConsecutiveNow <- 0
  }
}
# Print out the highest count, and its index.
highestCount

```

```
## [1] 2
```

```
indexHighestCount
```

```
## [1] 221
```

Unfortunately, our realized data matches the above statistics when it comes to consecutive occurrences of palindromes, since the max amount of consecutive occurrences matched that of the third quartile value above. This means there were no unusually long strings of consecutively occurring palindromes, therefore we'll have to resort to the poisson process in order to help us better determine unusual clusters for the replication site.

Poisson Process

First, we'll generate a histogram forming the basis of the poisson distribution of clusters given fixed intervals of one hundred.

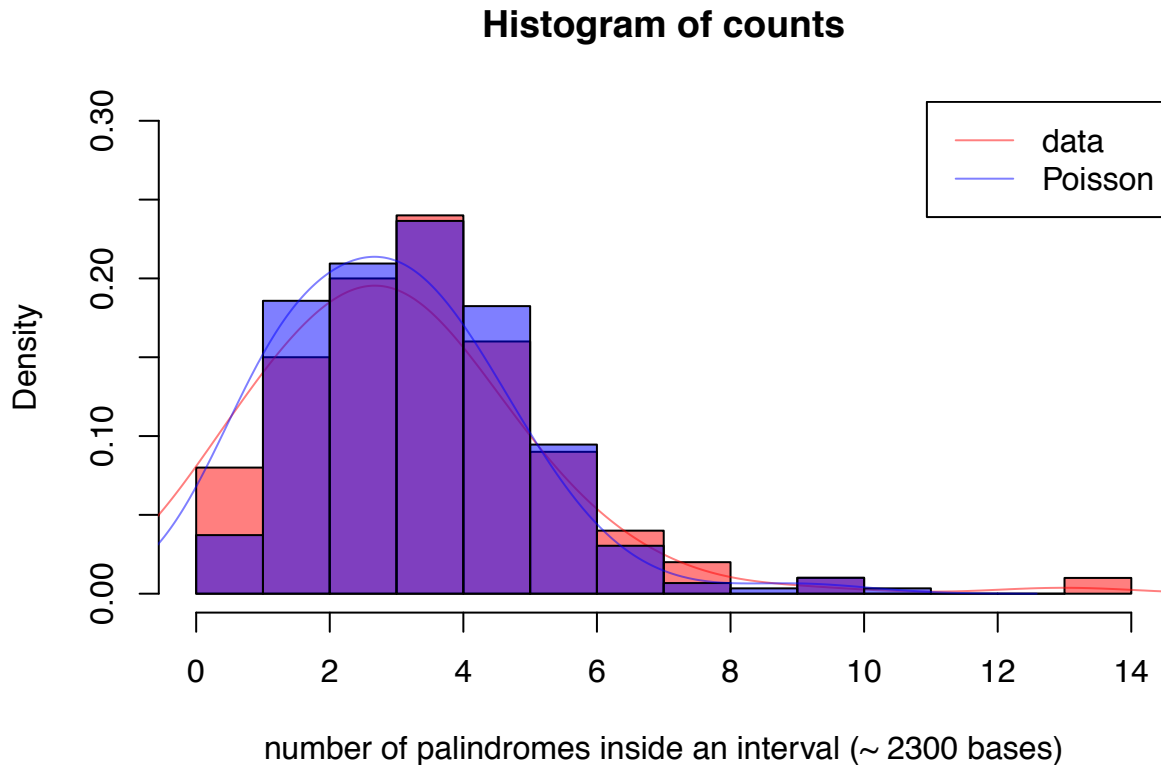
```

k <- 100; # The interval length for clusters of palindromes
n <- 296;
# Split the data into the clustered intervals
tab <- table(cut(data, breaks = seq(0, 230000,
  length.out = k+1), include.lowest = TRUE));

counts <- as.vector(tab);

```

```
hist(counts, breaks=seq(0,14,by=1), col = rgb(1,0,0,0.5),
     probability = TRUE,
     xlab = "number of palindromes inside an interval (~ 2300 bases)",
     ylim = c(0,0.3), include.lowest = TRUE, right = FALSE);
lines(density(counts, adjust = 2), col = rgb(1,0,0,0.5))
Pois <- rpois(296, lambda = mean(counts))
hist(Pois, breaks=seq(0,13,by=1), col = rgb(0,0,1,0.5), probability = TRUE, add = TRUE,
     include.lowest = TRUE, right = FALSE);
lines(density(Pois, adjust = 2), col = rgb(0,0,1,0.5))
legend("topright", legend = c("data", "Poisson"), lty = c(1,1), col = c(rgb(1,0,0,0.5), rgb(0,0,1,0.5)))
```



Finding the Cluster

Then, we'll calculate the highest likely amount of clusters within any one given interval, and compare it to the highest amount of palindromes we have in any one cluster in the actualized data.

```
# Here, the 99.95th percentile is calculated to help determine upper outlier.
highestLikelyCluster <- qpois(.9995, mean(tab)) # Using sample statistic of lambda hat = x bar
print('The highest likely amount in a cluster is:')
```

```
## [1] "The highest likely amount in a cluster is:"
highestLikelyCluster
```

```
## [1] 10
print('and the highest amount of any cluster in our actualized data is:')
```

```
## [1] "and the highest amount of any cluster in our actualized data is:"
```

```
max(tab)
```

```
## [1] 13
```

It's clear that with 99.95% probability, a maximum of ten palindromes will be in any one cluster. Due to our outlier of 13, and given it's the only above 10, it seems like a likely candidate for our replication site.

Chi-Squared Test

First we perform chi-squared test on the simulated data.

```
regionsplit <- function(n.region, gene, site){  
  count.int <- table(cut(site, breaks = seq(1, length(gene), length.out=n.region+1), include.lowest=TRUE))  
  count.vector <- as.vector(count.int)  
  count.tab <- table(count.vector)  
  return (count.tab)  
}
```

```
n.region <- 50  
regionsplit(n.region, gene, site.random)
```

```
## count.vector  
##  2  3  4  5  6  7  8  9 10 11  
##  2  2  5 18  6  6  5  3  2  1
```

```
chisqtable <- function(n.region, site, N){  
  n <- length(site)  
  # lambda estimate  
  lambda.est <- n/n.region  
  # cut into n.region number of non-overlapping intervals  
  count.int <- table(cut(site, breaks = seq(1, length(gene), length.out=n.region+1), include.lowest=TRUE))  
  # get the count levels range  
  count.vector <- as.vector(count.int)  
  count.range <- max(count.vector) - min(count.vector) + 1  
  
  # create contingency table  
  table <- matrix(rep(NA, count.range*3), count.range, 3)  
  for (i in 1:count.range){  
    offset <- min(count.vector) - 1  
    # first column = count level  
    table[i, 1] <- i + offset  
    # second column = observed count  
    table[i, 2] <- sum(count.vector == i + offset)  
    # third column = expected count  
    if ((i + offset == min(count.vector)) && (min(count.vector) != 0))  
      table[i, 3] <- ppois(i+offset, lambda.est)*n.region  
    else if (i + offset == max(count.vector))  
      table[i, 3] <- 1 - ppois(i + offset - 1, lambda.est)  
    else  
      table[i, 3] <- (ppois(i+offset, lambda.est) - ppois(i + offset - 1, lambda.est))*n.region  
  }  
  return (table)  
}  
site.random.tabtemp <- chisqtable(n.region, site.random, N)
```

```

site.random.tab <- matrix(rep(NA, 7*2), 7, 2)
site.random.tab[1,] <- colSums(site.random.tabtemp[1:2, 2:3])
site.random.tab[2:6,] <- site.random.tabtemp[3:7, 2:3]
site.random.tab[7,] <- colSums(site.random.tabtemp[7:9, 2:3])
site.random.stats <- sum((site.random.tab[,2] - site.random.tab[,1])^2/site.random.tab[,2])
pchisq(site.random.stats, 7 - 2, lower.tail=FALSE) #if lower.tail=TRUE then you're testing something el.

```

```
## [1] 0.01025207
```

Here we get a result of about .01.

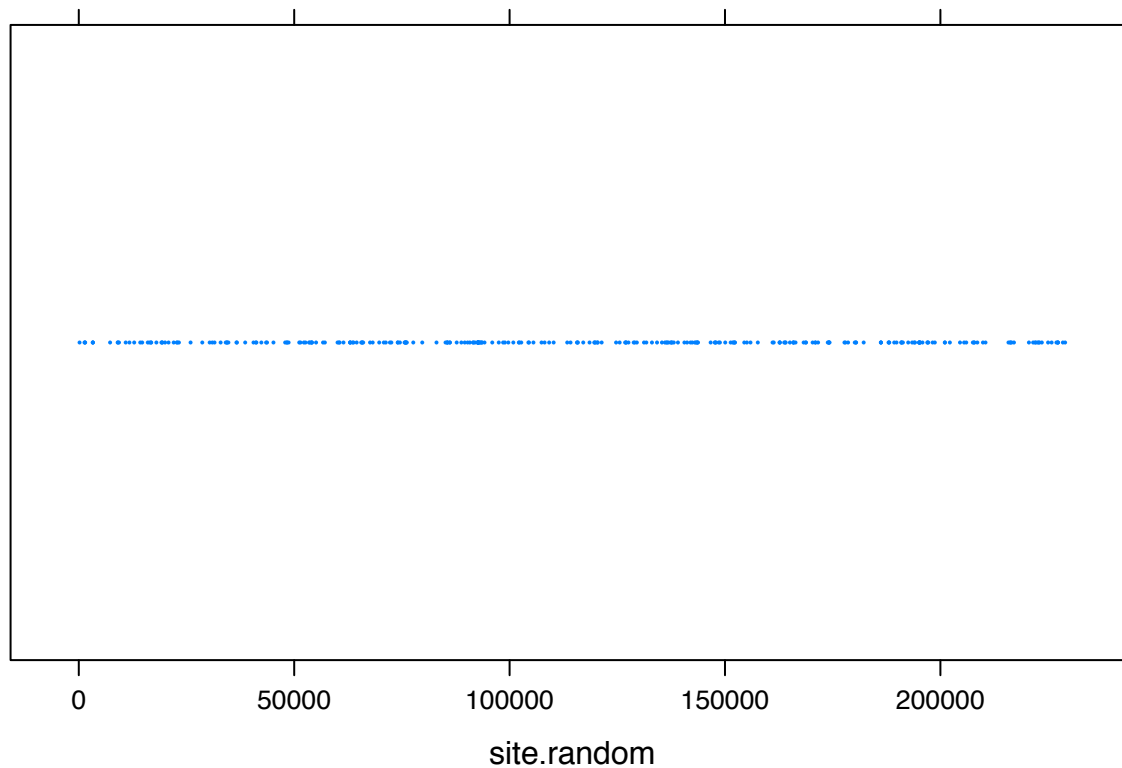
We then again perform the chi-squared test on the real data.

```

#actual data
N <- 229354
n <- 296
site.random <- editGene[["location"]]

library(lattice)
stripplot(site.random, pch=16, cex=0.25)

```



```

n.region <- 50

regionsplit <- function(n.region, gene, site){
  count.int <- table(cut(site, breaks = unique(seq(1, N, length.out=n.region+1)), include.lowest=TRUE))
  count.vector <- as.vector(count.int)
  count.tab <- table(count.vector)
  return (count.tab)
}

```

```

regionsplit(n.region, gene, site.random)

## count.vector
## 0 1 2 3 4 5 6 7 8 9 10 13 15
## 1 2 1 4 8 8 5 9 4 5 1 1 1

chisqtable <- function(n.region, site, N){
  n <- length(site)
  # lambda estimate
  lambda.est <- n/n.region
  # cut into n.region number of non-overlapping intervals
  count.int <- table(cut(site, breaks = unique(seq(1, N, length.out=n.region+1)), include.lowest=TRUE))
  # get the count levels range
  count.vector <- as.vector(count.int)
  count.range <- max(count.vector) - min(count.vector) + 1

  # create contingency table
  table <- matrix(rep(NA, count.range*3), count.range, 3)
  for (i in 1:count.range){
    offset <- min(count.vector) - 1
    # first column = count level
    table[i, 1] <- i + offset
    # second column = observed count
    table[i, 2] <- sum(count.vector == i + offset)
    # third column = expected count
    if ((i + offset == min(count.vector)) && (min(count.vector) != 0))
      table[i, 3] <- ppois(i+offset, lambda.est)*n.region
    else if (i + offset == max(count.vector))
      table[i, 3] <- 1 - ppois(i + offset - 1, lambda.est)
    else
      table[i, 3] <- (ppois(i+offset, lambda.est) - ppois(i + offset - 1, lambda.est))*n.region
  }
  return (table)
}
site.random.tabtemp <- chisqtable(n.region, site.random, N)

site.random.tab <- matrix(rep(NA, 7*2), 7, 2)
site.random.tab[1,] <- colSums(site.random.tabtemp[1:2, 2:3])
site.random.tab[2:6,] <- site.random.tabtemp[3:7, 2:3]
site.random.tab[7,] <- colSums(site.random.tabtemp[7:9, 2:3])
site.random.stats <- sum((site.random.tab[,2] - site.random.tab[,1])^2/site.random.tab[,2])
pchisq(site.random.stats, 7 - 2, lower.tail=FALSE) #if lower.tail=TRUE then you're testing something el.

## [1] 0.2219605

```

We here get a result of about .22.