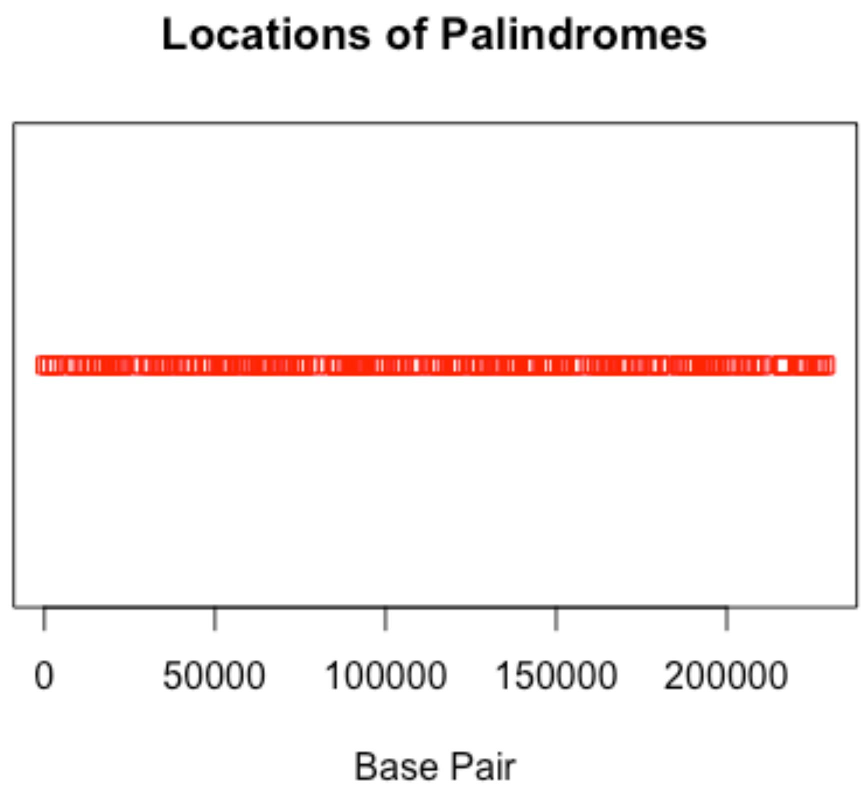
case study 3

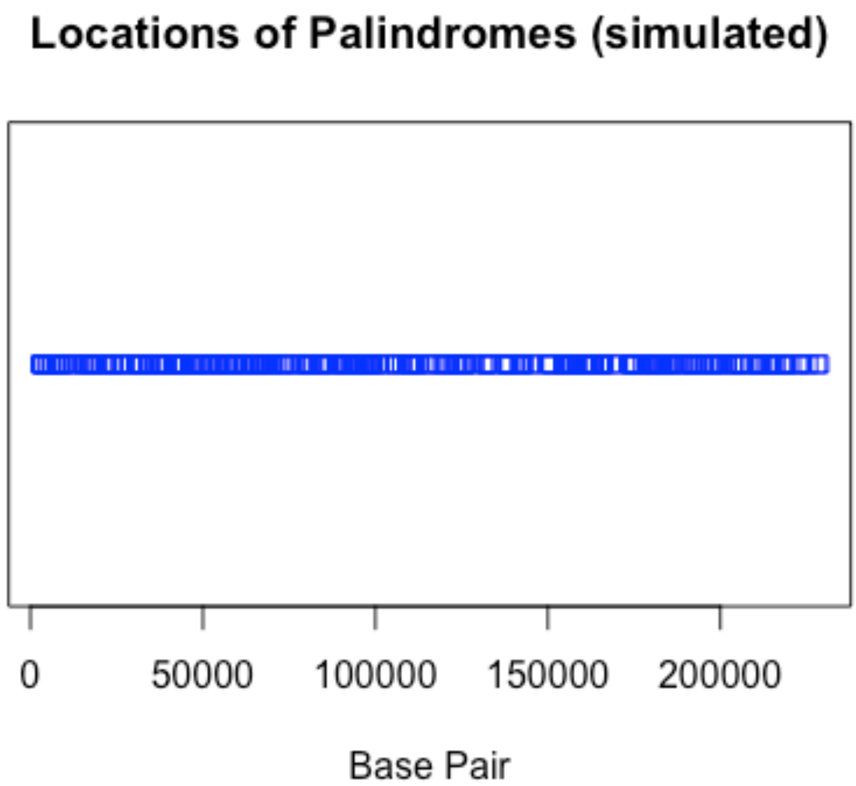
```r
hcmv <- read.table("~/Desktop/MATH 189/Homework 3/hcmv-263hxkx-1qhtfgz.txt",
header = TRUE, sep="")
library(ggplot2)
```

Scenario 1

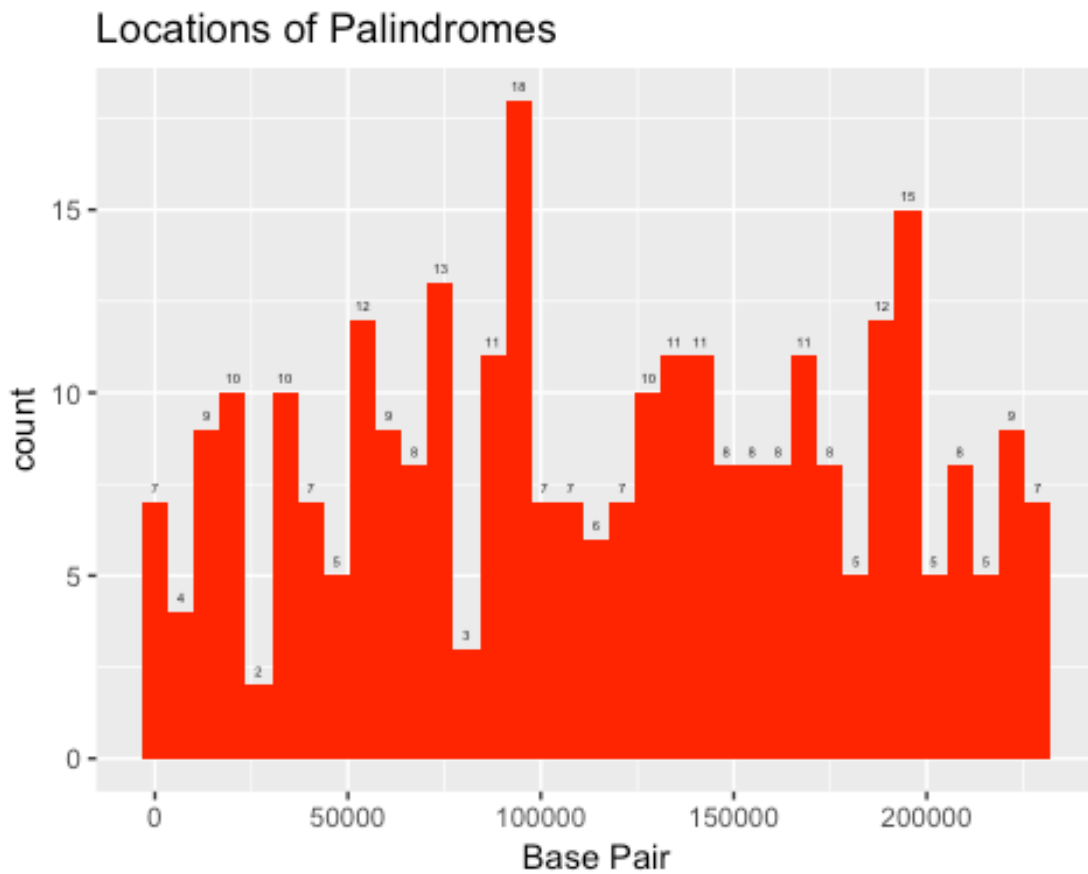
```r
sample=sort(runif(296, min=0, max=229354))
stripchart(hcmv, col='red', main='Locations of Palindromes', xlab='Base
Pair')
```



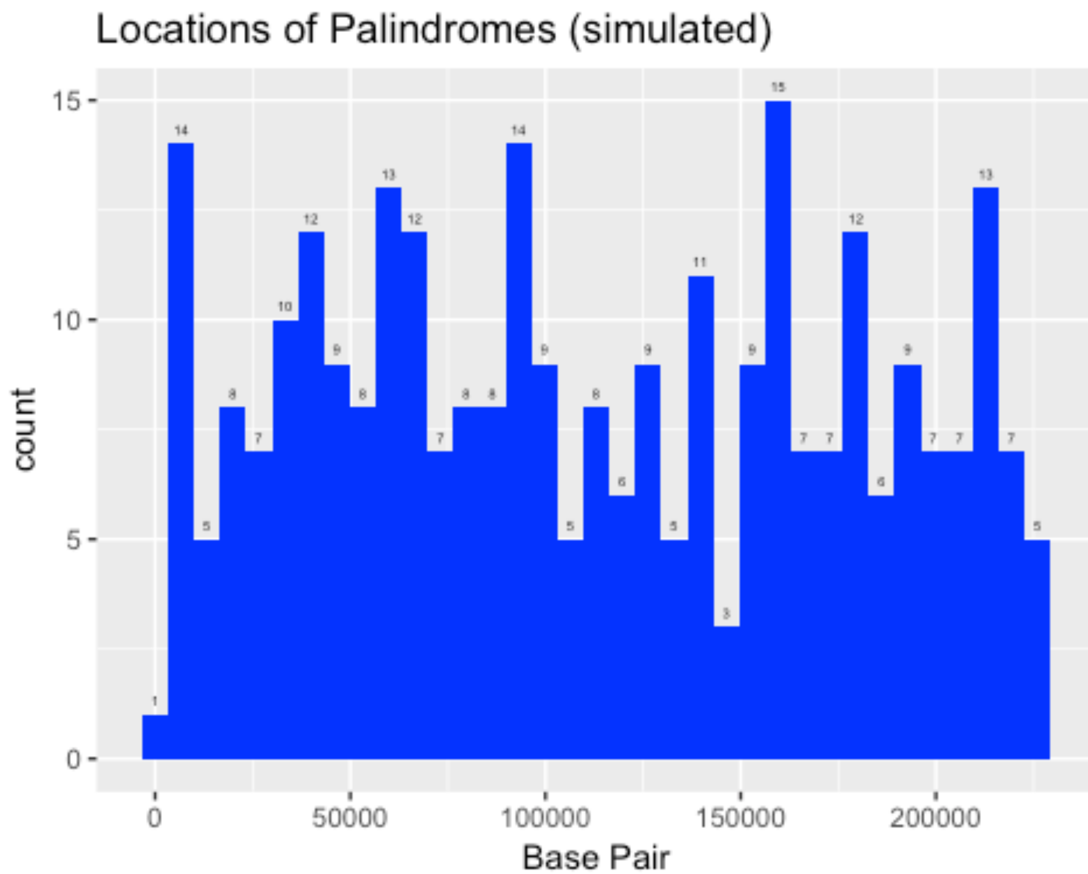

```r
stripchart(sample, col='blue', main='Locations of Palindromes (simulated)',
xlab='Base Pair')
```



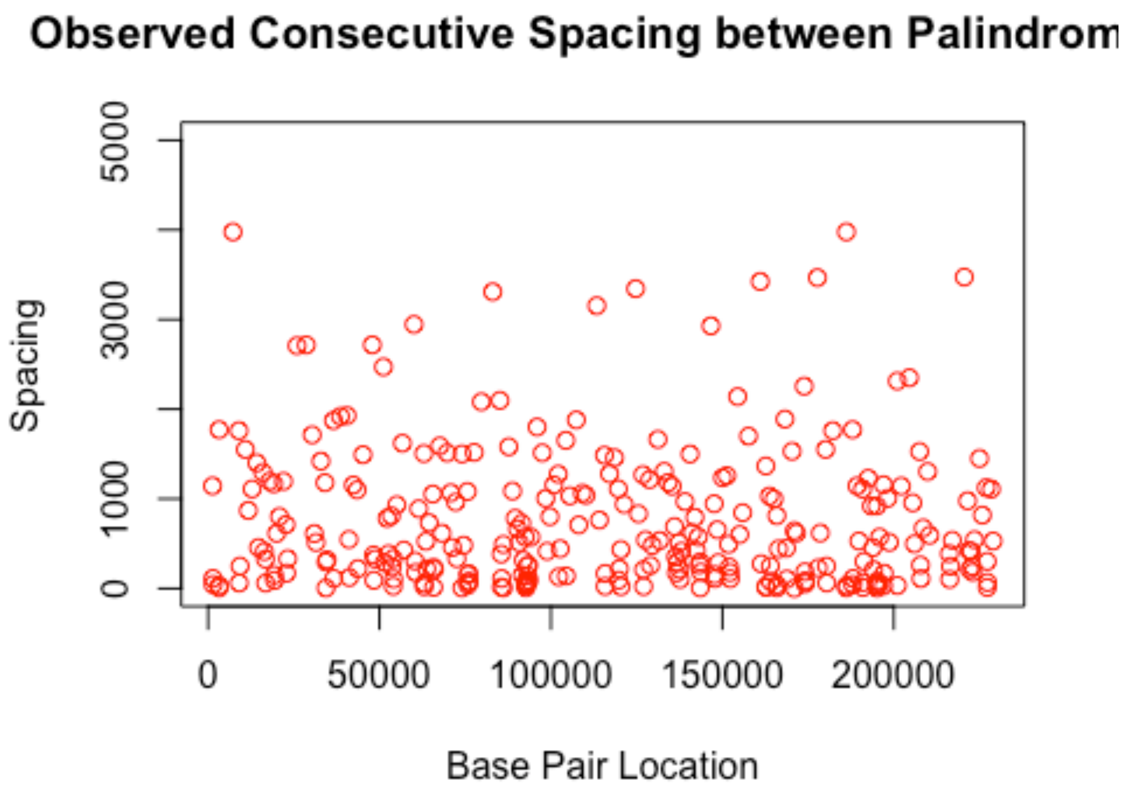

```
ggplot(hcmv, aes(x=location))+
  geom_histogram(bins=35, fill='red')+
  labs(title='Locations of Palindromes', x='Base Pair')+
  stat_bin(bins=35, geom='text', aes(label=..count..), size=1.5, vjust=-1)
```



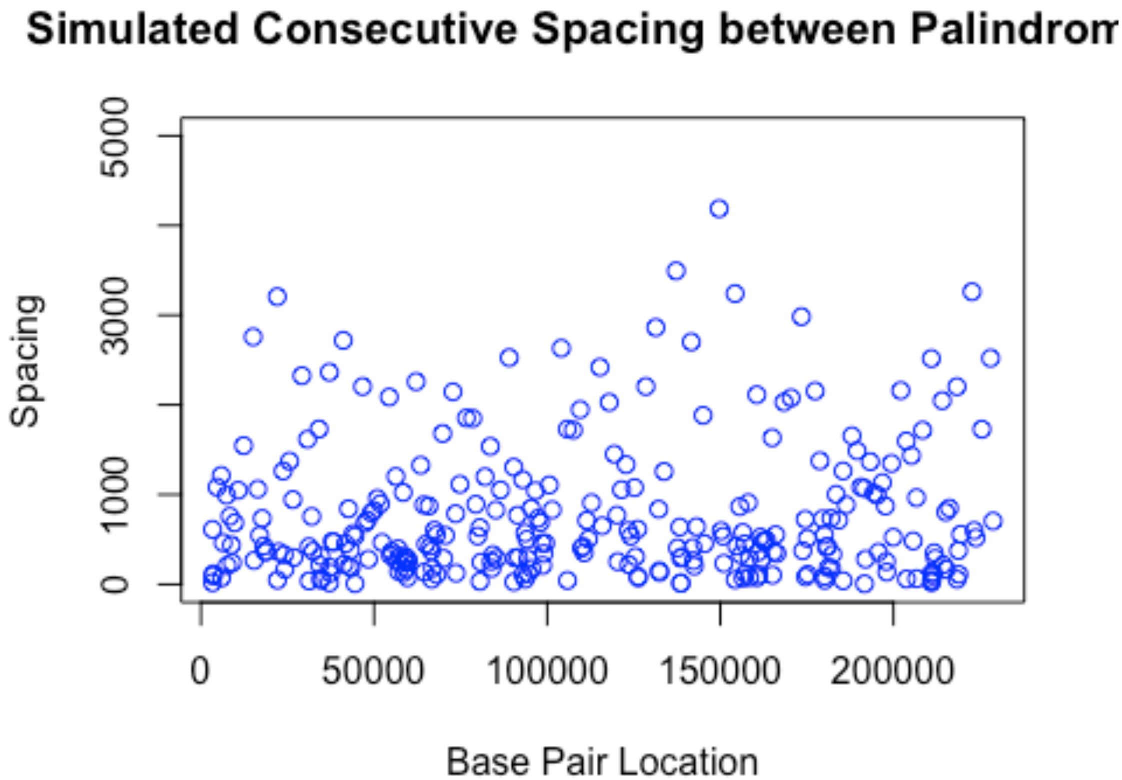

```r
df<-data.frame('location'=sample)
ggplot(df, aes(x=location))+
  geom_histogram(bins=35,fill='blue')+
  labs(title='Locations of Palindromes (simulated)', x='Base Pair')+
  stat_bin(bins=35, geom='text', aes(label=..count..), size=1.5, vjust=-1)
```



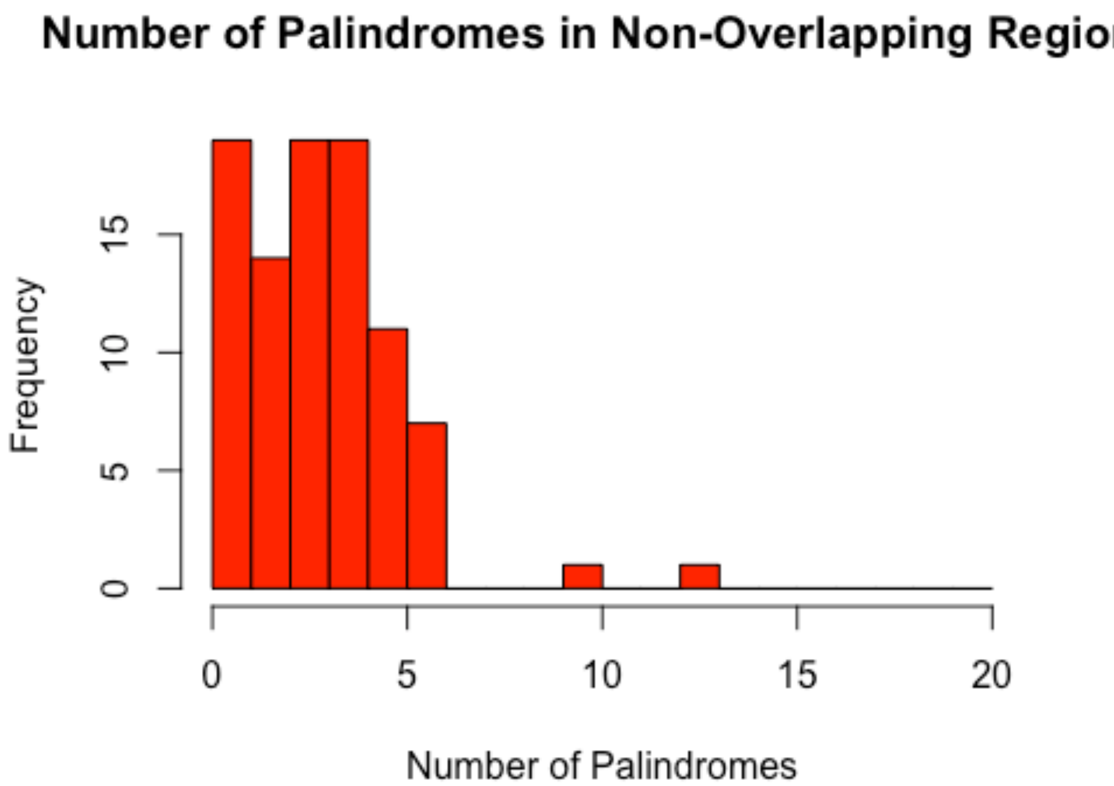

```
plot(hcmv$location[-1], diff(hcmv$location), col='red', ylim=c(0,5000),
main='Observed Consecutive Spacing between Palindromes', xlab='Base Pair
Location', ylab='Spacing')
```



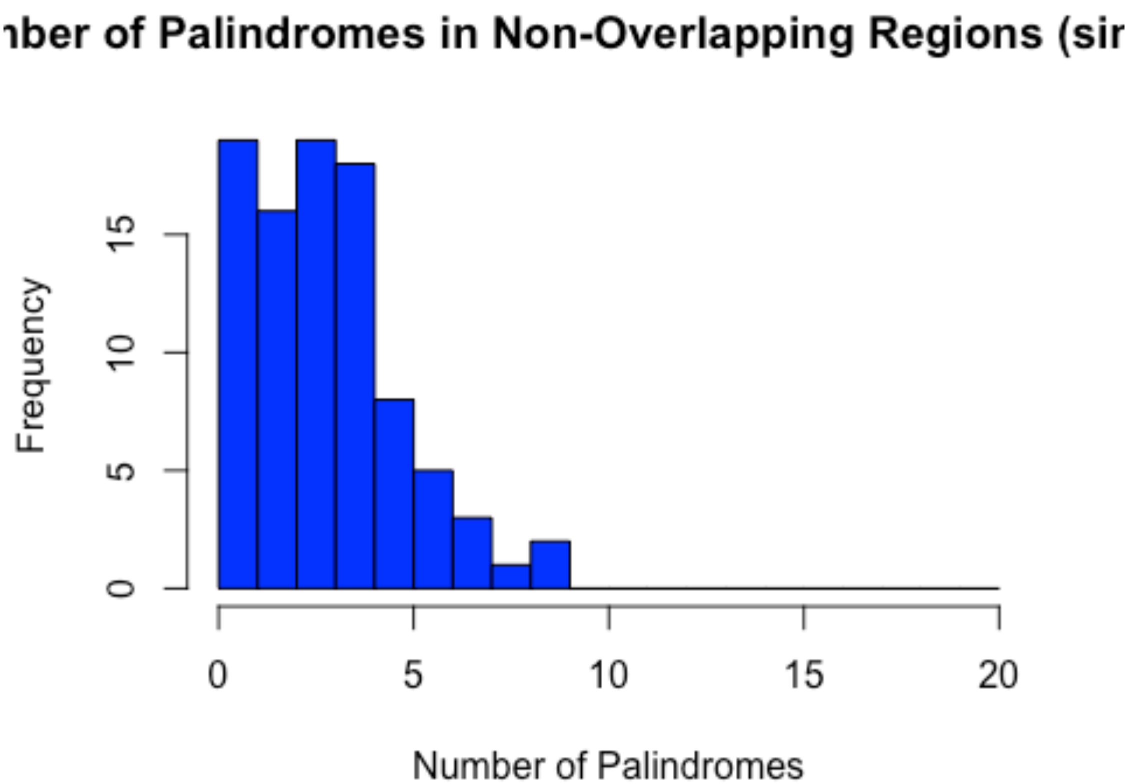

```
plot(df$location[-1], diff(df$location), col='blue', ylim=c(0,5000),
main='Simulated Consecutive Spacing between Palindromes', xlab='Base Pair
Location', ylab='Spacing')
```




```
hist(as.vector(table(cut(hcmv$location, breaks=seq(0,229354,2500),
include.lowest=TRUE))), breaks=seq(0,20,1), col='red', main='Number of
Palindromes in Non-Overlapping Regions', xlab='Number of Palindromes')
```




```
hist(as.vector(table(cut(df$location, breaks=seq(0,229354,2500),
include.lowest=TRUE))), breaks=seq(0,20,1), col='blue', main='Number of
Palindromes in Non-Overlapping Regions (simulated)', xlab='Number of
Palindromes')
```



ıber of Palindromes in Non-Overlapping Regions (sir
Frequency
0
5
10
15
0
5
10
15
20
Number of Palindromes

```
> setwd("/Users/nuochen/Desktop/Math189")
> data <- read.table("case3.txt",header = T)
>
> #count spacing
> i = 2
> number <- c()
>     for(i in 2:296){
+         a <- data$location[i]-data$location[i-1]
+         number <- c(number,a)
+     }
> number
  [1] 1144  112   44 1771    7   31 3977 1760   61  249 1551  870 1109 1400  456 1294
412
 [18]  327   60 1197 1167  149   90  615  802 1195  712  171  331 2708 2716 1713
612  513
 [35] 1420 1180  295    5  320 1873  111 1919 1928  546  122 1154 1099  221 1492 2717
374
 [52]   91  329 2471  291  782  386  810  239  334   25  105  933 1620  428 2945
306  178
 [69]  889 1505   57   20  526  220  733 1053  234   13  213 1590  616 1512 1067
457  963
 [86]  333 1500    6  482 1081  153   37   66  165   81 1518 2082 3309 2097  383
16  111
[103]  491    6 1580 1086  783  665  512  727  147  316  573   44   73   58    8
38   36
[120]   76  251  140  261   90  573 1801 1513 1005  415  801 1155 1275  129  443
1652  139
[137] 1032 1880  709 1062 1039 3154  763 1486  167   24 1279 1458 1110   92  220  434
21
[154]  938 3344  832 1269  209   22  541 1214  256  480 1663  534 1306 1181 1140
690  354
[171]  173  292  510  213  102  416  969 1499  622  793  422  575  261  297    6
183 2929
[188]  945  155  111  655  288 1235 1258  492  239  177  109 2140  602  845 1699
3424  275
[205] 1366   21   12 1030  250   77  999  812    8   40  441 1889  449  105 1530
643    1
[222]  618 2256  186   83   53   75 3467  229  618 1551  249   61 1760 3977   31
7 1771
[239]   44  112 1144  529 1108   67   11  302 1229  920  455  209  921   80    5
34   70
```

```
[256]     41    573  1157     30    169  1004    514  2314     33  1142  2350    955    497  1527    261
110    674
[273]  1304    593  5333    388    102    247    537  3473    978    422    210    414    246    182    543
1450    818
[290] 1124    302      11      67 1108    529
> min(number)
[1] 1
> max(number)
[1] 5333
> mean(number)
[1] 775.5119
>
> #divide to 20 intervals
> k <- 20
> tab <- table(cut(number,breaks = seq(0,5333,length.out = k+1),include.lowest = T))
> tab

            [0,267]           (267,533]           (533,800]     (800,1.07e+03]
                106                  47                  30                 26
(1.07e+03,1.33e+03]  (1.33e+03,1.6e+03]  (1.6e+03,1.87e+03] (1.87e+03,2.13e+03]
                 31                  19                  10                  7
 (2.13e+03,2.4e+03]  (2.4e+03,2.67e+03] (2.67e+03,2.93e+03]  (2.93e+03,3.2e+03]
                  4                   1                   4                  2
 (3.2e+03,3.47e+03] (3.47e+03,3.73e+03]    (3.73e+03,4e+03]    (4e+03,4.27e+03]
                  3                   2                   2                  0
(4.27e+03,4.53e+03]  (4.53e+03,4.8e+03]  (4.8e+03,5.07e+03] (5.07e+03,5.33e+03]
                  0                   0                   0                  1
> count <- as.vector(tab)
> count
 [1]  106    47    30    26    31    19    10     7     4     1     4     2     3     2     2     0     0     0     0
1
>
> #generate a ramdom exponential distribution
> set.seed(300)
> simdata = rexp(n=296, rate= 0.00129 )
>
> #draw histogram of spacing
> hist(simdata, breaks = 25, col = rgb(1,0,0,0.5),probability = TRUE, xlab= "spacing of consecutive
hits", main= "Histogram of spacing")
> hist(number, breaks = 25, col = rgb(0,0,1,0.5), probability = TRUE, add = TRUE)
> lines(density(simdata, adjust = 2),lwd=2, col = rgb(1,0,0,0.5))
```

```
> lines(density(number, adjust = 2), lwd=2, col = rgb(0,0,1,0.5))
> legend("topright", legend = c("simdata", "number"), lty = c(1,1), col = c(rgb(1,0,0,0.5),rgb(0,0,1,0.5)))
>
> #draw scatterplot
> plot(number, main ="Scatterplot of spacing of observed data", xlab="295 consecutive palindromes", ylab="Spacing", col=rgb(0,0,1,0.5))
> plot(simdata, main ="Scatterplot of spacing of simulated data", xlab="295 consecutive palindromes", ylab="Spacing", col=rgb(1,0,0,0.5))
>
> #count spacing between consecutive pairs
> i = 2
> pairsnumber <- c()
> for(i in 2:296){
+     b <- data$location[i]-data$location[i-2]
+     pairsnumber <- c(pairsnumber,b)
+ }
>
> #count spacing between triplets
> i = 3
> tripletsnumber <- c()
> for(i in 3:296){
+     d <- data$location[i]-data$location[i-3]
+     tripletsnumber <- c(tripletsnumber,d)
+ }
> max(pairsnumber)
[1] 5926
>
> #generate a random gamma distribution for pairs spacing
> set.seed(300)
> simpairsdata <- rgamma(296,shape=2,scale=1/0.00129)
>
> #generate a random gamma distribution for triplets spacing
> set.seed(300)
> simtripletsdata <- rgamma(296,shape=3,scale=1/0.00129)
>
> #draw histograms of pairs spacing
> hist(pairsnumber, probability = TRUE,breaks = 30, col = rgb(0,0,1,0.5))
> hist(simpairsdata, probability = TRUE,breaks =30, col = rgb(1,0,0,0.5), add=TRUE)
> lines(density(simpairsdata, adjust = 2),lwd=2, col = rgb(1,0,0,0.5))
> lines(density(pairsnumber, adjust = 2), lwd=2, col = rgb(0,0,1,0.5))
```

```
> legend("topright", legend = c("simpairsdata", "pairsnumber"), lty = c(1,1), col = c(rgb(1,0,0,0.5),rgb(0,0,1,0.5)))
>
> #draw histograms of triplets spacing
> hist(tripletsnumber, probability = TRUE,breaks = 30, col = rgb(0,0,1,0.5))
> hist(simtripletsdata, probability = TRUE,breaks =30, col = rgb(1,0,0,0.5), add=TRUE)
> lines(density(simtripletsdata, adjust = 2),lwd=2, col = rgb(1,0,0,0.5))
> lines(density(tripletsnumber, adjust = 2), lwd=2, col = rgb(0,0,1,0.5))
> legend("topright", legend = c("simtripletsdata", "tripletsnumber"), lty = c(1,1), col = c(rgb(1,0,0,0.5),rgb(0,0,1,0.5)))
>
> #regroup the interval (merge <5)
> mcount <-c(106,47,30,26,31,19,10,7,5,6,8)
> mcount
 [1] 106  47  30  26  31  19  10   7   5   6   8
>
> #max likehood function
> 1/mean(number)
[1] 0.001289471
>
> set.seed(1000)
> #generate 296 ramdom exponential distribution 500 times
> a1<-c(50)
> a2<-c(50)
> a3<-c(50)
> a4<-c(50)
> a5<-c(50)
> a6<-c(50)
> a7<-c(50)
> a8<-c(50)
> a9<-c(50)
> a10<-c(50)
> a11<-c(50)
> for(i in 1:500){
+     temp<-rexp(295,0.00129)
+     a1[i]<-sum(temp<=267)
+     a2[i]<-sum(temp>267 & temp<=533)
+     a3[i]<-sum(temp>533 & temp<=800)
+     a4[i]<-sum(temp>800 & temp<=1070)
+     a5[i]<-sum(temp>1070 & temp<=1330)
+     a6[i]<-sum(temp>1330 & temp<=1600)
```

```
+     a7[i]<-sum(temp>1600 & temp<=1870)
+     a8[i]<-sum(temp>1870 & temp<=2130)
+     a9[i]<-sum(temp>2130 & temp<=2670)
+     a10[i]<-sum(temp>2670 & temp<=3200)
+     a11[i]<-sum(temp>3200 & temp<=5333)
+ }
> temp
   [1]    730.711968    694.082152    974.336653   1329.588723    314.335433      66.468647
212.030640
   [8]     59.287701    400.332735    242.908501    483.712468   1933.751225    609.758534
305.137951
  [15]    609.470304    537.460675    517.890987    148.048381    965.528494    886.896955
95.267191
  [22]    184.406250    370.921824   2273.022523    687.350188   1392.202368       2.771273
752.943116
  [29]    262.715131    286.105489    164.125880    615.460052   1815.519564    282.206906
253.975143
  [36]    685.467966   4066.803463    843.201430     54.641369    445.791152    558.986193
665.182608
  [43]   2090.321948    493.389336   1670.510037     63.829631    330.338474     69.780223
54.851270
  [50]    473.016164   1320.478386    307.042016   2107.146392    552.997609    321.297818
225.363717
  [57]    939.898015   1793.759736    691.711212    920.576734    156.072088    397.387786
1497.814413
  [64]   1095.942336    461.997456    271.656543   1243.409341    181.623620    662.611656
44.469200
  [71]      1.726196    212.767248    388.157671   2533.836478    316.034374     76.754764
1047.196224
  [78]     64.159077    675.144577    402.254838    403.899279    239.282014    563.745840
228.572222
  [85]    257.835936     80.969634   2114.201278   1850.520474    531.705809    696.503948
188.660137
  [92]   2793.736939    563.122005   1987.108381    348.984031   1441.904679    457.412900
1053.079330
  [99]    656.577862    328.638089    338.246617    904.328499   2302.189697    266.199658
254.366442
 [106]    229.343738    471.169257   1818.936699   1551.416002    392.838088     98.053088
811.925891
 [113]   1856.235861     10.629545    417.391891    721.732529   1782.929586   1252.843614
147.553649
```

```
[120]     224.184667      49.417040   1287.958809     933.470280   1165.400109     152.511845
18.595365
[127]     209.495420     136.039987     758.960624     166.127202     439.811834   1230.925189
2282.460628
[134]     625.636965     237.142076     638.835689     446.261167     355.627768     435.617073
339.303026
[141]     245.570548   1464.177893     506.759609     809.950131     162.038794     860.945871
56.922072
[148]   2021.898819     595.583480   2114.177704   2289.677463     108.022566      49.173159
1298.504246
[155]     175.358416     810.238509     442.594447     461.048960      93.357439   2766.115853
262.003190
[162]     852.799828   1442.325885   1063.349461     911.795923      42.418384     172.802483
153.929090
[169]     148.957883   1368.139690     569.117080      69.859574   1502.809390     369.979447
1706.129808
[176]      75.853233   1935.163846     584.153175     812.512809     161.848756     249.484066
224.396821
[183]     627.880833      87.008010   2610.041761      87.146769     828.335963     416.034528
189.606289
[190]     340.609770     610.970059     135.691447     234.355909     475.121885     860.562828
120.395931
[197]   3504.029139     586.948397     252.361139     300.128225     512.495556     406.085324
303.224466
[204]     486.918847   1011.915256      13.150791     277.618001   1151.963020     137.288558
169.831301
[211]     103.249320     176.666095      51.956143     511.384739   1662.562870     387.201722
1591.787081
[218]     290.167602     102.642014       5.794059   2089.053527     966.402432     666.220067
181.679600
[225]   1442.090360     373.912389     218.833456     237.019371     313.094065     295.849588
55.258398
[232]     197.711741     910.199763   1316.525956     654.406936      31.245568     536.817169
190.728459
[239]     471.561024     252.119139     417.684797     114.855076     554.663905       6.599344
336.088675
[246]     380.650435     558.283463     703.552419     549.407810     685.611836      87.860525
63.426279
[253]     470.721109     188.290256     756.482075     312.592208   1227.865964   1215.285756
653.140914
[260]     420.689954   2454.507992   1272.887912     587.469429     269.437477      41.491714
```

```
135.845240
[267]    125.694894    292.766467    251.742252    421.120568   1587.731534   1142.680812
473.253370
[274]   1217.178478   1115.079217   1053.491876    667.205424    423.651828    568.774966
103.774301
[281]   1882.357573    301.894143   1026.077909     36.991832   3264.767847   2851.550643
561.020982
[288]   1475.850742    324.180696    258.727155   2529.093916    641.150211    279.487319
5.632043
[295]    559.284949
> result=c(11)
> result[1]=mean(a1)
> result[2]=mean(a2)
> result[3]=mean(a3)
> result[4]=mean(a4)
> result[5]=mean(a5)
> result[6]=mean(a6)
> result[7]=mean(a7)
> result[8]=mean(a8)
> result[9]=mean(a9)
> result[10]=mean(a10)
> result[11]=mean(a11)
> result
 [1] 86.006 60.828 43.488 30.794 20.940 15.740 10.950  7.350  9.560  4.652  4.390
>
> #conduct the chi-square test
> var <- c(11)
> chi <- c(11)
> for (i in 1:11){
+   var[i] <- mcount[i]-result[i]
+   chi[i] <- (var[i]^2)/result[i]
+ }
> var
 [1]  19.994  -13.828  -13.488   -4.794   10.060    3.260   -0.950   -0.350   -4.560    1.348
3.610
> chi
 [1]  4.64804823  3.14351259  4.18336424  0.74632838  4.83302770  0.67519695  0.08242009
0.01666667
 [9] 2.17506276 0.39060705 2.96858770
> sum(chi)
[1] 23.86282
```

```
>
> cr <- qchisq(0.05,9,lower = F) #chi-square with significant level
> pvar <- c(19.684,14.026,13.630,4.920,9.978,3.216,0.984,0.372,4.582,1.342,3.596)
> pvar
 [1] 19.684 14.026 13.630  4.920  9.978  3.216  0.984  0.372  4.582  1.342  3.596
> sqresult <- sqrt(result)
> sqresult
 [1]  9.273942  7.799231  6.594543  5.549234  4.576024  3.967367  3.309078  2.711088  3.091925
2.156850
[11] 2.095233
> residual <- pvar/sqresult
> plot(residual, type = "h", ylab = "standardized residuals", xlab = "interval index")
```

```
> setwd("~/Documents/math189")
> getwd()
[1] "/Users/misiyao/Documents/math189"
>
> set.seed(189189)
> n <-296
> randomsample <- runif(n,min=0,max=229354)
> hist(randomsample,breaks=57,probability=TRUE, col=4,
+        main="Uniform distribution Samples",
+        axes=FALSE)
> axis(2)
> axis(1,at=seq(0,229354,4000), labels=seq(0,229354,4000))
> lines(density(randomsample,adjust=2),col=2,lwd=2)
>
> data <- read.table("data case4.txt",header=TRUE)
> hist(data$location, breaks=57,probability=TRUE, col=4,
+        main= "Distribution of Palindromes in CMV DNA",
+        xlab = "Locations in CMV DNA",
+        ylab = "Palindrome density",
+        axes=FALSE)
> axis(2)
> axis(1,at=seq(0,229343,4000), labels=seq(0,229354,4000))
> lines(density(data$location,adjust=2),col=2,lwd=2)
> raw=data[,1]
> ta=table(cut(raw,breaks=57))
> b=as.vector(ta)
> chisq.test(b,p=rep(1/57,57))

    Chi-squared test for given probabilities

data:  b
X-squared = 72.189, df = 56, p-value = 0.0715
>
> k <- 41
> tab <- table(cut(sample, breaks = seq(0, 229354, length.out = k+1),
+                 include.lowest = TRUE))
> tab

         [0,5.59e+03]        (5.59e+03,1.12e+04]        (1.12e+04,1.68e+04]
(1.68e+04,2.24e+04]  (2.24e+04,2.8e+04]
                      7                          5                          7
8                     4
  (2.8e+04,3.36e+04]       (3.36e+04,3.92e+04]        (3.92e+04,4.48e+04]
(4.48e+04,5.03e+04] (5.03e+04,5.59e+04]
```

```
                              5                                        7                                        6
5                               10
(5.59e+04,6.15e+04]             (6.15e+04,6.71e+04]             (6.71e+04,7.27e+04]
(7.27e+04,7.83e+04] (7.83e+04,8.39e+04]
                              6                                       10                                        7
10                                2
(8.39e+04,8.95e+04]             (8.95e+04,9.51e+04]             (9.51e+04,1.01e+05]
(1.01e+05,1.06e+05] (1.06e+05,1.12e+05]
                              8                                       19                                        5
7                                 4
(1.12e+05,1.17e+05]             (1.17e+05,1.23e+05]             (1.23e+05,1.29e+05]
(1.29e+05,1.34e+05]    (1.34e+05,1.4e+05]
                              6                                        7                                        6
7                               10
  (1.4e+05,1.45e+05]            (1.45e+05,1.51e+05]             (1.51e+05,1.57e+05]
(1.57e+05,1.62e+05] (1.62e+05,1.68e+05]
                              9                                        7                                        8
3                               11
(1.68e+05,1.73e+05]             (1.73e+05,1.79e+05]             (1.79e+05,1.85e+05]
(1.85e+05,1.9e+05]    (1.9e+05,1.96e+05]
                              7                                        8                                        4
8                               14
(1.96e+05,2.01e+05]             (2.01e+05,2.07e+05]             (2.07e+05,2.13e+05]
(2.13e+05,2.18e+05] (2.18e+05,2.24e+05]
                              8                                        4                                        6
5                                 8
(2.24e+05,2.29e+05]
                              8
> Counts <- as.vector(tab)
> Counts
 [1]  7  5  7  8  4  5  7  6  5 10  6 10  7 10  2  8 19  5  7  4  6
7  6  7 10  9  7  8  3 11  7  8  4
[34]  8 14  8  4  6  5  8  8
>
> n=296
> k<-41
> E_i <- n/k
> p_i <- rep(E_i/n, k)
> chisq.test(Counts, p = p_i)

        Chi-squared test for given probabilities

data:  Counts
X-squared = 49.453, df = 40, p-value = 0.1453
```

```
>
> k1 <- 57
> tab <- table(cut(sample, breaks = seq(0, 229354, length.out = k1+1),
+                          include.lowest = TRUE))
> tab

          [0,4.02e+03]          (4.02e+03,8.05e+03]          (8.05e+03,1.21e+04]
(1.21e+04,1.61e+04] (1.61e+04,2.01e+04]
                    7                           1                            5
4                   8
(2.01e+04,2.41e+04]          (2.41e+04,2.82e+04]          (2.82e+04,3.22e+04]
(3.22e+04,3.62e+04] (3.62e+04,4.02e+04]
                    5                           1                            4
5                   3
(4.02e+04,4.43e+04]          (4.43e+04,4.83e+04]          (4.83e+04,5.23e+04]
(5.23e+04,5.63e+04] (5.63e+04,6.04e+04]
                    6                           3                            5
7                   3
(6.04e+04,6.44e+04]          (6.44e+04,6.84e+04]          (6.84e+04,7.24e+04]
(7.24e+04,7.65e+04] (7.65e+04,8.05e+04]
                    8                           7                            4
10                   2
(8.05e+04,8.45e+04]          (8.45e+04,8.85e+04]          (8.85e+04,9.25e+04]
(9.25e+04,9.66e+04] (9.66e+04,1.01e+05]
                    1                           7                            8
13                   4
(1.01e+05,1.05e+05]          (1.05e+05,1.09e+05]          (1.09e+05,1.13e+05]
(1.13e+05,1.17e+05] (1.17e+05,1.21e+05]
                    6                           3                            2
5                   7
(1.21e+05,1.25e+05]          (1.25e+05,1.29e+05]          (1.29e+05,1.33e+05]
(1.33e+05,1.37e+05] (1.37e+05,1.41e+05]
                    2                           5                            5
6                   7
(1.41e+05,1.45e+05]          (1.45e+05,1.49e+05]          (1.49e+05,1.53e+05]
(1.53e+05,1.57e+05] (1.57e+05,1.61e+05]
                    8                           6                            6
3                   1
(1.61e+05,1.65e+05]          (1.65e+05,1.69e+05]          (1.69e+05,1.73e+05]
(1.73e+05,1.77e+05] (1.77e+05,1.81e+05]
                    8                           8                            4
5                   6
(1.81e+05,1.85e+05]          (1.85e+05,1.89e+05]          (1.89e+05,1.93e+05]
```

```
(1.93e+05,1.97e+05] (1.97e+05,2.01e+05]
                  1                         6                         7
12                   5
(2.01e+05,2.05e+05]       (2.05e+05,2.09e+05]       (2.09e+05,2.13e+05]
(2.13e+05,2.17e+05] (2.17e+05,2.21e+05]
                  2                         6                         2
5                  1
(2.21e+05,2.25e+05] (2.25e+05,2.29e+05]
                  8                    7
> Counts_2 <- as.vector(tab)
> Counts_2
 [1]  7  1  5  4  8  5  1  4  5  3  6  3  5  7  3  8  7  4 10  2  1
7  8 13  4  6  3  2  5  7  2  5  5
[34]  6  7  8  6  6  3  1  8  8  4  5  6  1  6  7 12  5  2  6  2  5
1  8  7
>
> n=296
> k1<-57
> E_i <- n/k1
> p_i <- rep(E_i/n, k1)
> chisq.test(Counts_2, p = p_i)

	Chi-squared test for given probabilities

data:  Counts_2
X-squared = 77.966, df = 56, p-value = 0.02778



>
>
>
> SampleK <- 57
> Sampletab <- table(cut(sample, breaks = seq(0, 229354, length.out = SampleK+1),
+                        include.lowest = TRUE))
> Sampletab

       [0,4.02e+03]       (4.02e+03,8.05e+03]       (8.05e+03,1.21e+04]
(1.21e+04,1.61e+04] (1.61e+04,2.01e+04]
                  7                         1                         5
4                  8
(2.01e+04,2.41e+04]       (2.41e+04,2.82e+04]       (2.82e+04,3.22e+04]
(3.22e+04,3.62e+04] (3.62e+04,4.02e+04]
                  5                         1                         4
```

```
5                    3
(4.02e+04,4.43e+04]          (4.43e+04,4.83e+04]          (4.83e+04,5.23e+04]
(5.23e+04,5.63e+04] (5.63e+04,6.04e+04]
                   6                             3                             5
7                    3
(6.04e+04,6.44e+04]          (6.44e+04,6.84e+04]          (6.84e+04,7.24e+04]
(7.24e+04,7.65e+04] (7.65e+04,8.05e+04]
                   8                             7                             4
10                   2
(8.05e+04,8.45e+04]          (8.45e+04,8.85e+04]          (8.85e+04,9.25e+04]
(9.25e+04,9.66e+04] (9.66e+04,1.01e+05]
                   1                             7                             8
13                   4
(1.01e+05,1.05e+05]          (1.05e+05,1.09e+05]          (1.09e+05,1.13e+05]
(1.13e+05,1.17e+05] (1.17e+05,1.21e+05]
                   6                             3                             2
5                    7
(1.21e+05,1.25e+05]          (1.25e+05,1.29e+05]          (1.29e+05,1.33e+05]
(1.33e+05,1.37e+05] (1.37e+05,1.41e+05]
                   2                             5                             5
6                    7
(1.41e+05,1.45e+05]          (1.45e+05,1.49e+05]          (1.49e+05,1.53e+05]
(1.53e+05,1.57e+05] (1.57e+05,1.61e+05]
                   8                             6                             6
3                    1
(1.61e+05,1.65e+05]          (1.65e+05,1.69e+05]          (1.69e+05,1.73e+05]
(1.73e+05,1.77e+05] (1.77e+05,1.81e+05]
                   8                             8                             4
5                    6
(1.81e+05,1.85e+05]          (1.85e+05,1.89e+05]          (1.89e+05,1.93e+05]
(1.93e+05,1.97e+05] (1.97e+05,2.01e+05]
                   1                             6                             7
12                   5
(2.01e+05,2.05e+05]          (2.05e+05,2.09e+05]          (2.09e+05,2.13e+05]
(2.13e+05,2.17e+05] (2.17e+05,2.21e+05]
                   2                             6                             2
5                    1
(2.21e+05,2.25e+05] (2.25e+05,2.29e+05]
                   8                   7
> SampleCounts <- as.vector(SampleTab)
> SampleCounts
 [1]  7  1  5  4  8  5  1  4  5  3  6  2  6  7  3  8  7  4 10  2  1
7  7 14  4  6  3  2  5  7  1  6  5
[34]  6  7  8  5  7  3  1  8  8  4  5  6  1  6  7 10  5  4  6  2  5
```

```
1    8    7
>
> Stab <- matrix(rep(NA, 1*38), 38,1)
> Stab[1,] <- SampleTab[1, ]
> Stab[2,] <- SampleCounts[2]+SampleCounts[3]
> Stab[3,] <- SampleCounts[4]+SampleCounts[5]
> Stab[4,] <- SampleTab[6, ]
> Stab[5,] <- SampleCounts[7]+SampleCounts[8]
> Stab[6,] <- SampleCounts[9]+SampleCounts[10]
> Stab[7,] <- SampleTab[11, ]
> Stab[8,] <- SampleCounts[12]+SampleCounts[13]
> Stab[9,] <- SampleCounts[14]+SampleCounts[15]
> Stab[10,] <- SampleTab[16, ]
> Stab[11,] <- SampleCounts[17]+SampleCounts[18]
> Stab[12,] <- SampleTab[19, ]
> Stab[13,] <- SampleCounts[20]+SampleCounts[21]+SampleCounts[22]
> Stab[14,] <- SampleTab[23, ]
> Stab[15,] <- SampleTab[24, ]
> Stab[16,] <- SampleCounts[25]+SampleCounts[26]
> Stab[17,] <- SampleCounts[27]+SampleCounts[28]
> Stab[18,] <- SampleTab[29,]
> Stab[19,] <- SampleTab[30,]
> Stab[20,] <- SampleCounts[31]+SampleCounts[32]
> Stab[21,] <- SampleTab[33,]
> Stab[22,] <- SampleTab[34,]
> Stab[23,] <- SampleTab[35,]
> Stab[24,] <- SampleTab[36,]
> Stab[25,] <- SampleTab[37,]
> Stab[26,] <- SampleCounts[38]+SampleCounts[39]
> Stab[27,] <- SampleCounts[40]+SampleCounts[41]
> Stab[28,] <- SampleTab[42,]
> Stab[29,] <- SampleCounts[43]+SampleCounts[44]
> Stab[30,] <- SampleTab[45,]
> Stab[31,] <- SampleCounts[46]+SampleCounts[47]
> Stab[32,] <- SampleTab[48,]
> Stab[33,] <- SampleTab[49,]
> Stab[34,] <- SampleCounts[50]+SampleCounts[51]
> Stab[35,] <- SampleTab[52,]
> Stab[36,] <- SampleCounts[53]+SampleCounts[54]
> Stab[37,] <- SampleCounts[55]+SampleCounts[56]
> Stab[38,] <- SampleTab[57,]
> Scounts <- as.vector(Stab)
> Scounts
 [1]  7  6 12  5  5  8  6  8 10  8 11 10 10  7 14 10  5  5  7  7  5  6
```

```
7    8    5 10    9    8    9    6    7    7 10
[34]    9    6    7    9    7
>
>
> E_i <- 296/38
> p_i <- rep(E_i/n, 38)
> chisq.test(Scounts, p = p_i)

        Chi-squared test for given probabilities

data:    Scounts
X-squared = 21.865, df = 37, p-value = 0.9772
```

project3

March 3, 2019

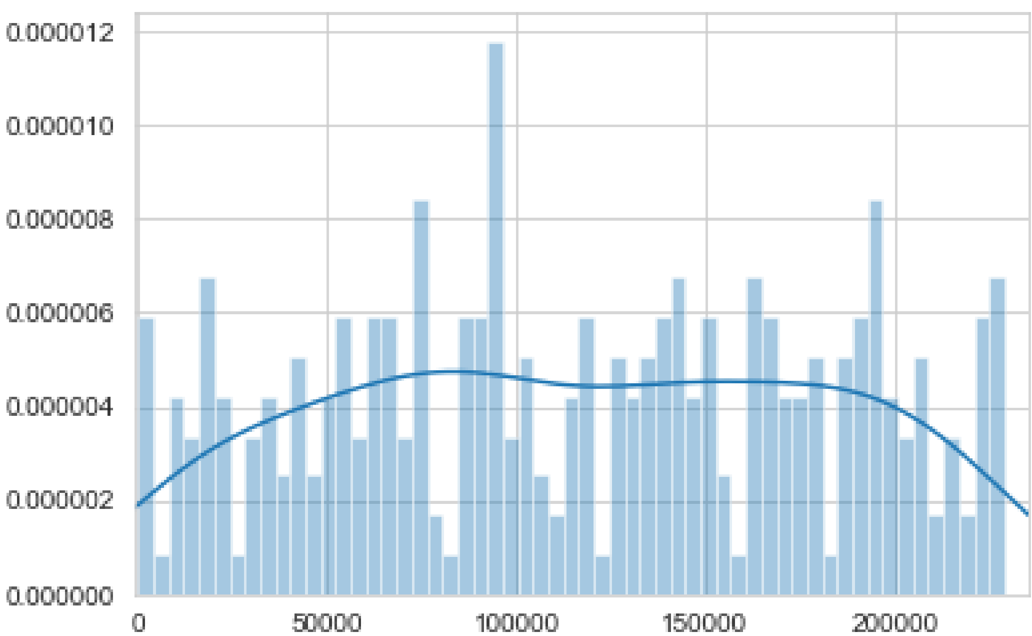
In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import norm
from scipy.stats import linregress
from sklearn.utils import resample
import seaborn as sns
import math
```

In [2]:
```
raw=np.loadtxt('3-data.txt')
```

In [3]:
```
sns.set_style("whitegrid")
```

In [6]:
```
sns.distplot(raw,bins=57,kde=True)
plt.xlim([-1000,235000])
```

Out[6]:
```
(-1000, 235000)
```

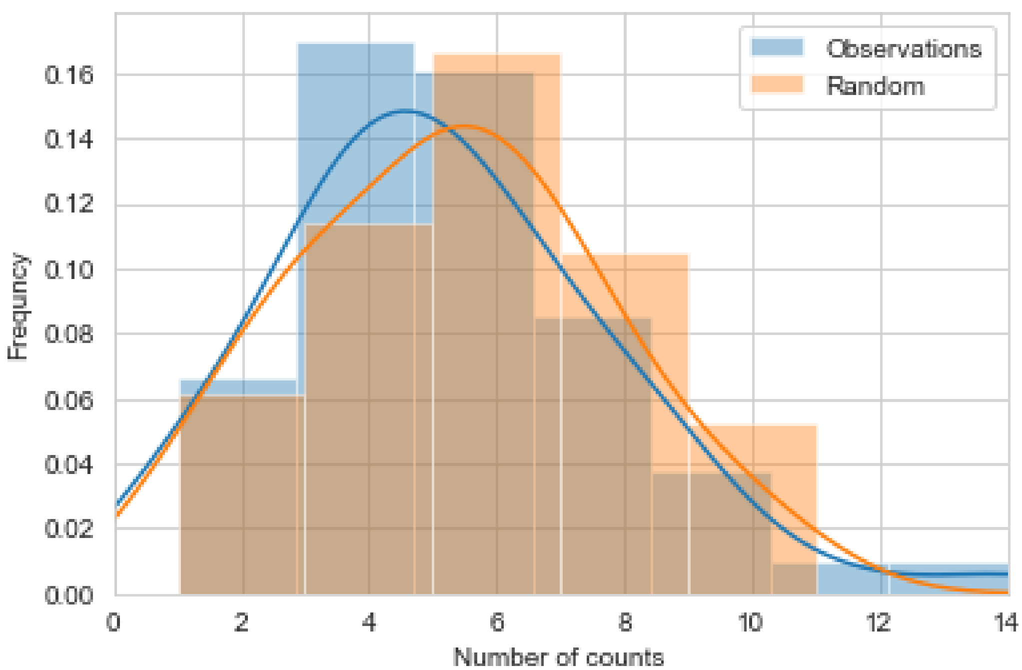


In [4]:
```
# Divide the data by interval size 4000
Palindrome_counts=[7,1,5,3,8,6,1,4,5,3,6,2,5,8,2,9,6,4,9,4,1,7,7,14,4
                   ,4,4,3,5,5,3,6,5,3,9,9,4,5,6,1,7,6,7,5,3,4,4,8,11,5
                   ,3,6,3,1,4,8,6]
```

In [5]:
```
E=296/57
```

In [44]:
```
sns.distplot(Palindrome_counts,kde=True,label='Observations')
s = np.random.poisson(E,57)
sns.distplot(s,kde=True,label='Random')
plt.xlabel('Number of counts')
plt.ylabel('Frequncy')
plt.xlim([0,14])
plt.legend()
```

Out[44]: <matplotlib.legend.Legend at 0x231cc2405f8>



In [7]:
```
#Define a calculation process for poisson
def poisson(n,m):
    E=0
    for k in range(n,m):
        E=E+57*np.exp(-296/57)*(296/57)**k/math.factorial(k)
    return E
```

In [23]:
```
def po(k):
    prob=np.exp(-296/57)*(296/57)**k/math.factorial(k)
    return prob
```

In [13]:
```
poisson(0,3)
```

Out[13]: 6.230737144534155

In [12]:
```
np.exp(-296/57)*(296/57)**3/math.factorial(3)*57
```

Out[12]: 7.390784615591427

In [14]:
```
np.exp(-296/57)*(296/57)**4/math.factorial(4)*57
```

Out[14]: 9.595053711469571

In [15]:
```
np.exp(-296/57)*(296/57)**5/math.factorial(5)*57
```

Out[15]: 9.965389117877168

In [18]:
```
np.exp(-296/57)*(296/57)**8/math.factorial(8)*57
```

Out[18]: 4.153417132554027

In [28]:
```
poisson(9,30)
```

Out[28]: 4.641095660536359

In [45]:
```
table={'Palindrome count':["0-2",3,4,5,6,7,8,'9+'],
'Number of Observed':[7,8,10,9,8,5,4,6],
'Interval expected':[6.23,7.39,9.60,9.97,8.63,6.40,4.15,4.64]}
pal=pd.DataFrame(table)
pal
```

Out[45]:

	Palindrome count	Number of Observed	Interval expected
0	0-2	7	6.23
1	3	8	7.39
2	4	10	9.60
3	5	9	9.97
4	6	8	8.63
5	7	5	6.40
6	8	4	4.15
7	9+	6	4.64

In [46]:
```
table_1={'Palindrome count':["0-2",3,4,5,6,7,'8+'],
'Number of Observed':[7,8,10,9,8,5,10],
'Interval expected':[6.23,7.39,9.60,9.97,8.63,6.40,8.79]}
```

In [47]:
```
pa2=pd.DataFrame(table_1)
pa2
```

Out[47]:

	Palindrome count	Number of Observed	Interval expected
0	0-2	7	6.23
1	3	8	7.39
2	4	10	9.60
3	5	9	9.97
4	6	8	8.63
5	7	5	6.40
6	8+	10	8.79

In [48]:
```
stats.chisquare(pa2['Number of Observed'],pa2['Interval expected'])
```

Out[48]:
```
Power_divergenceResult(statistic=0.7753651597433034, pvalue=0.992717984868693)
```

In [6]:
```
#Divide the data to 57 even size avergage 4015
hist, bins = np.histogram(raw, bins=57)
bin_counts = zip(bins, bins[1:], hist)  # [(bin_start, bin_end, count), ... ]
```

In [7]:
```
C0_2=[i for i in hist if i<=2]
C3=[i for i in hist if i==3]
C4=[i for i in hist if i==4]
C5=[i for i in hist if i==5]
C6=[i for i in hist if i==6]
C7=[i for i in hist if i==7]
C8=[i for i in hist if i>=8]
```

In [8]:
```
l={'Palindrome Count':["0-2",3,4,5,6,7,'8+'],
'Number of observed':[len(C0_2),len(C3),len(C4),len(C5),len(C6),len(C7),len(C8)],
'Interval expected':[6.23,7.39,9.60,9.97,8.63,6.40,8.79]}
pa3=pd.DataFrame(l)
pa3
```

Out[8]:

	Palindrome Count	Number of observed	Interval expected
0	0-2	10	6.23
1	3	4	7.39
2	4	7	9.60
3	5	10	9.97
4	6	7	8.63
5	7	12	6.40
6	8+	7	8.79

In [53]:
```
stats.chisquare(pa3['Number of observed'],pa3['Interval expected'])
```

Out[53]:
```
Power_divergenceResult(statistic=10.113093658831776, pvalue=0.11996948435507813)
```

In [22]:
```
set(bin_counts)
```

Out[22]:
```
{(177.0, 4190.6140350877195, 7),
 (4190.6140350877195, 8204.228070175439, 1),
 (8204.228070175439, 12217.842105263158, 5),
```

```
(12217.842105263158, 16231.456140350878, 4),
(16231.456140350878, 20245.070175438595, 8),
(20245.070175438595, 24258.684210526317, 5),
(24258.684210526317, 28272.298245614038, 1),
(28272.298245614038, 32285.912280701756, 4),
(32285.912280701756, 36299.52631578947, 5),
(36299.52631578947, 40313.14035087719, 3),
(40313.14035087719, 44326.754385964916, 6),
(44326.754385964916, 48340.36842105263, 3),
(48340.36842105263, 52353.98245614035, 5),
(52353.98245614035, 56367.596491228076, 7),
(56367.596491228076, 60381.210526315794, 4),
(60381.210526315794, 64394.82456140351, 7),
(64394.82456140351, 68408.43859649124, 7),
(68408.43859649124, 72422.05263157895, 4),
(72422.05263157895, 76435.66666666667, 10),
(76435.66666666667, 80449.28070175438, 2),
(80449.28070175438, 84462.8947368421, 1),
(84462.8947368421, 88476.50877192983, 7),
(88476.50877192983, 92490.12280701754, 7),
(92490.12280701754, 96503.73684210527, 14),
(96503.73684210527, 100517.35087719299, 4),
(100517.35087719299, 104530.9649122807, 6),
(104530.9649122807, 108544.57894736843, 3),
(108544.57894736843, 112558.19298245615, 2),
(112558.19298245615, 116571.80701754386, 5),
(116571.80701754386, 120585.42105263159, 7),
(120585.42105263159, 124599.0350877193, 1),
(124599.0350877193, 128612.64912280702, 6),
(128612.64912280702, 132626.26315789475, 5),
(132626.26315789475, 136639.87719298247, 6),
(136639.87719298247, 140653.49122807017, 7),
(140653.49122807017, 144667.1052631579, 8),
(144667.1052631579, 148680.71929824562, 5),
(148680.71929824562, 152694.33333333334, 7),
(152694.33333333334, 156707.94736842107, 3),
(156707.94736842107, 160721.56140350876, 1),
(160721.56140350876, 164735.1754385965, 8),
(164735.1754385965, 168748.7894736842, 7),
(168748.7894736842, 172762.40350877194, 5),
(172762.40350877194, 176776.01754385966, 5),
(176776.01754385966, 180789.6315789474, 6),
(180789.6315789474, 184803.24561403508, 1),
(184803.24561403508, 188816.8596491228, 6),
(188816.8596491228, 192830.47368421053, 7),
(192830.47368421053, 196844.08771929826, 10),
(196844.08771929826, 200857.70175438598, 5),
(200857.70175438598, 204871.31578947368, 4),
```

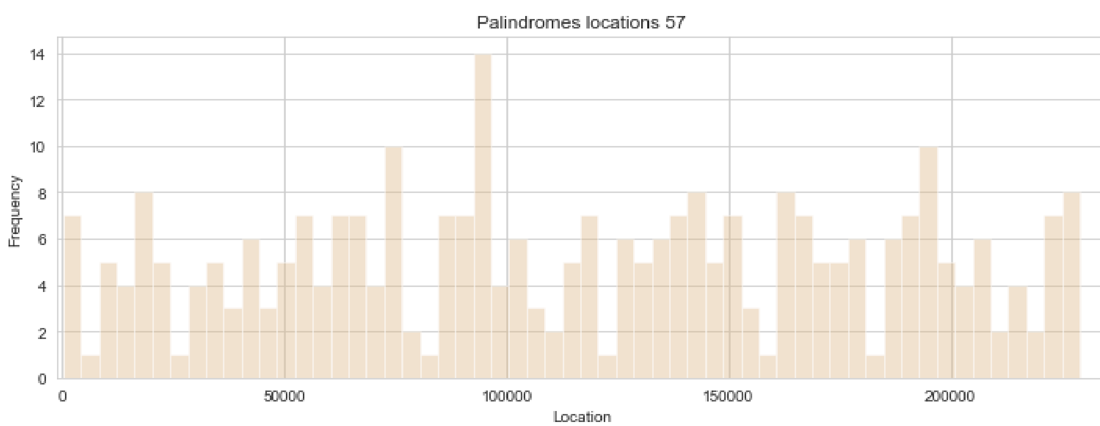
```
(204871.31578947368, 208884.9298245614, 6),
(208884.9298245614, 212898.54385964913, 2),
(212898.54385964913, 216912.15789473685, 4),
(216912.15789473685, 220925.77192982458, 2),
(220925.77192982458, 224939.3859649123, 7),
(224939.3859649123, 228953.0, 8)}
```

```
In [18]: hist
```

```
Out[18]: array([ 7,  1,  5,  4,  8,  5,  1,  4,  5,  3,  6,  3,  5,  7,  4,  7,  7,
                 4, 10,  2,  1,  7,  7, 14,  4,  6,  3,  2,  5,  7,  1,  6,  5,  6,
                 7,  8,  5,  7,  3,  1,  8,  7,  5,  5,  6,  1,  6,  7, 10,  5,  4,
                 6,  2,  4,  2,  7,  8], dtype=int64)
```

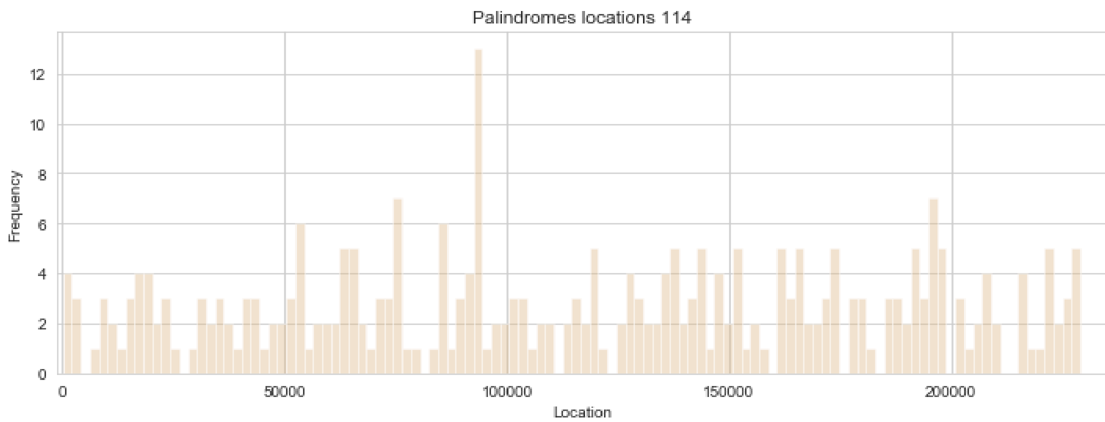
```
In [86]: plt.figure(figsize=(12,4))
         sns.distplot(raw,bins=57,kde=False, color='burlywood')
         plt.xlim([-1000,235000])
         plt.title('Palindromes locations 57')
         plt.xlabel('Location')
         plt.ylabel('Frequency')
```

```
Out[86]: Text(0, 0.5, 'Frequency')
```



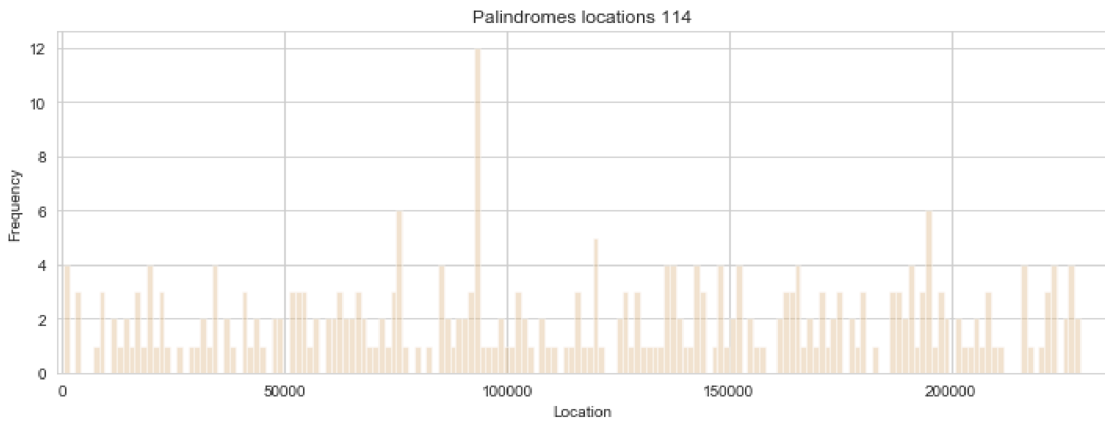

```
In [88]: plt.figure(figsize=(12,4))
         sns.distplot(raw,bins=114,kde=False,color='burlywood')
         plt.xlim([-1000,235000])
         plt.title('Palindromes locations 114')
         plt.xlabel('Location')
         plt.ylabel('Frequency')
```

```
Out[88]: Text(0, 0.5, 'Frequency')
```



In [89]:
```
plt.figure(figsize=(12,4))
sns.distplot(raw,bins=171,kde=False,color='burlywood')
plt.xlim([-1000,235000])
plt.title('Palindromes locations 171')
plt.xlabel('Location')
plt.ylabel('Frequency')
```

Out[89]: Text(0, 0.5, 'Frequency')



In [26]:
```
poisson_data=[po(i) for i in range(16)]
```

In [39]:
```
pdd=pd.DataFrame({'Counts':[i for i in range(16)],
                  'Probability':poisson_data}).transpose()
pdd
```

Out[39]:

	0	1	2	3	4	5	\
Counts	0.000000	1.000000	2.000000	3.000000	4.000000	5.000000	

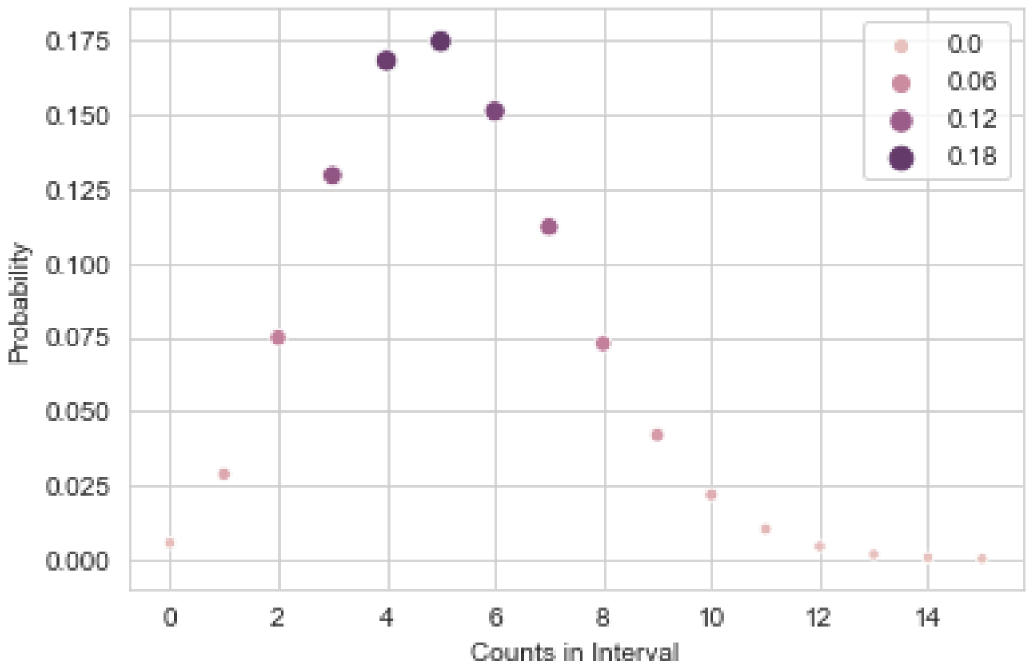
```
Probability  0.005555  0.028849  0.074907  0.129663  0.168334  0.174831

                    6         7         8         9          10          11  \
Counts       6.000000  7.000000  8.000000  9.000000  10.000000  11.000000
Probability  0.151316  0.112255  0.072867  0.042044   0.021833   0.010307

                   12         13         14         15
Counts       12.00000  13.000000  14.000000  15.000000
Probability   0.00446   0.001782   0.000661   0.000229
```

In [41]:
```python
cmap = sns.cubehelix_palette(dark=.3, light=.8, as_cmap=True)
sns.scatterplot(x=[i for i in range(16)],y=poisson_data, hue=poisson_data,size=poisso
plt.xlabel('Counts in Interval')
plt.ylabel('Probability')
```

Out[41]:
```
Text(0, 0.5, 'Probability')
```



In [150]:
```python
import jenkspy
breaks = jenkspy.jenks_breaks(raw, nb_class=57)
```

In [157]:
```python
hist1, bins1 = np.histogram(raw, bins=breaks)
bin_counts1 = zip(bins1, bins[1:], hist1)  # [(bin_start, bin_end, count), ... ]
```

In [160]:
```python
hist1
```

```
Out[160]: array([ 6,  4,  3,  6,  6,  4,  2,  3,  5,  3,  4,  3,  4,  4,  6,  2,  4,
                  6,  6,  5,  9,  2,  7,  5, 16,  5,  4,  3,  4,  6,  7,  6,  3,  4,
                  9,  3,  7,  6,  6,  4,  8,  5,  3,  4,  5,  3,  4,  6,  7, 10,  5,
                  3,  3,  6,  5,  8,  9], dtype=int64)
```

```
In [158]: set(bin_counts1)
```

```
Out[158]: {(177.0, 4190.6140350877195, 6),
           (3286.0, 8204.228070175439, 4),
           (9333.0, 12217.842105263158, 3),
           (12863.0, 16231.456140350878, 6),
           (16812.0, 20245.070175438595, 6),
           (20832.0, 24258.684210526317, 4),
           (23241.0, 28272.298245614038, 2),
           (28665.0, 32285.912280701756, 3),
           (31503.0, 36299.52631578947, 5),
           (34723.0, 40313.14035087719, 3),
           (38626.0, 44326.754385964916, 4),
           (42376.0, 48340.36842105263, 3),
           (45188.0, 52353.98245614035, 4),
           (48699.0, 56367.596491228076, 4),
           (52629.0, 60381.210526315794, 6),
           (55075.0, 64394.82456140351, 2),
           (57123.0, 68408.43859649124, 4),
           (61441.0, 72422.05263157895, 6),
           (64502.0, 76435.66666666667, 6),
           (68221.0, 80449.28070175438, 5),
           (72553.0, 84462.8947368421, 9),
           (76124.0, 88476.50877192983, 2),
           (79724.0, 92490.12280701754, 7),
           (86137.0, 96503.73684210527, 5),
           (90763.0, 100517.35087719299, 16),
           (94174.0, 104530.9649122807, 5),
           (99709.0, 108544.57894736843, 4),
           (102711.0, 112558.19298245615, 3),
           (105534.0, 116571.80701754386, 4),
           (110224.0, 120585.42105263159, 6),
           (117097.0, 124599.0350877193, 7),
           (121370.0, 128612.64912280702, 6),
           (127587.0, 132626.26315789475, 3),
           (129537.0, 136639.87719298247, 4),
           (134221.0, 140653.49122807017, 9),
           (138111.0, 144667.1052631579, 3),
           (141201.0, 148680.71929824562, 7),
           (143738.0, 152694.33333333334, 6),
           (148821.0, 156707.94736842107, 6),
           (152331.0, 160721.56140350876, 4),
           (157617.0, 164735.1754385965, 8),
```

```
 (164072.0, 168748.7894736842, 5),
 (166372.0, 172762.40350877194, 3),
 (168815.0, 176776.01754385966, 4),
 (171607.0, 180789.6315789474, 5),
 (174260.0, 184803.24561403508, 3),
 (178574.0, 188816.8596491228, 4),
 (182195.0, 192830.47368421053, 6),
 (188137.0, 196844.08771929826, 7),
 (192527.0, 200857.70175438598, 10),
 (195835.0, 204871.31578947368, 5),
 (198709.0, 208884.9298245614, 3),
 (202198.0, 212898.54385964913, 3),
 (206000.0, 216912.15789473685, 6),
 (210469.0, 220925.77192982458, 5),
 (217076.0, 224939.3859649123, 8),
 (223544.0, 228953.0, 9)}
```

In [131]:
```
#Find location
def centroid(start,end,pal):
    pal_interval=[j for j in pal if (j>=start) and (j<=end)]
    loc=[]
    for i in range(start,end):
        dis=sum([abs(k-i) for k in pal_interval])/len(pal_interval)
        loc.append([i,dis])
    return loc
```

In [161]:
```
distance1=centroid(90763,100517,raw)
```

In [162]:
```
disdf1=pd.DataFrame(distance1,columns=['Location','Mean Distance'])
```

In [164]:
```
distance2=centroid(192527, 200857,raw)
disdf2=pd.DataFrame(distance2,columns=['Location','Mean Distance'])
```

In [176]:
```
disdf1.loc[disdf1['Mean Distance'].idxmin()]
```

Out[176]:
```
Location         92783.000000
Mean Distance     1661.636364
Name: 2020, dtype: float64
```

In [178]:
```
disdf2.loc[disdf2['Mean Distance'].idxmin()]
```

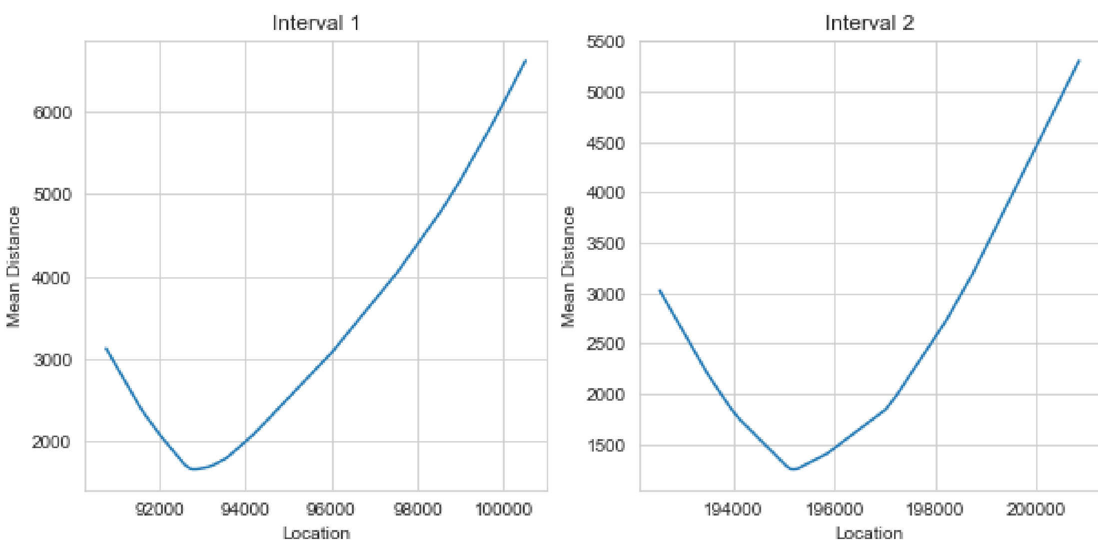
Out[178]:
```
Location         195151.00
Mean Distance      1251.75
Name: 2624, dtype: float64
```

In [166]:
```
fig=plt.figure(figsize=(10,10))
ax1=fig.add_subplot(221)
sns.lineplot(x=disdf1['Location'],y=disdf1['Mean Distance'],ax=ax1)
```

```
ax1.set_xlabel('Location')
ax1.set_ylabel('Mean Distance')
ax1.set_title('Interval 1')

ax2=fig.add_subplot(222)
sns.lineplot(x=disdf2['Location'],y=disdf2['Mean Distance'],ax=ax2)
ax2.set_xlabel('Location')
ax2.set_ylabel('Mean Distance')
ax2.set_title('Interval 2')
```

Out[166]: Text(0.5, 1.0, 'Interval 2')



In [ ]: