HW3 Code Part

Group Member:

- Yunlin Tang a14664383
- Yong Liu a15126460
- Jian Jiao a14525939

Set Up

In [4]:

```
# import library
library(lattice)
library(ggplot2)
library(scales)
```

```
Warning message:
"package 'ggplot2' was built under R version 3.6.3"
```

In [5]:

```
# read data
cmv <- read.table('hcmv.txt', header=TRUE)
# check # of rows in data
nrow(cmv)
```

296

In [6]:

```
# initialize the sizes, sites is n, bases is N
sites <- 296
bases <- 229354
```

In [9]:

```
cmv
```

location
177
1321
1433
1477
3248
3255
3286
7263
9023
9084
9333
10884
11754
12863
14263
14719
16013
16425
16752
16812
18009
19176
19325
19415
20030
20832
22027
22739
22910
23241
...
204548
205503
206000
207527
207788
207898
208572

location
209876
210469
215802
216190
216292
216539
217076
220549
221527
221949
222159
222573
222819
223001
223544
224994
225812
226936
227238
227249
227316
228424
228953

Locations (Random Scatter)

Here the goal is to graphically compare your sample palindrome locations to random uniform scatter. To do this, you can visualize the distribution of your sample, the distributions of random uniform scatter instances, and the theoretical uniform distribution. You can visualize the distributions using either histograms or empirical cdfs. Be sure to simulate the random uniform scatter several times (at least 5 times).

In [11]:

```
# Multiple plot function
#
# ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects)
# - cols:   Number of columns in layout
# - layout: A matrix specifying the layout. If present, 'cols' is ignored.
#
# If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE),
# then plot 1 will go in the upper left, 2 will go in the upper right, and
# 3 will go all the way across the bottom.
#
# function import from http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(gg
plot2)/
multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

 if (numPlots==1) {
    print(plots[[1]])

  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                      layout.pos.col = matchidx$col))
    }
  }
}
```

In [12]:

```
# convert the data into a vector
loc.vec <- c(cmv$location)
```

In [13]:

```
loc.vec
```

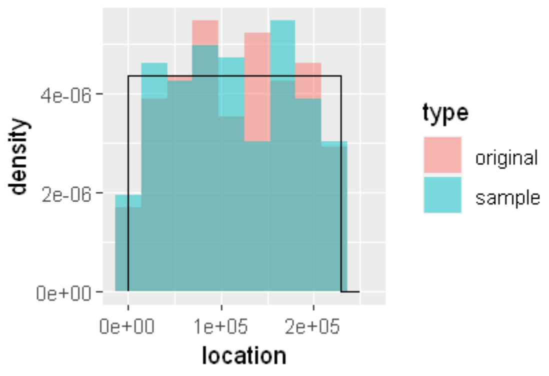
```
177   1321   1433   1477   3248   3255   3286   7263   9023   9084   9333   10884
11754   12863   14263   14719   16013   16425   16752   16812   18009   19176   19325
19415   20030   20832   22027   22739   22910   23241   25949   28665   30378   30990
31503   32923   34103   34398   34403   34723   36596   36707   38626   40554   41100
41222   42376   43475   43696   45188   47905   48279   48370   48699   51170   51461
52243   52629   53439   53678   54012   54037   54142   55075   56695   57123   60068
60374   60552   61441   62946   63003   63023   63549   63769   64502   65555   65789
65802   66015   67605   68221   69733   70800   71257   72220   72553   74053   74059
74541   75622   75775   75812   75878   76043   76124   77642   79724   83033   85130
85513   85529   85640   86131   86137   87717   88803   89586   90251   90763   91490
91637   91953   92526   92570   92643   92701   92709   92747   92783   92859   93110
93250   93511   93601   94174   95975   97488   98493   98908   99709   100864
102139   102268   102711   104363   104502   105534   107414   108123   109185
110224   113378   114141   115627   115794   115818   117097   118555   119665
119757   119977   120411   120432   121370   124714   125546   126815   127024
127046   127587   128801   129057   129537   131200   131734   133040   134221
135361   136051   136405   136578   136870   137380   137593   137695   138111
139080   140579   141201   141994   142416   142991   143252   143549   143555
143738   146667   147612   147767   147878   148533   148821   150056   151314
151806   152045   152222   152331   154471   155073   155918   157617   161041
161316   162682   162703   162715   163745   163995   164072   165071   165883
165891   165931   166372   168261   168710   168815   170345   170988   170989
171607   173863   174049   174132   174185   174260   177727   177956   178574
180125   180374   180435   182195   186172   186203   186210   187981   188025
188137   189281   189810   190918   190985   190996   191298   192527   193447
193902   194111   195032   195112   195117   195151   195221   195262   195835
196992   197022   197191   198195   198709   201023   201056   202198   204548
205503   206000   207527   207788   207898   208572   209876   210469   215802
216190   216292   216539   217076   220549   221527   221949   222159   222573
222819   223001   223544   224994   225812   226936   227238   227249   227316
228424   228953
```

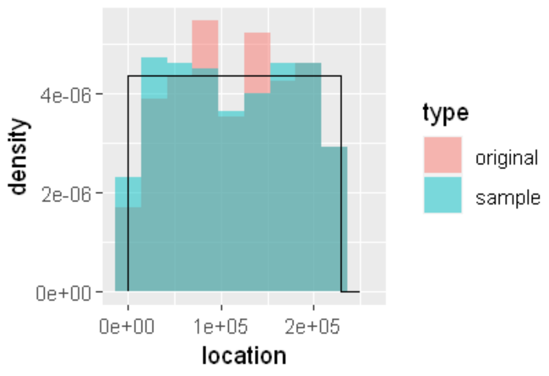
In [15]:

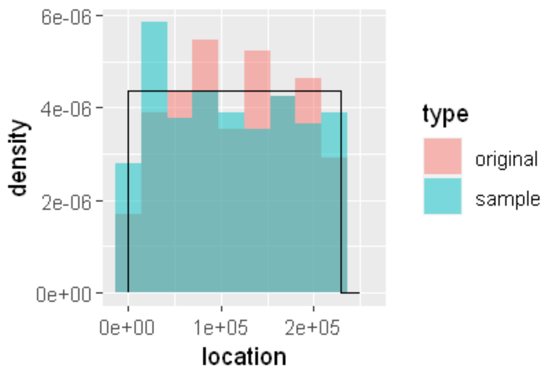
```
# construct the plot of 2 distributions and theoretical line for 6 times
for(i in 1:6){

    # simulate the random uniform scatter
    sample <- as.integer(runif(sites, min = 1, max=bases))

    # combine the random scatter and original data into one dataframe
    data <- data.frame(
            type = c(rep('original', sites), rep('sample', sites)),
            location = c(loc.vec, sample))

    # calculate the density of theoretical normal distribution
    x <- seq(from=0, to=250000, length.out=250000/2)
    y <- dunif(x, min=1, max=bases)
    unif <- data.frame(x=x, y=y)

    # plot two histograms for distributions + one theoretical distribution density
    plot <- ggplot() +
            geom_histogram(data, mapping=aes(x=location, y=..density.., fill=type), bin
s=10,
                           position='identity', alpha=0.5)+
            geom_line(data=unif,mapping=aes(x=x, y=y))

    # save plot
    name <- paste('plot', i, sep='_')
    assign(name, plot)
}
```

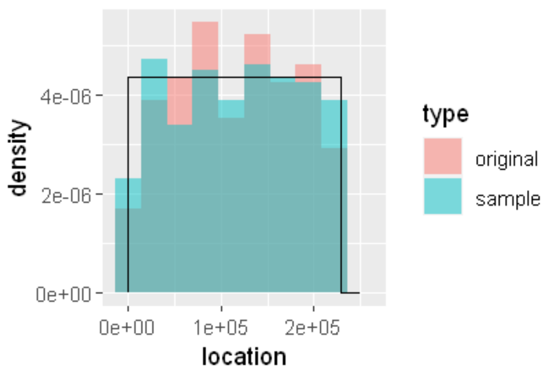
In [16]:

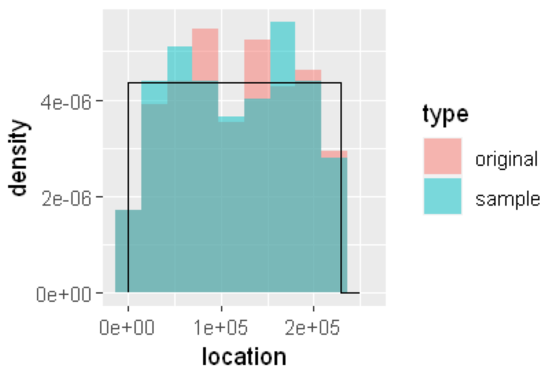
```
# combine 6 plots on one canvas
multiplot(plot_1, plot_2, plot_3, plot_4, plot_5, plot_6, cols=2)
```

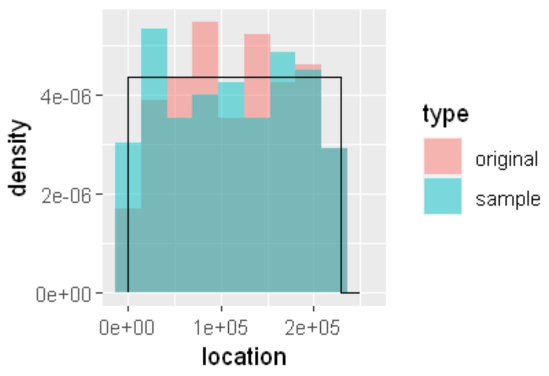

density
4e-06
2e-06
0e+00
0e+00
1e+05
2e+05
location
type
original
sample


density
4e-06
2e-06
0e+00
0e+00
1e+05
2e+05
location
type
original
sample


density
6e-06
4e-06
2e-06
0e+00
0e+00
1e+05
2e+05
location
type
original
sample


density
4e-06
2e-06
0e+00
0e+00
1e+05
2e+05
location
type
original
sample


density
4e-06
2e-06
0e+00
0e+00
1e+05
2e+05
location
type
original
sample


density
4e-06
2e-06
0e+00
0e+00
1e+05
2e+05
location
type
original
sample

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

Spacings

Here the goal is to graphically examine the distribution of your sample spacings. There are 3 types of spacings to examine: spacings between consecutive palindromes, spacings between palindromes with one in between (i.e. sums of pairs of consecutive spacings), and spacings between palindromes with two in between (i.e. sums of triplets of consecutive spacings). Next, you can graphically compare these 3 types of spacings to those that come from random uniform scatter (using empirical cdf or histograms). Again, you should simulate at least 5 random uniform scatters. Lastly, using theoretical results discussed in lecture, identify the theoretical distributions of the spacings from random uniform scatter (you won't be expected to know the distribution for the sums of consecutive triplets but it shouldn't be too hard to intuit). Overlay these theoretical distributions as a cdf or density on your plots.

In [ ]:

```

```

In [ ]:

```

```

Counts

Here the goal is to use graphical and formal statistical methods to examine the counts of palindromes in regions of the DNA. Be sure to do this for a few different (but reasonable) interval lengths. The graphical displays should compare the distribution of counts to random uniform scatter (this will be analogous to the locations and spacings sections). Using results from lecture, you can organize a formal statistical test to further examine your distributions of counts.

In [37]:

```
# create intervals
regionsplit <- function(n.region, gene, site){
  count.int <- table(cut(site, breaks = seq(1, length(gene), length.out=n.region+1), in
clude.lowest=TRUE))
  count.vector <- as.vector(count.int)
  count.tab <- table(factor(count.vector,levels=0:max(count.vector)))
  return (count.tab)
}
```

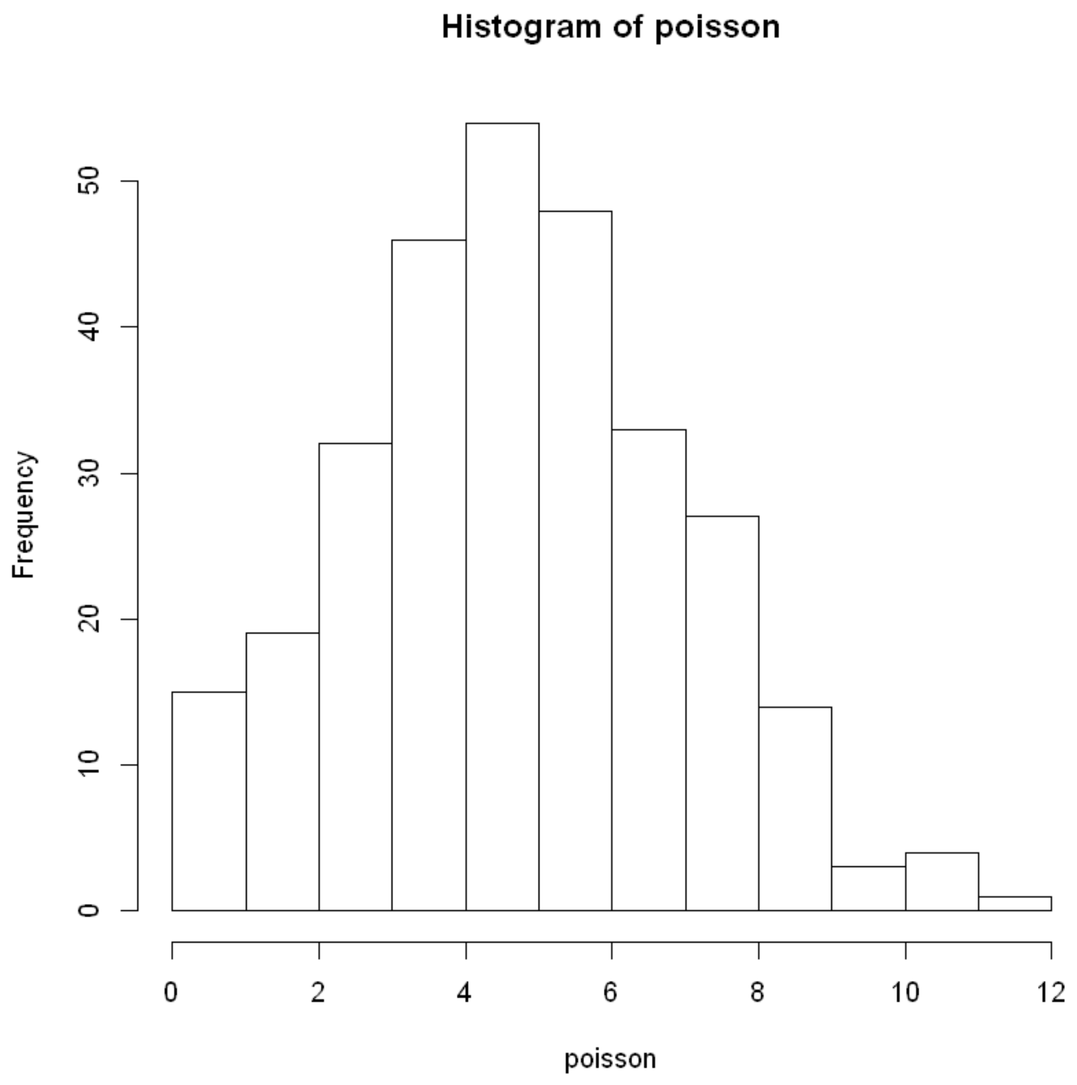
In [38]:

```
ranges <- c(30, 57, 100)
vectors <- vector(mode = "list", length = 3)
for (i in 1:3){
  vectors[[i]]=regionsplit(ranges[i], gene, data$location)
}
```

```
Error: $ operator is invalid for atomic vectors
Traceback:

1. regionsplit(ranges[i], gene, data$location)
2. table(cut(site, breaks = seq(1, length(gene), length.out = n.region +
 .     1), include.lowest = TRUE))   # at line 3 of file <text>
3. cut(site, breaks = seq(1, length(gene), length.out = n.region +
 .     1), include.lowest = TRUE)
```

In [39]:

```
poisson <- rpois(296, lambda=5)
hist(poisson)
```

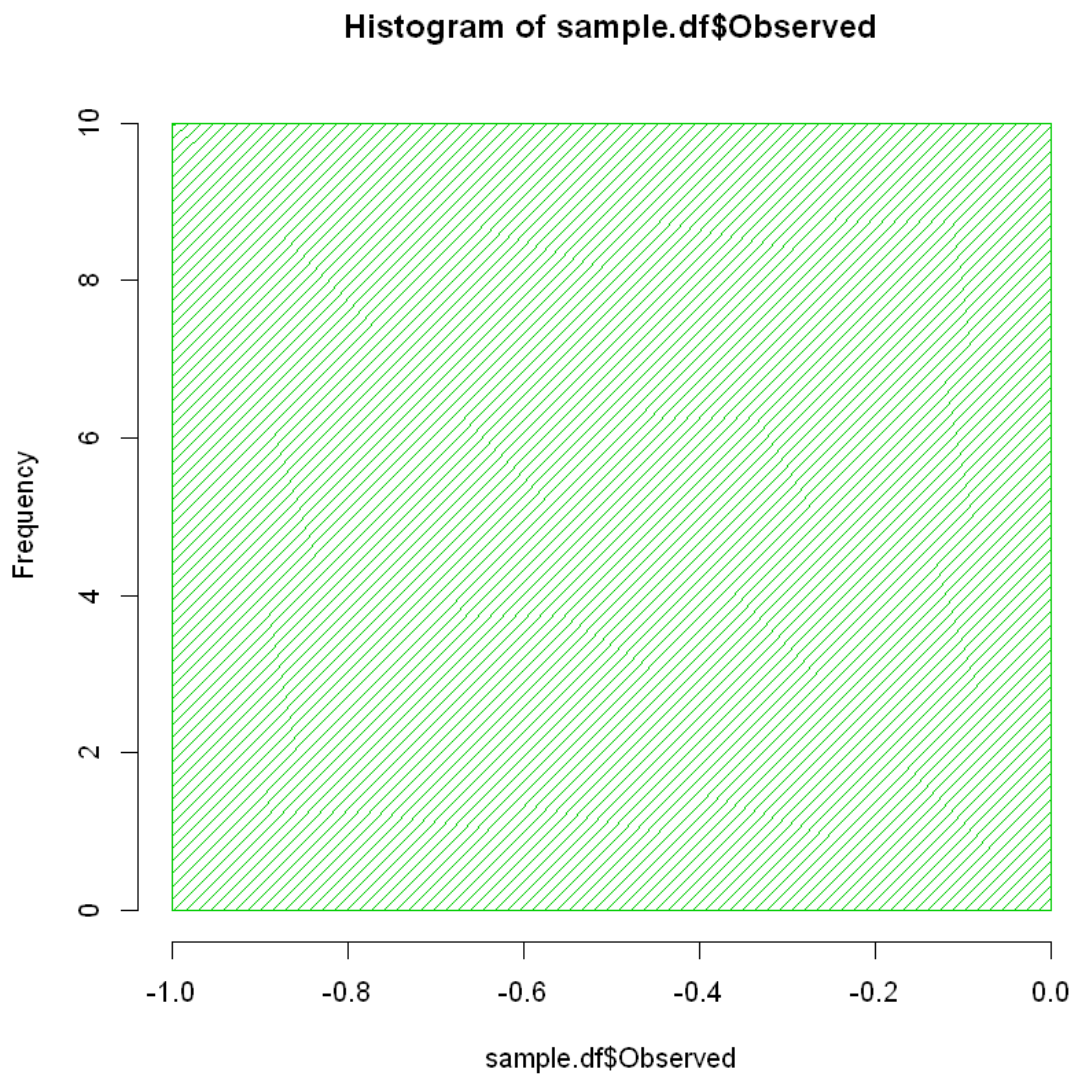

Histogram of poisson
Frequency
0
10
20
30
40
50
0
2
4
6
8
10
12
poisson

In [40]:

```
#30
trunc=9
lvls=factor(c(0:(trunc-1),paste(">=",trunc,sep="")),levels=c(0:(trunc-1),paste(">=",tru
nc,sep="")))

sample.vec=as.vector(vectors[[1]])
sample.trunc=c(sample.vec[1:trunc],sum(sample.vec[-(1:trunc)]))
lambda=n/ranges[1]
p=c(dpois(0:(trunc-1),lambda),1-sum(dpois(0:(trunc-1),lambda)))
E=p*ranges[1]
sample.df=data.frame(levels=lvls,Observed=sample.trunc,Expected=E)
hist(sample.df$Observed, breaks=20, probability = FALSE, density = 20, col = 3, border
= 3)
print(sample.df)
print(chisq.test(sample.trunc,p=p,simulate.p.value=TRUE))
```

```
   levels Observed     Expected
1       0        0  0.001556261
2       1        0  0.015355106
3       2        0  0.075751857
4       3        0  0.249139441
5       4        0  0.614543954
6       5        0  1.212700069
7       6        0  1.994217891
8       7        0  2.810897599
9       8        0  3.466773705
10    >=9        0 19.559064117
```

```
Error in chisq.test(sample.trunc, p = p, simulate.p.value = TRUE): at leas
t one entry of 'x' must be positive
Traceback:

1. print(chisq.test(sample.trunc, p = p, simulate.p.value = TRUE))
2. chisq.test(sample.trunc, p = p, simulate.p.value = TRUE)
3. stop("at least one entry of 'x' must be positive")
```


Histogram of sample.df$Observed
Frequency
0
2
4
6
8
10
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
sample.df$Observed

In [ ]:

```
# 57
trunc=9
lvls=factor(c(0:(trunc-1),paste(">=",trunc,sep="")),levels=c(0:(trunc-1),paste(">=",tru
nc,sep="")))

sample.vec=as.vector(vectors[[2]])
sample.trunc=c(sample.vec[1:trunc],sum(sample.vec[-(1:trunc)]))
lambda=n/ranges[2]
p=c(dpois(0:(trunc-1),lambda),1-sum(dpois(0:(trunc-1),lambda)))
E=p*ranges[2]
sample.df=data.frame(levels=lvls,Observed=sample.trunc,Expected=E)
hist(sample.df$Observed, breaks=20, probability = FALSE, density = 20, col = 3, border
= 3)
print(sample.df)
print(chisq.test(sample.trunc,p=p,simulate.p.value=TRUE))
```

In [ ]:

~~~
# 100 intervals
```{r}
trunc=9
lvls=factor(c(0:(trunc-1),paste(">=",trunc,sep="")),levels=c(0:(trunc-1),paste(">=",tru
nc,sep="")))

sample.vec=as.vector(vectors[[3]])
sample.trunc=c(sample.vec[1:trunc],sum(sample.vec[-(1:trunc)]))
lambda=n/ranges[3]
p=c(dpois(0:(trunc-1),lambda),1-sum(dpois(0:(trunc-1),lambda)))
E=p*ranges[3]
sample.df=data.frame(levels=lvls,Observed=sample.trunc,Expected=E)
hist(sample.df$Observed, breaks=20, probability = FALSE, density = 20, col = 3, border
= 3)
# df graph
print(sample.df)
# test stat
print(chisq.test(sample.trunc,p=p,simulate.p.value=TRUE))
~~~

Biggest Cluster

Here the goal is to use randomization or theory to examine the largest cluster of palindromes in a sub-interval. Again, you're expected to try a few different interval sizes. With respect to the randomization, focus on the probability of obtaining, in any subinterval, a count as large or larger than the count you observe in your sample. There is also a theoretical approach to obtaining such a probability. You are free to implement either method.

In [3]:

```
data <- cmv$location
N <- 229354 #size of DNA chain
n <- 296 #number of palindromes
k <- 58 #interval size
```

```
Error in eval(expr, envir, enclos): 找不到对象'cmv'
Traceback:
```

In [ ]:

```
choice <- c(200, 2000, 4000, 5000, 10000, 50000)
intervals <- ceiling(N / choice)
lambda <- c()
maxcount <- c()
p_value <- c()
```

In [ ]:

```
library(hash)
for(k in intervals) {
  dict <- hash()
  count <- as.vector(table(cut(data, breaks = seq(0, N, length.out = k+1), include.lowe
st = TRUE)))
  lambda <- c(lambda, mean(count))
  maxcount <- c(maxcount, max(count))


  for (i in 0:max(count)) {
    key <- toString(i)
    dict[[key]] <- 0
  }
  key <- toString(max(count)+1)
  dict[[key]] <- 0

  for (i in count) {
    dict[[toString(i)]] <- dict[[toString(i)]] + 1
  }

  observed <- c()
  for (i in 0:max(count)) {
    key <- toString(i)
    observed <- c(observed, dict[[key]])
  }
  observed <- c(observed, dict[[toString(max(count)+1)]])


  expected <- c()
  for (i in 0:max(count)) {
    expected <- c(expected, dpois(i, lambda))
  }
  expected <- c(expected, 1 - sum(expected))

  counts_expected <- expected * k

  chi<- sum((observed - counts_expected)^2 / expected)

  p_value <- c(p_value, pchisq(chi, df = max(count) - 2))

}
result <- data.frame(choice, intervals, lambda, maxcount, p_value)
result
```

In [106]:

```
data <- cmv$location
```

Advanced Analysis

Anything can further help us to answer the question: "How would you advise biologist who is about to start experimental searching for the origin of replication?"

In [ ]:

```
```

In [ ]:

```
```