```r
# set working directory
setwd("D:/STORAGE/College Work/year 4/Y4Q3/MATH 189/hw/hw3")

# read data
data <- read.table("hcmv.txt", header=TRUE)
head(data)


# Investigation 1
# global variables
numPalindromes <- 296
dnaLen <- 229354
binSeq <- seq(0, 232000, 4000)

# Graphing the observed distribution
hist(data[['location']], breaks = binSeq, ylim =c(0, 15),
     col = 'grey',
     main = "Distribution of Counts of Palindromes Bins=4000",
     xlab = 'Count intervals')
abline(a=296/58, b=0, lw = 3)


# Graphing several uniform distributions
# H1
sampleUnif = runif(numPalindromes, min = 0, max = length(binSeq))
h1 <- hist(sampleUnif, breaks= length(binSeq), plot = FALSE,
           main = 'Sample from Uniform Dist',
           xlab = 'Palindrome Occurrences per ')

# H2
sampleUnif = runif(numPalindromes, min = 0, max = length(binSeq))
h2 <- hist(sampleUnif, breaks= length(binSeq), plot = FALSE,
           main = 'Sample from Uniform Dist',
           xlab = 'Palindrome Occurrences per ')

# H3
sampleUnif = runif(numPalindromes, min = 0, max = length(binSeq))
h3 <- hist(sampleUnif, breaks= length(binSeq), plot = FALSE,
           main = 'Sample from Uniform Dist',
           xlab = 'Palindrome Occurrences per ')
# H4
sampleUnif = runif(numPalindromes, min = 0, max = length(binSeq))
h4 <- hist(sampleUnif, breaks= length(binSeq), plot = FALSE,
           main = 'Sample from Uniform Dist',
           xlab = 'Palindrome Occurrences per ')
# H5
sampleUnif = runif(numPalindromes, min = 0, max = length(binSeq))
h5 <- hist(sampleUnif, breaks= length(binSeq), plot = FALSE,
           main = 'Sample from Uniform Dist',
           xlab = 'Palindrome Occurrences per ')

# Setting up transparent colors for graph
c1 <- col2rgb("lightblue")

mycol1 <- rgb(c1[1], c1[2], c1[3], max = 255, alpha = 70, names = "blue50")

c1 <- col2rgb("pink")

mycol2 <- rgb(c1[1], c1[2], c1[3], max = 255, alpha = 70, names = "blue50")

c1 <- col2rgb("lightyellow")

mycol3 <- rgb(c1[1], c1[2], c1[3], max = 255, alpha = 70, names = "blue50")

c1 <- col2rgb("lightgreen")

mycol4 <- rgb(c1[1], c1[2], c1[3], max = 255, alpha = 70, names = "blue50")

c1 <- col2rgb("violet")

mycol5 <- rgb(c1[1], c1[2], c1[3], max = 255, alpha = 70, names = "blue50")


plot(h1, col = mycol1, ylim = c(0, 18), xlab = 'Bins (Size 4000 Intervals)')
plot(h2, col = mycol2, add = TRUE)
plot(h3, col = mycol3, add = TRUE)
plot(h4, col = mycol4, add = TRUE)
plot(h5, col = mycol5, add = TRUE)

abline(a=296/58, b=0, lw = 3)


# Investigation 2
# Complementary palidrome is one type of pattern in DNA that
# can be indiciative of an important site in DNA, such as the origin of
# replication.
# A is complementary to T
# G is complementary to C
# A complementary palindrome is a sequence of letters that
# read in reverse as the complement of the forward sequence.

library(lattice)
stripplot(data$location, pch=1, cex=0.25)
hist(data$location, breaks=nrow(data)*400)

# Goals: 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)
#   spacings between palindromes with two in between
#     (i.e. sums of triplets of consecutive spacings)

# Graphically compare these 3 types of spacings to those
# that come from random uniform scatter (using empirical cdf or histograms)

# create a random uniform scatter
N <- 200000
n <- nrow(data)
gene <- seq(1, N)
# set.seed(100)
site.random <- sort(sample.int(N, size=n, replace=FALSE))
stripplot(site.random, pch=16, cex=0.25)

rand_scatt = diff(site.random)
# stripplot(rand_scatt, pch=1, cex=0.25)
# hist(rand_scatt)
barplot(rand_scatt)
plot(ecdf(rand_scatt))

graph_mixed_plot <- function(data, title, subt) {
  data = diff(data)
  hist(data, xlim=c(0, max(data)), main=paste(title, '\n', subt))
  # mtext(sode=3, line=3, at=-0.07, adj=0, cex=1, title)
  # mtext(sode=3, line=3, at=-0.07, adj=0, cex=1, title)
  par(new = T)
  plot(ecdf(data),
       xlim=c(0, max(data)),
       col=rgb(0, 0, 0, alpha=0),
       axes=F,
       xlab=NA,
       ylab=NA,
       main=NA)
  lines(ecdf(data))
}

# spacings between consecutive palindromes
palin_consec = data$location
graph_mixed_plot(palin_consec,
                 "Consecutive palindrome distance distribution",
                 "Population data")

# Random uniform scatter comparisons
site.random <- sort(sample.int(N, size=n, replace=FALSE))
graph_mixed_plot(site.random,
                 "Consecutive palindrome distance distribution",
                 "Uniform Sampled data 1")
site.random <- sort(sample.int(N, size=n, replace=FALSE))
graph_mixed_plot(site.random,
                 "Consecutive palindrome distance distribution",
                 "Uniform Sampled data 2")
pnorm((mean(site.random) - mean(palin_consec)) / (sd(palin_consec) / length(palin_consec)))
mean(site.random) - mean(palin_consec)


# spacings between palindromes with one in between
palin_one_sep = data$location[seq(0, nrow(data), 2)]
graph_mixed_plot(palin_one_sep,
                 "Distribution of palindrome spacing with 1 between",
                 "Population data")

# Random uniform scatter comparisons
site.random <- sort(sample.int(N, size=n, replace=FALSE))[seq(0, nrow(data), 2)]
graph_mixed_plot(site.random,
                 "Distribution of palindrome spacing with 1 between",
                 "Uniform Sampled data 1")
site.random <- sort(sample.int(N, size=n, replace=FALSE))[seq(0, nrow(data), 2)]
graph_mixed_plot(site.random,
                 "Distribution of palindrome spacing with 1 between",
                 "Uniform Sampled data 2")


# spacings between palindromes with two in between
palin_two_sep = data$location[seq(0, nrow(data), 3)]
graph_mixed_plot(palin_two_sep,
                 "Distribution of palindrome spacing with 2 between",
                 "Population data")

# Random uniform scatter comparisons
site.random <- sort(sample.int(N, size=n, replace=FALSE))[seq(0, nrow(data), 3)]
graph_mixed_plot(site.random,
                 "Distribution of palindrome spacing with 2 between",
                 "Uniform Sampled data 1")
site.random <- sort(sample.int(N, size=n, replace=FALSE))[seq(0, nrow(data), 3)]
graph_mixed_plot(site.random,
                 "Distribution of palindrome spacing with 2 between",
                 "Uniform Sampled data 2")


# Investigation 3
get_bin_counts <- function(data_vec, bins, subt) {
  bin_counts <- c()
  bin_names <- c()
  bin_size <- (max(data_vec) / bins)

  for (i in seq(bins)) {
    lower_bound = (bin_size * (i - 1))
    upper_bound = (bin_size * i)
    bin_counts = append(bin_counts,
                        length(data_vec[(data_vec <= upper_bound) & (data_vec > lower_bound)]))
    bin_names = append(bin_names, as.character(round(upper_bound)))
  }

  barplot(bin_counts, names.arg=bin_names, las=2, main=paste("Location counts\n", subt))
  axis(side = 1, labels = FALSE)


  return (bin_size)
}

bins = 100

par(mfrow=c(2, 2), pin=c(5, 3))
bin_size = get_bin_counts(data$location, bins, "Population data")
plot.new()

sample_count_plots <- function(data_vec, bin_size, subt){
  bin_counts <- c()
  bin_names <- c()

  for (i in seq(bins)) {
    lower_bound = (bin_size * (i - 1))
    upper_bound = (bin_size * i)
    bin_counts = append(bin_counts,
                        length(data_vec[(data_vec <= upper_bound) & (data_vec > lower_bound)]))
    bin_names = append(bin_names, as.character(round(upper_bound)))
  }
  barplot(bin_counts, names.arg=bin_names, las=2, main=paste("Location counts\n", subt))
  axis(side = 1, labels = FALSE)
}


site.random <- sort(sample.int(N, size=n, replace=FALSE))
sample_count_plots(site.random, bin_size, "Sample data 1")
site.random <- sort(sample.int(N, size=n, replace=FALSE))
sample_count_plots(site.random, bin_size, "Sample data 2")

qqnorm(data$location, pch = 1, frame = FALSE, main="Population Q-Q Plot")
qqnorm(site.random, pch = 1, frame = FALSE, main="Sample Uniform Distribution Q-Q Plot")

chisq.test(data$location)
chisq.test(site.random)



### Investigation 4 BIGGEST CLUSTER ###

#Does the interval with the greatest number of palindromes
#indicate a potential origin of replication? Be careful in
#making your intervals, for any small, but significant
#deviations from random scatter, such as a tight cluster of
#a few palindromes, could easily go undetected if the regions
#examined are too large. Also, if the regions are too small,
#a cluster of palindromes may be split between adjacent
```

```r
#intervals and not appear as a high-count interval.

locations <- read.table('hcmv.txt', header=TRUE)$location
head(locations)

# Given (see slide 13-14).
N <- 229354  # Base pairs, or length of CMV genome
n <- 296  # Palindromes

intervals <- c(2500,4000,5500,7000)
k_intervals <- ceiling(N / intervals)
k_lambda_hat <- c()
k_max_count <- c()
k_p_value <- c()

for(k in k_intervals) {

  k_count <- as.vector(table(cut(locations, breaks = seq(0, N, length.out = k+1), include.lowest = TRUE)))
  lambda_hat <- mean(k_count)
  k_lambda_hat <- c(k_lambda_hat, lambda_hat)
  k_max_count <- c(k_max_count, max(k_count))

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

  for (c in k_count) {
    key <- toString(c)
    dict[[key]] <- dict[[key]] + 1
  }

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

  k_expected_poisson <- c()
  for (i in 0:max(k_count)) {
    k_expected_poisson <- c(k_expected_poisson, dpois(i, lambda_hat))
  }
  k_expected_poisson <- c(k_expected_poisson, 1 - sum(k_expected_poisson))

  k_counts_expected <- k_expected_poisson * k

  k_chi2<- sum((k_counts_observed - k_counts_expected)^2 / k_counts_expected)
  k_chi2_compare <- qchisq(p = 0.95, df = max(k_count) - 2)
  k_p_value <- c(k_p_value, pchisq(k_chi2, df = max(k_count) - 2, lower.tail = FALSE))

}
result <- data.frame(intervals, k_intervals, k_lambda_hat, k_max_count, k_p_value)
result
```