R Notebook

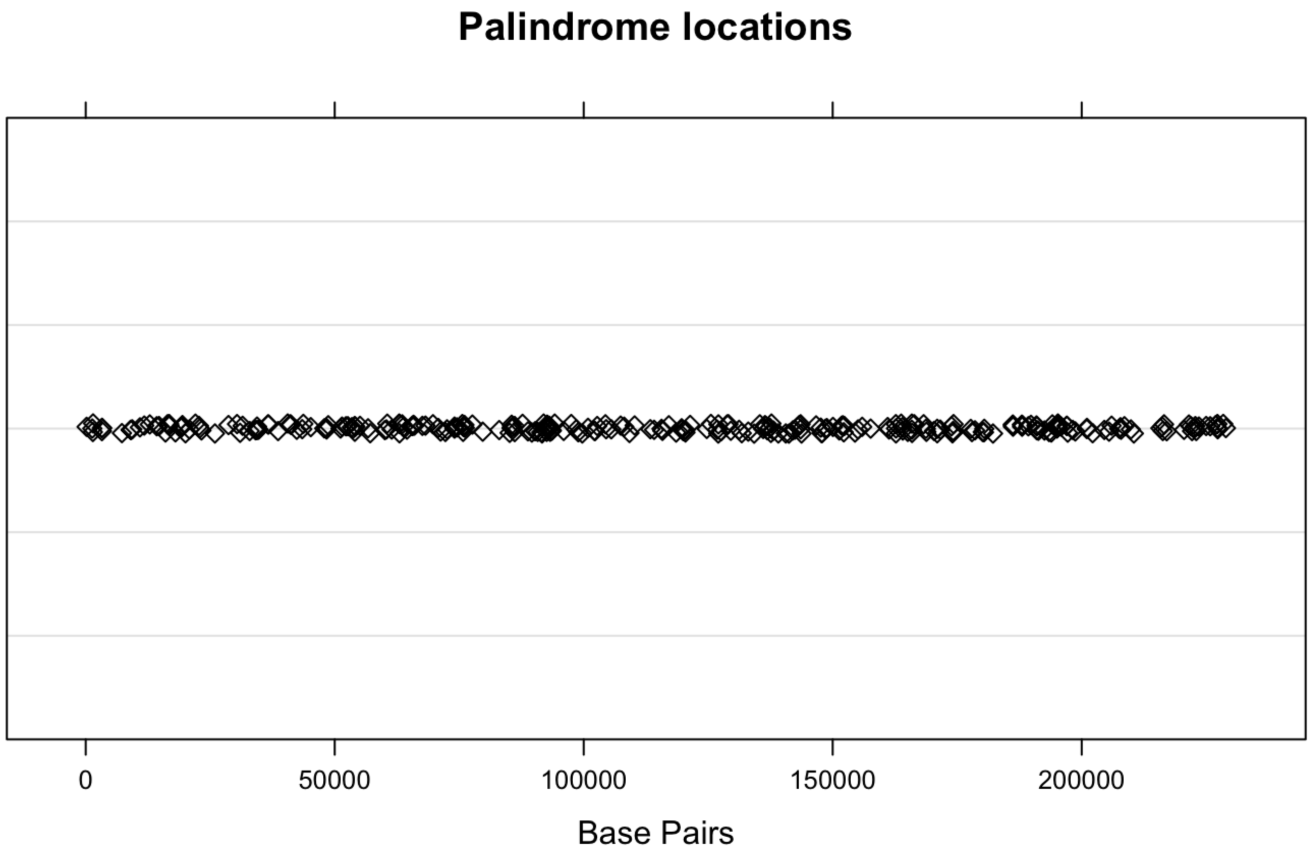
Code

This is an R Markdown (http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

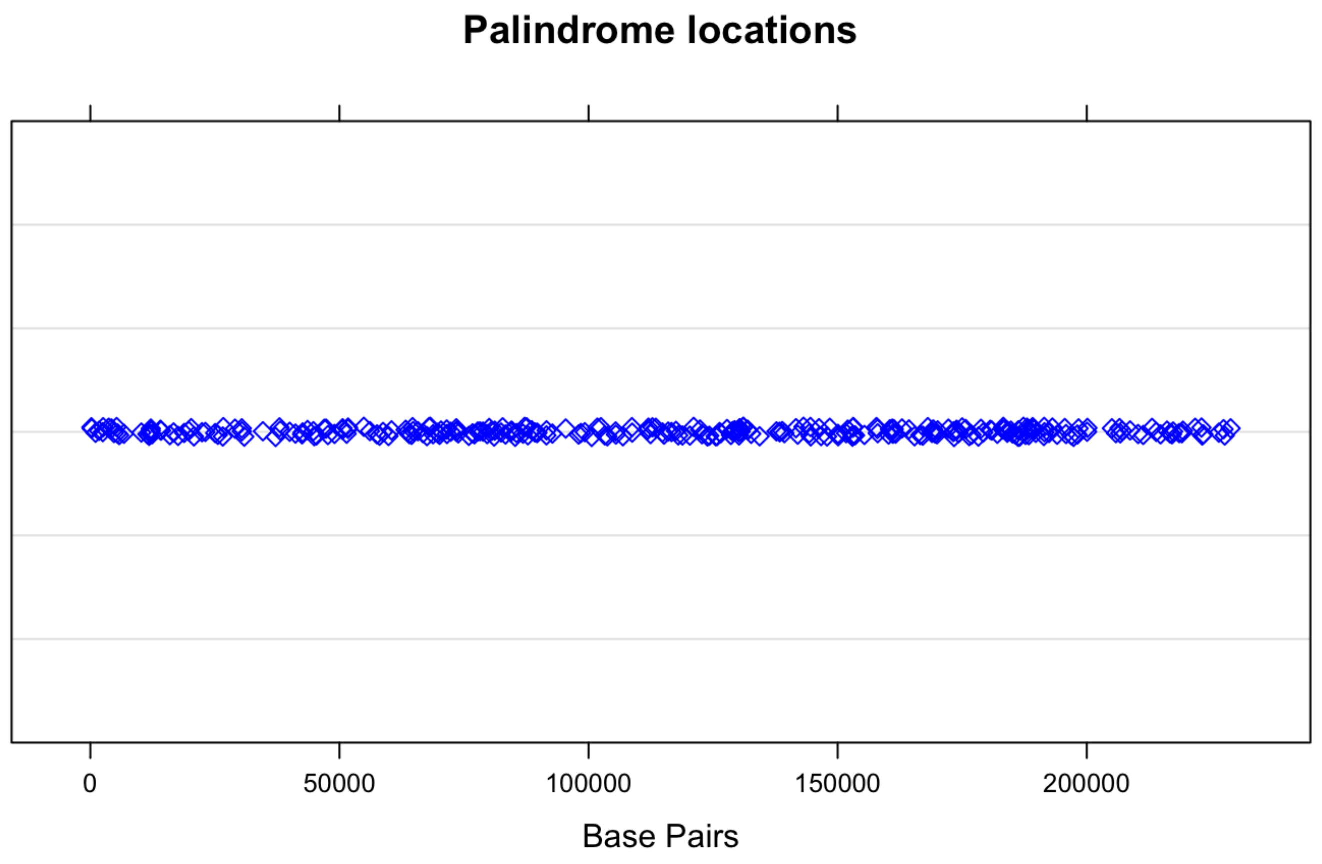
Hide

```
## Question 1: Random Scatter
set.seed(296)
data = read.csv("hcmv-263hxkx-1qhtfgz.txt", header=T)
site.original = data.matrix(data)
n.base <- 229354 #number of DNA sequence bases
n.site <- 296 #296 palindrome sites
Title1 = 'Palindrome locations'
library(lattice)
#Create a scatter plot of the locations of the original data
site.original.plot =stripplot(site.origianl, pch=5, cex=0.8,col="black",
           main=Title1, xlab= 'Base Pairs', jitter.data=T, grid="h")
site.original.plot
```



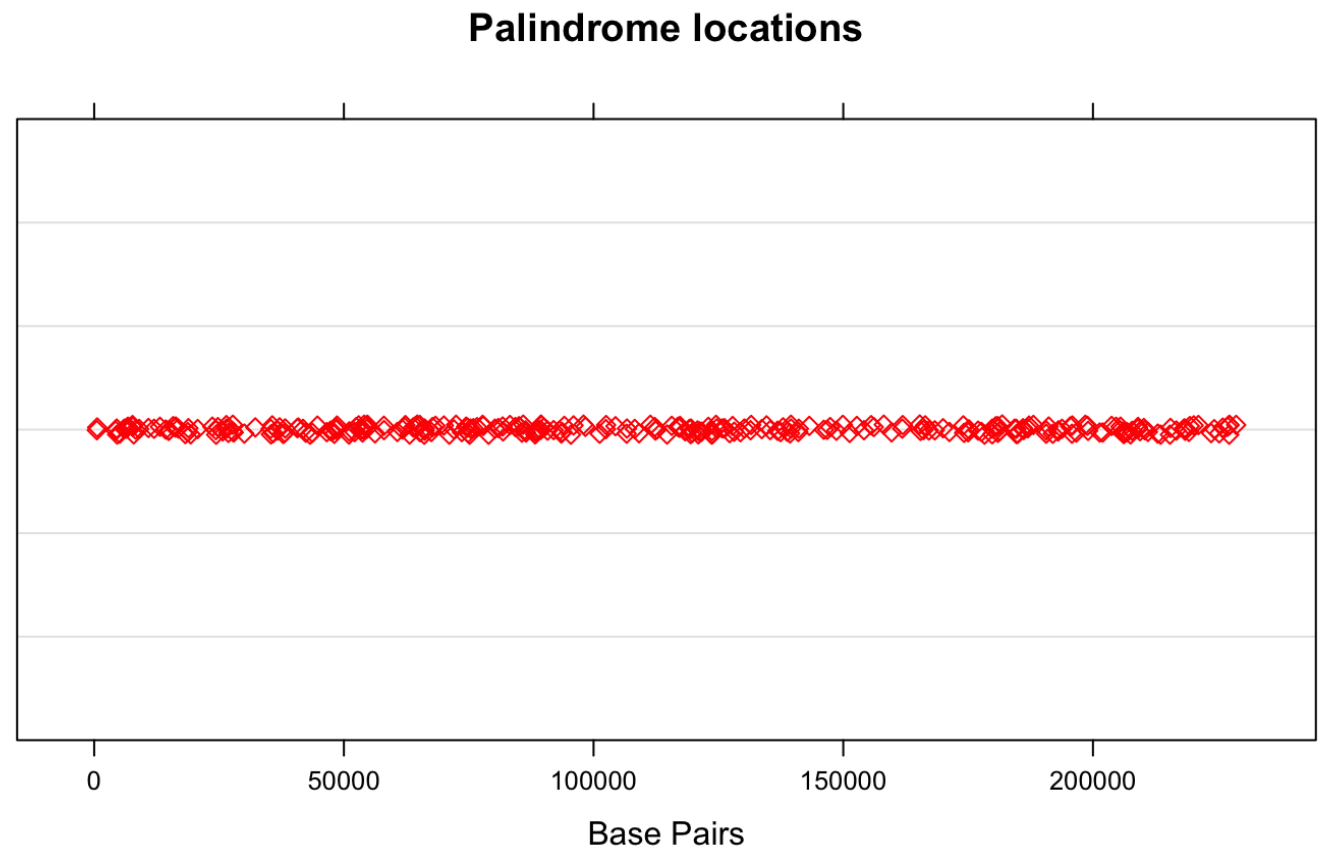
Hide

```
#randomly generate 3 different 296 panlindrome locations from 229354 DNA sequence bases
site.random1 = round(runif(n.site,0,n.base))
site.random2 = round(runif(n.site,0,n.base))
site.random3 = round(runif(n.site,0,n.base))
#plot 3 scatter plots of runif
stripplot(site.random1, pch=5, cex=0.8, jitter.data=T, grid = "h", col="blue", #3 differ
ent scatter plots
          main = Title1, xlab = 'Base Pairs') #1-dimensional scatter plot
```



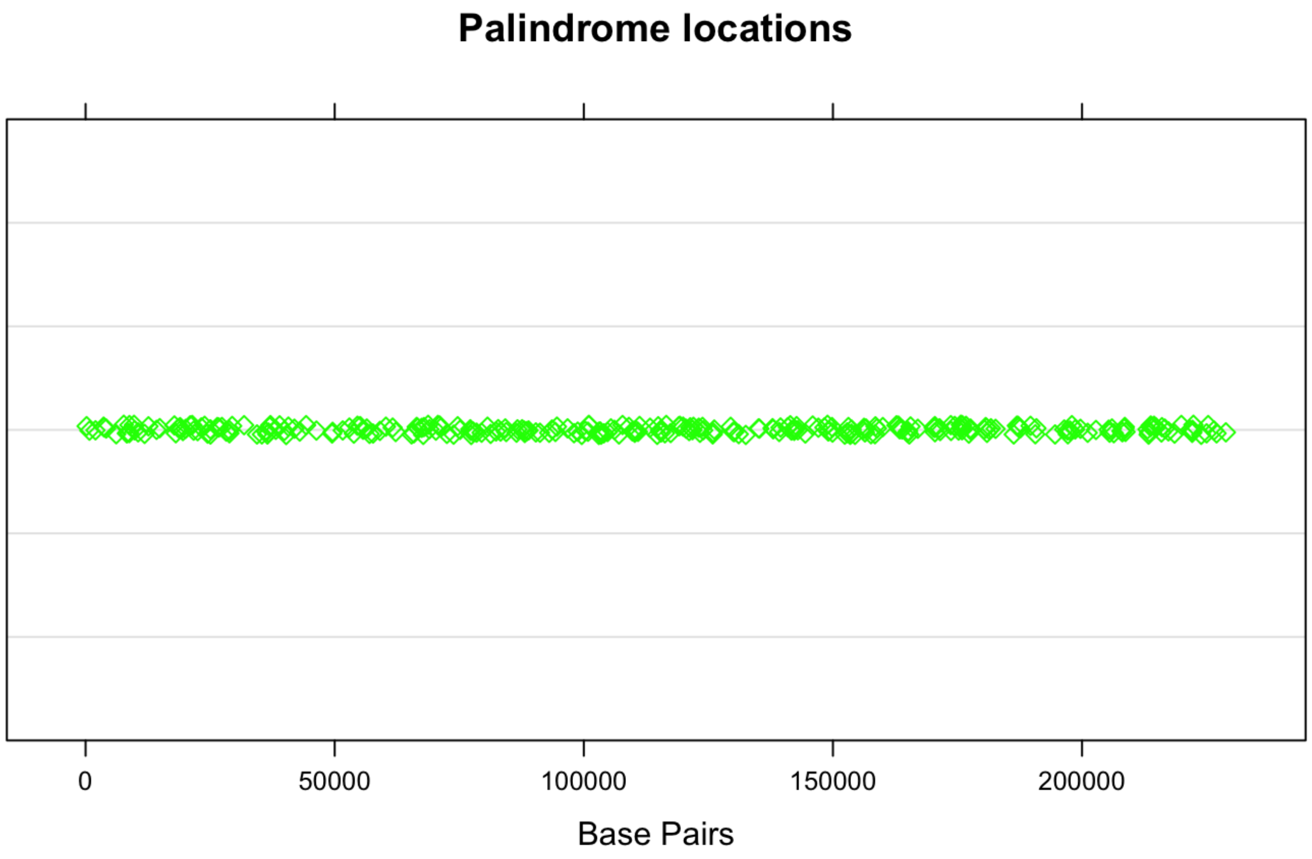
Hide

```
stripplot(site.random2, pch=5, cex=0.8, jitter.data=T, grid = "h", col="red",
          main = Title1, xlab = 'Base Pairs')
```



Hide

```
stripplot(site.random3, pch=5, cex=0.8, jitter.data=T, grid = "h", col="green",
          main = Title1, xlab = 'Base Pairs', add=T)
```

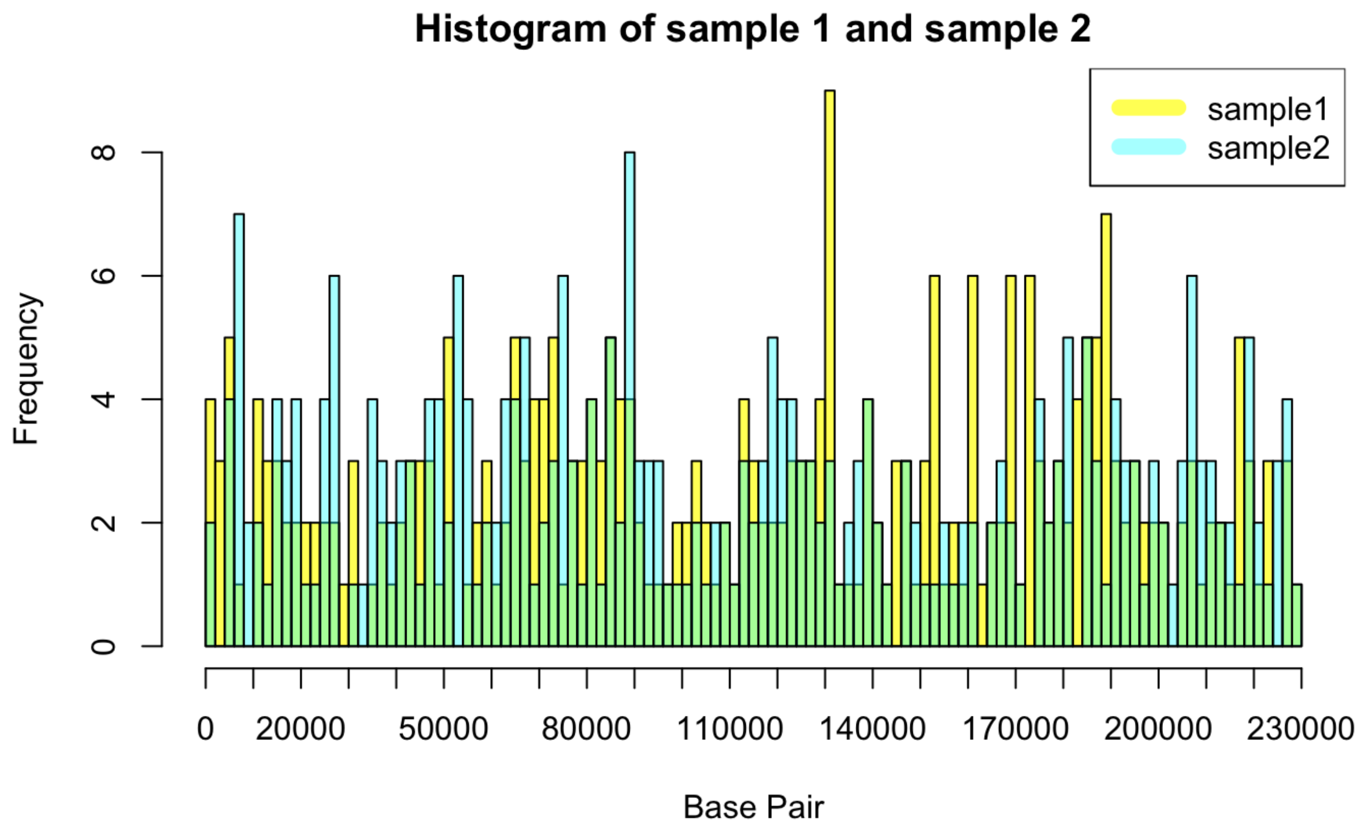


Hide

```
#plot histograms comparing sample 1 and sample 2
hist(site.random1, breaks=85, col=rgb(1,1,0,0.7), main="Histogram of sample 1 and sample
2", xlab="Base Pair",xaxt='n')
hist(site.random2, breaks=85, col=rgb(0,1,1,0.4), main="Histogram of sample 2", xlab="Ba
se Pair", add=T, xaxt='n')
```

Hide

```
axis(1, at=seq(0, n.base+10000, 10000))
legend("topright", c("sample1", "sample2"), col=c(rgb(1,1,0,0.7), rgb(0,1,1,0.4)), lwd=8
)
```

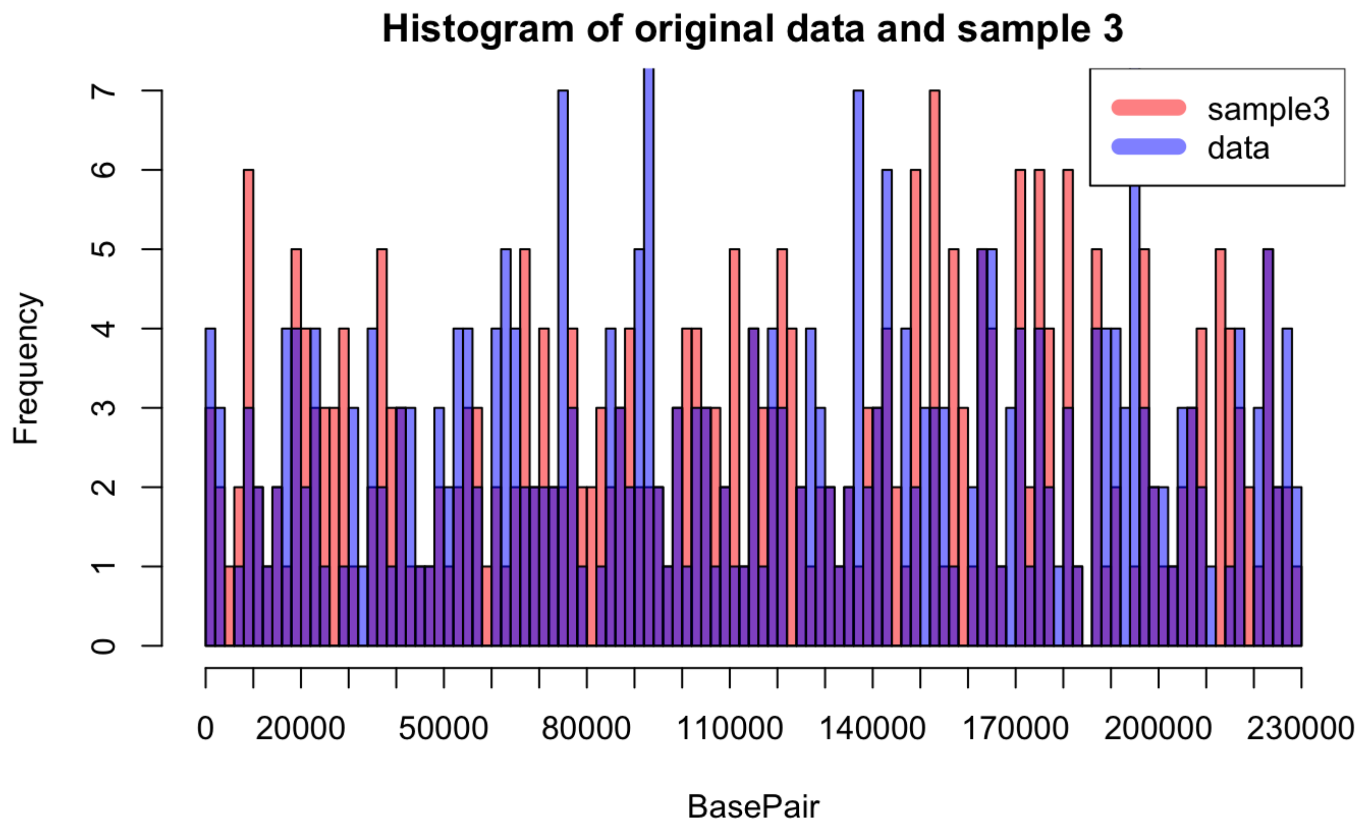


Hide

```
#plot histograms comparing original data and sample3
hist(site.random3, breaks=85, col=rgb(1,0,0,0.5), main="Histogram of original data and s
ample 3", xlab="BasePair", xaxt='n')
hist(site.origianl, breaks=85, col=rgb(0,0,1,0.5), main="Histogram of original data", xl
ab="Base Pair", add=T, xaxt='n')
```

Hide

```
axis(1, at=seq(0, n.base+10000, 10000))
legend("topright", c("sample3", "data"), col=c(rgb(1,0,0,0.5), rgb(0,0,1,0.5)), lwd=8)
```



Hide

```
for(x in c(51,57,66)){
 count.sample = rep(n.site/x, x)
 count.data = as.vector(table(cut(data, breaks=seq(0, n.base, length.out = x + 1), inclu
de.lowest=TRUE)))

 hist(data, breaks=x, probability=T, col=rgb(1,0,0,0.5))
}
```

```
Error in cut.default(data, breaks = seq(0, n.base, length.out = x + 1),  :
  'x' must be numeric
```

Hide

```
#chi-squared
data = read.csv("hcmv-263hxkx-1qhtfgz.txt", header=T)
array.data = data.matrix(data)
n.base <- 229354 #number of DNA sequence bases
n.site <- 296 #296 palindrome sites
gene <- seq(1,n.base)
reg.split <- function(n.region, gene, site){ #function to split regions into n regions
  count.int <- table(cut(site, breaks = seq(1, length(gene), length.out=n.region+1), inc
lude.lower=TRUE))
  count.vector <- as.vector(count.int)
  count.tab <-table(count.vector)
  return(count.tab)
}
n.region <- 50 # n=50 regions here
reg.split(n.region, gene, site.random) #generate a table for the dataset
```

```
count.vector
 1  2  3  4  5  6  7  8  9 10 12
 1  2  3  6 14  7  4  7  2  2  2
```

Hide

```
chisqtable <- function(n.region, site, n.base){ #chi-squared table for the dataset
  n <- length(site)
  lambda.est <- n/n.region #estimate for lambda
  count.int <- table(cut(site, breaks = seq(1, length(gene), length.out=n.region+1), inc
lude.lowest=TRUE)) #divide into n.region number of non-overlapping intervals
  count.vector <- as.vector(count.int)  #get the count levels range
  count.range <- max(count.vector) - min(count.vector) +1

  table <- matrix(rep(NA, count.range*3), count.range, 3) #create a table
  for(i in 1:count.range){
    offset <-min(count.vector) - 1
    table[i, 1] <- i + offset #first column is the count level
    table[i, 2] <- sum(count.vector == i + offset) #2nd column is the observed 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))*n.region
        else
          table[i, 3] <- (ppois(i + offset, lambda.est) - ppois(i + offset - 1, lambda.e
st))*n.region
  }

  return(table)
}
site.random.table <- chisqtable(n.region, site.random, n.base)
site.random.table
```

```
      [,1] [,2]      [,3]
 [1,]    1    1 0.9290793
 [2,]    2    2 2.3526650
 [3,]    3    3 4.6425922
 [4,]    4    6 6.8710365
 [5,]    5   14 8.1353072
 [6,]    6    7 8.0268365
 [7,]    7    4 6.7884103
 [8,]    8    7 5.0234236
 [9,]    9    2 3.3042964
[10,]   10    2 1.9561435
[11,]   11    0 1.0527609
[12,]   12    2 0.9174487
```

Hide

```
install.packages("gplots")
```

```
also installing the dependencies ‘gtools’, ‘gdata’

trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.4/gtools_3.8.1.tgz'
Content type 'application/x-gzip' length 262688 bytes (256 KB)
==================================================
downloaded 256 KB

trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.4/gdata_2.18.0.tgz'
Content type 'application/x-gzip' length 1141723 bytes (1.1 MB)
==================================================
downloaded 1.1 MB

trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.4/gplots_3.0.1.1.tg
z'
Content type 'application/x-gzip' length 509475 bytes (497 KB)
==================================================
downloaded 497 KB
```

```


The downloaded binary packages are in
    /var/folders/xy/341ywwj15pq82lydhqf89xr80000gn/T//RtmpWrrJrc/downloaded_packages
```

Hide

```
#Monte Carlo Practice
set.seed(12)
n.site
```

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

Case Study 3: Patterns in DNA

Hide

```
# CMD + SHIFT + K
# CMD + OPTION + I
# CMD + SHIFT + ENTER
```

Hide

```
library(lattice)
library(purrr)
```

1 Locations

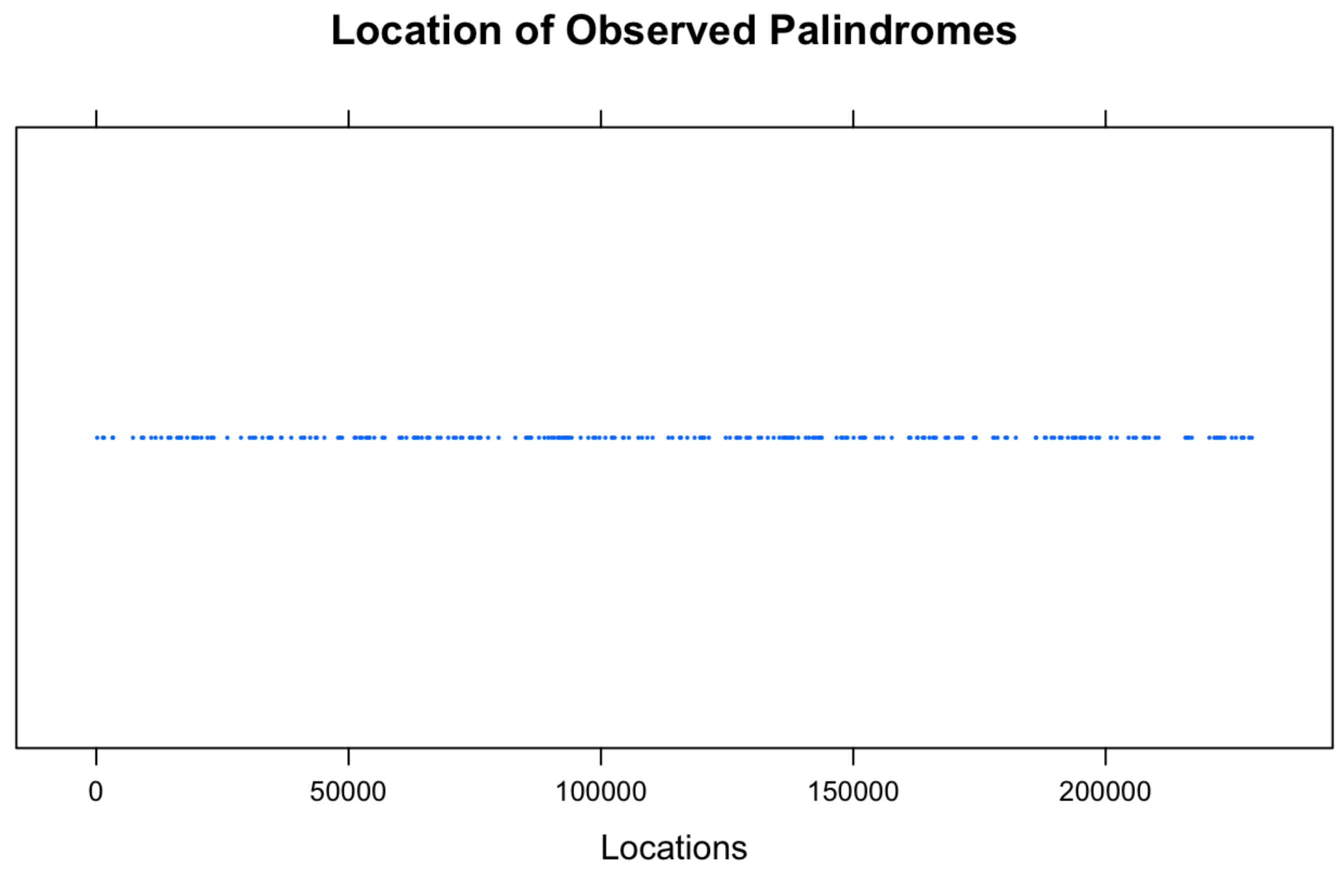
1.1 Hello

Hide

```
DNA.len <- 229354
# Import the data
data <- read.table("data.txt", header=TRUE)
locs = data$location
locs.len = length(obs_locs)
locs.min = min(obs_locs)
locs.max = max(obs_locs)
# Build the generated palindorme locations
r_locs <- round(runif(locs.len, 0, DNA.len))
```

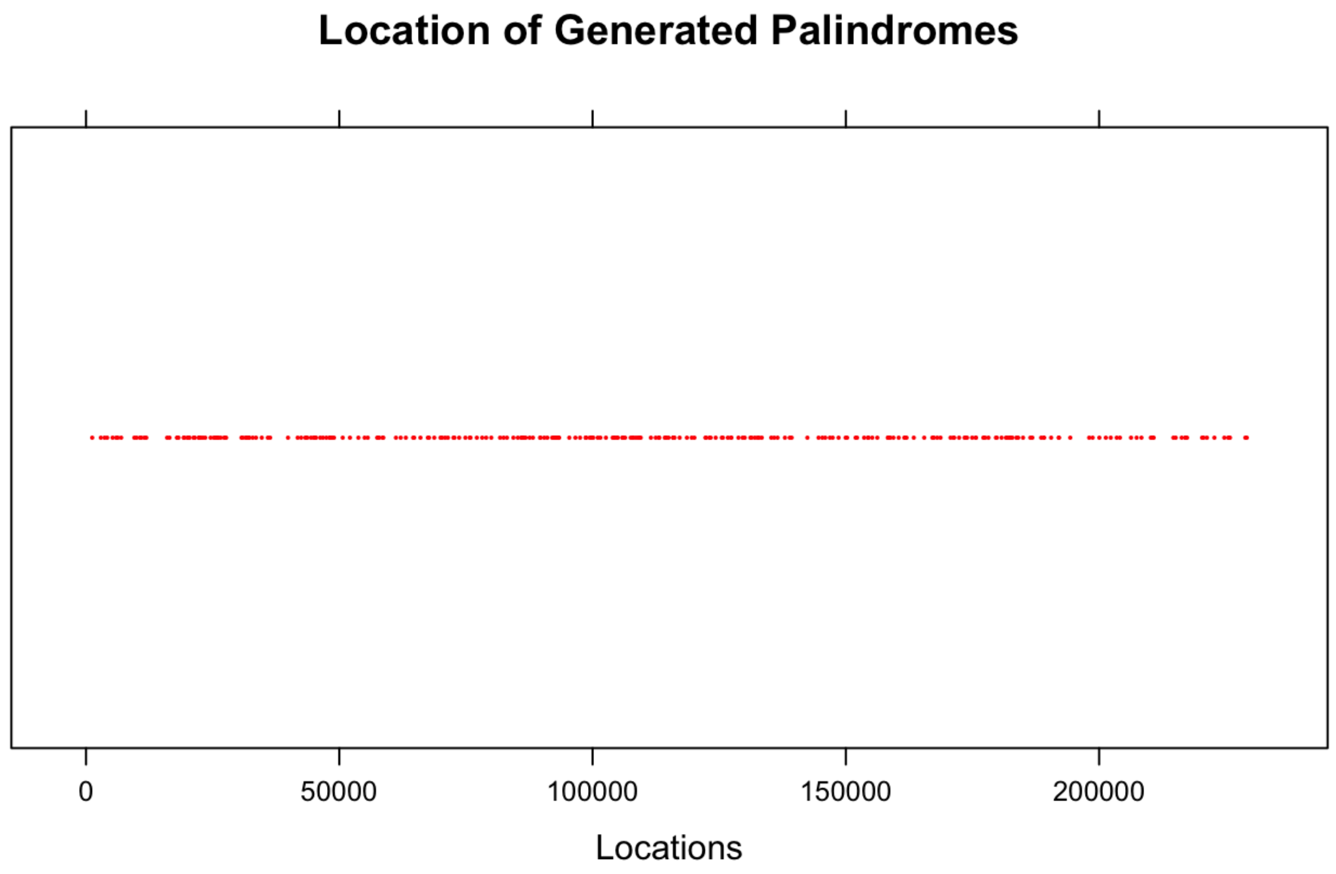
Hide

```
# Plotting the observed locations
stripplot(locs, main='Location of Observed Palindromes', xlab='Locations', pch=16, cex=0.2
5) #one-dimensional scatter plot
```



Hide

```
stripplot(r_locs, main='Location of Generated Palindromes', xlab='Locations', pch=16, cex=
0.25, col='red') #one-dimensional scatter plot
```

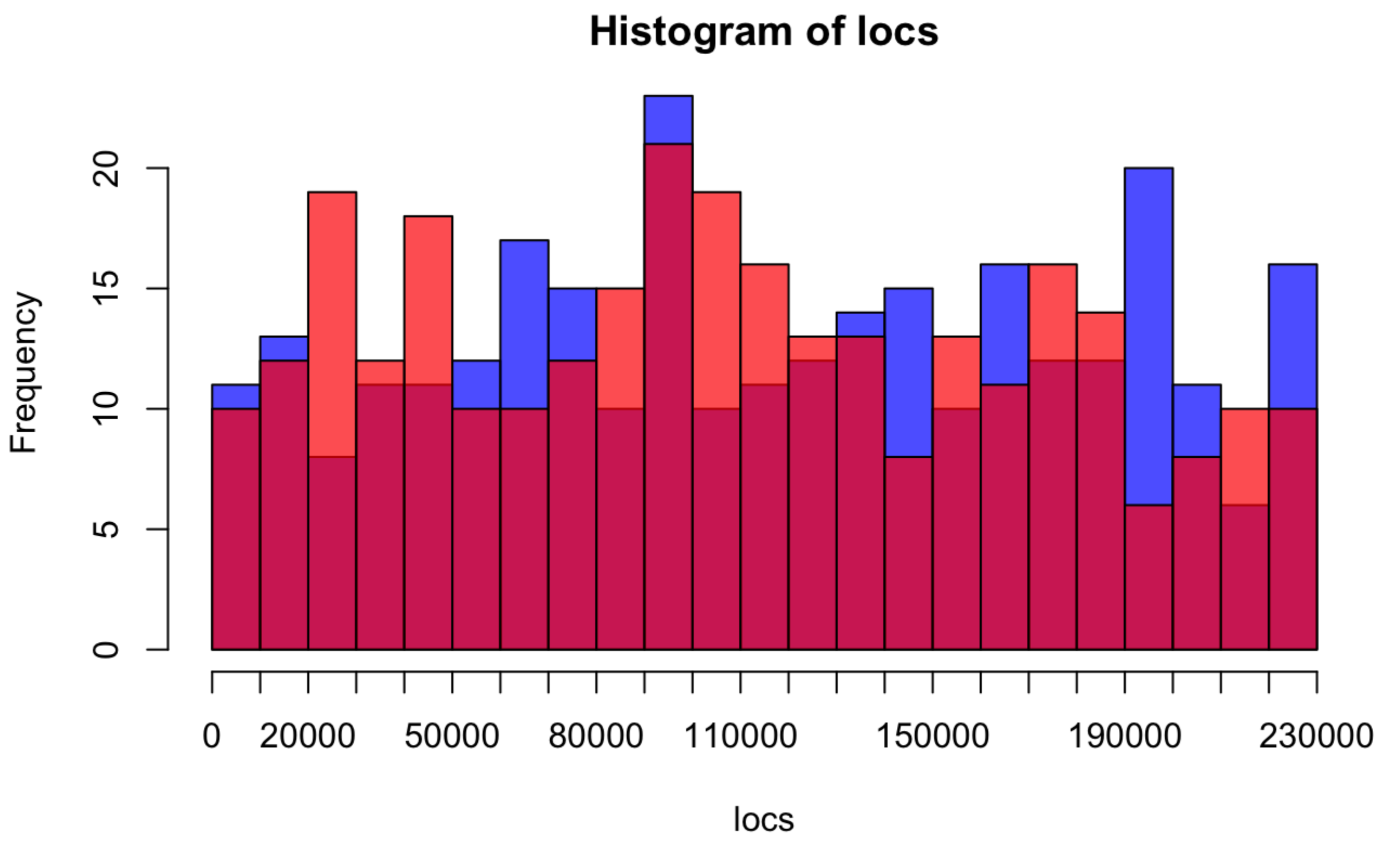


Hide

```
hist(locs, breaks = 30, xaxt='n', col=rgb(0,0,1,0.7), xaxt='n')
hist(r_locs, breaks = 30, col=rgb(1,0,0,0.7), xaxt='n', add=T)
```

Hide

```
axis(1, at=seq(0, DNA.len + 10000, 10000))
```



Hide

```
# Split the data into intervals
indx = 1
intervals.size <- 2050
intervals <- split(locs, cut(locs, seq(locs.min, locs.max, by=intervals.size),
                             include.lowest=TRUE, drop=FALSE))
intervals.num <- length(intervals)
# Count the total number of palindromes in our intervals
intervals.tot_pals = 0
for (intv in intervals) {
  intervals.tot_pals = intervals.tot_pals + length(intv)
}
lambda.est = intervals.tot_pals/intervals.num
# Print metrics
paste("Interval Size:", intervals.size)
```

```
[1] "Interval Size: 2050"
```

Hide

```
paste("Number of intervals:", intervals.num)
```

```
[1] "Number of intervals: 111"
```

Hide

```
paste("Total Palindromes (within intervals):", intervals.tot_pals)
```

```
[1] "Total Palindromes (within intervals): 294"
```

Hide

```
paste("Lambda Est:", lambda.est)
```

```
[1] "Lambda Est: 2.64864864864865"
```

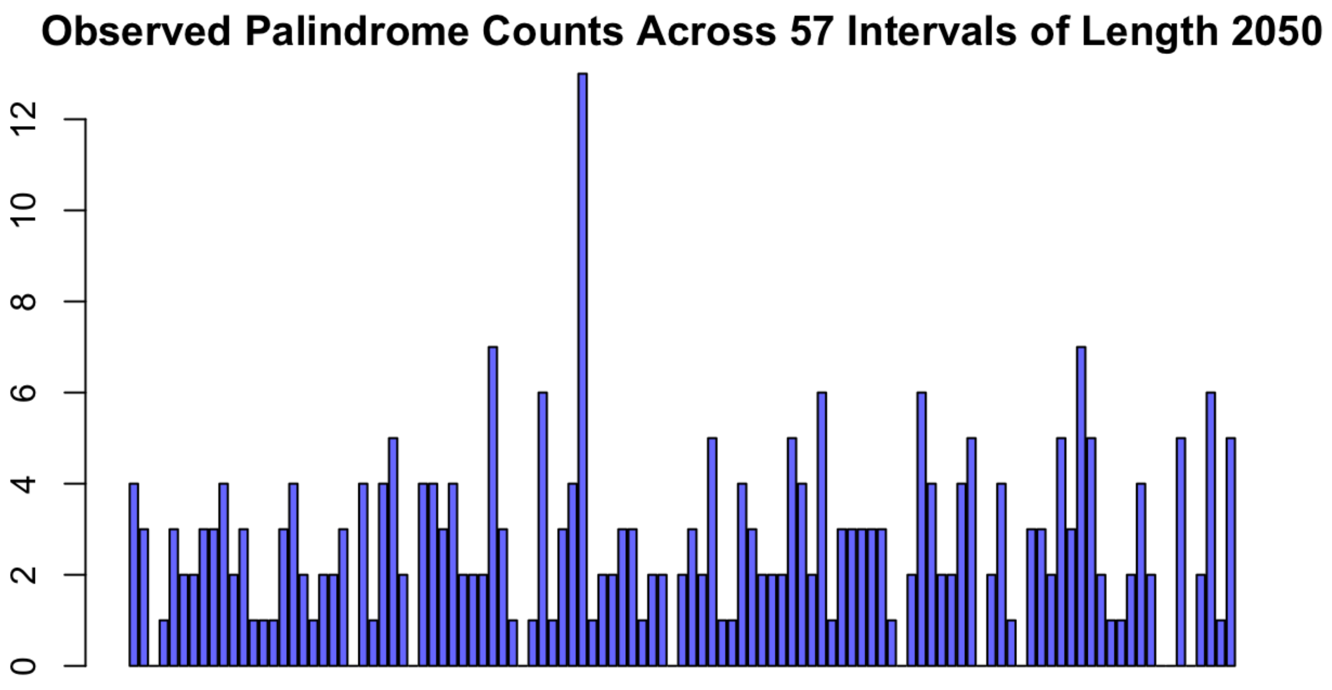
Hide

```
k_plus = 6
# Calculated the expected palindrome counts for {0, 1, ... 8, 9+}
pal.counts_est <- numeric(k_plus)
for (i in 0:k_plus) {
  if (i != k_plus) {
    pal.counts_est[i + 1] = (ppois(i, lambda.est, lower=T) -
      ppois(i - 1, lambda.est, lower=T)) * intervals.num
  } else {
    pal.counts_est[i + 1] = ppois(k_plus - 1, lambda.est, lower=F) * intervals.num
  }
}
paste("Pal Count Estimates:", toString(pal.counts_est))
```

```
[1] "Pal Count Estimates: 7.85288949548925, 20.7995451502148, 27.5453435773115, 24.3193123
475362, 16.1033284463416, 8.5304118256296, 5.84916915747714"
```

Hide

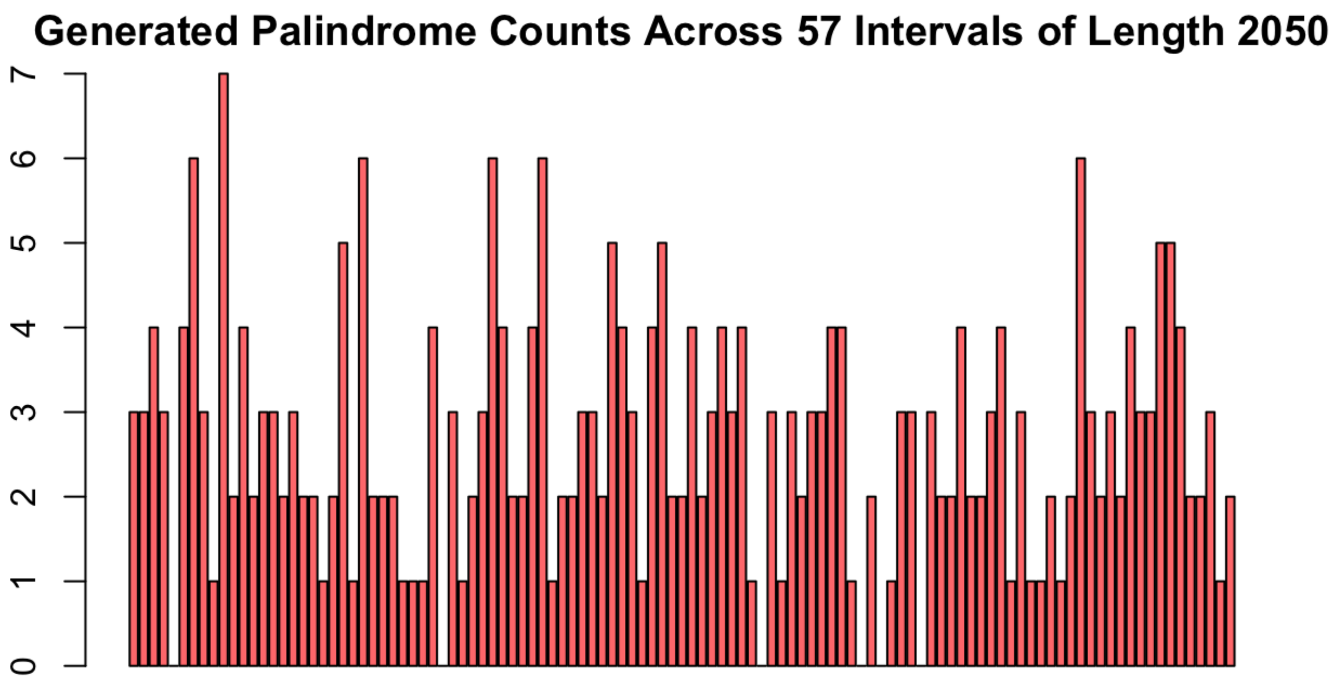
```
# Get raw counts for each k palindromes
pal.counts <- integer(intervals.num)
i = 1
for (p in intervals) {
  pal.counts[i] = length(p)
  i = i + 1
}
title = paste('Observed Palindrome Counts Across', intervals.len, 'Intervals of Length', i
ntervals.size)
barplot(pal.counts, col=rgb(0,0,1, 0.6), main=title, xlab=xlab)
```



Palindrome Counts Across 57 Intervals of Length 2050

Hide

```
# Get raw counts for each k palindromes
r_locs <- round(runif(locs.len, 0, DNA.len))
r_intervals <- split(r_locs, cut(r_locs, seq(min(r_locs), max(r_locs), by=intervals.size),
                             include.lowest=TRUE, drop=FALSE))
r_pal.counts <- integer(intervals.num)
i = 1
for (p in r_intervals) {
  r_pal.counts[i] = length(p)
  i = i + 1
}
title = paste('Generated Palindrome Counts Across', intervals.len, 'Intervals of Length',
 intervals.size)
barplot(r_pal.counts, col=rgb(1,0,0, 0.6), main=title, xlab=xlab)
```



Palindrome Counts Across 57 Intervals of Length 2050

Hide

```
pal.counter <- integer(k_plus + 1)
# Build observed palindrome counts for contigency table
for (count in pal.counts) {
  if ( count >= k_plus ) {
    pal.counter[k_plus + 1] = pal.counter[k_plus + 1] + 1
  } else {
    pal.counter[count + 1] = pal.counter[count + 1] + 1
  }
}
print("Observed Palindrome Counts")
```

```
[1] "Observed Palindrome Counts"
```

Hide

```
print(pal.counter)
```

```
[1] 11 19 29 22 15  8  7
```

Hide

```
print("Contigency Table")
```

```
[1] "Contigency Table"
```

Hide

```
for (i in seq_along(pal.counter) ) {
  plus = "  "
  if(i == k_plus + 1) {
    plus = "+"
  }
  str <- paste(toString(i - 1), plus, ": ", toString(pal.counter[i]),
               "     ,", toString(pal.counts_est[i]))
  print(str)
}
```

```
[1] "0   :  11      , 7.85288949548925"
[1] "1   :  19      , 20.7995451502148"
[1] "2   :  29      , 27.5453435773115"
[1] "3   :  22      , 24.3193123475362"
[1] "4   :  15      , 16.1033284463416"
[1] "5   :  8      , 8.5304118256296"
[1] "6 + :  7      , 5.84916915747714"
```

Hide

```
# Manually combine rows
n <- k_plus - 1 # Number of rows we'll be sizing down to
contigency_obs = numeric(n)
contigency_exp = numeric(n)
num_comb <- 1
for (i in 1:length(pal.counter)) {
  if (i <= num_comb) {
    contigency_obs[1] <- contigency_obs[1] + pal.counter[i]
    contigency_exp[1] <- contigency_exp[1] + pal.counts_est[i]
  } else {
    contigency_obs[i - num_comb + 1] <- pal.counter[i]
    contigency_exp[i - num_comb + 1] <- pal.counts_est[i]
  }
}
# Truncated Contigency Table
pal_cont_table <- data.frame("Obs Pal Counts"=contigency_obs, "Exp Pal Counts"=contigency_
exp)
pal_cont_table
```

Obs.Pal.Counts <dbl>	Exp.Pal.Counts <dbl>
11	7.852889
19	20.799545
29	27.545344

Obs.Pal.Counts <dbl>	Exp.Pal.Counts <dbl>
22	24.319312
15	16.103328
8	8.530412
7	5.849169

7 rows

1.3 Chi Squarted Function

Hide

```
# Compute the chi-squared
#
# Args:
#     obs: List of observed values
#     exp: List of expected values
chi_sqrd <- function(obs, exp, k) {
  n <- length(obs)
  x <- 0
  for(i in 1:length(obs)) {
    x <- x + ((obs[i] - exp[i])^2)/(exp[i])
  }
  #x <- sum( ((obs - exp)^2)/exp)

  print(n - k)
  return (c(x, pchisq(x, n - k, lower=F)))
}
```

Hide

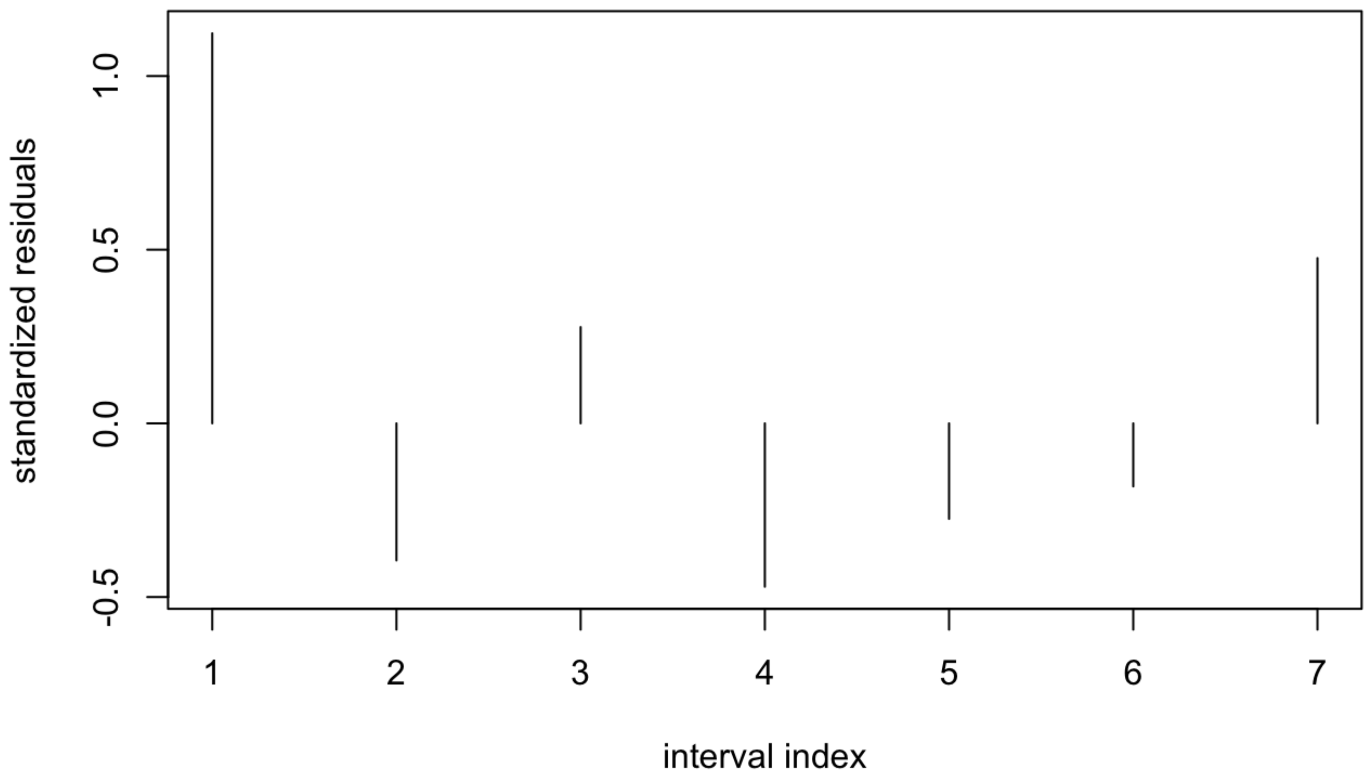
```
sample_obs <- c(7, 8, 10, 9, 8, 5, 4, 6)
sample_exp <- c(6.4, 7.5, 9.7, 10, 8.6, 6.3, 4.1, 4.5)
print(chi_sqrd(contigency_obs, contigency_exp, 2))
```

```
[1] 5
[1] 2.0499380 0.8421929
```

1.6 Residuals of Palindrome Counts

Hide

```
Residuals <- (contigency_obs - contigency_exp) / sqrt(contigency_exp)
plot(Residuals, type = 'h', ylab = "standardized residuals", xlab = "interval index")
```



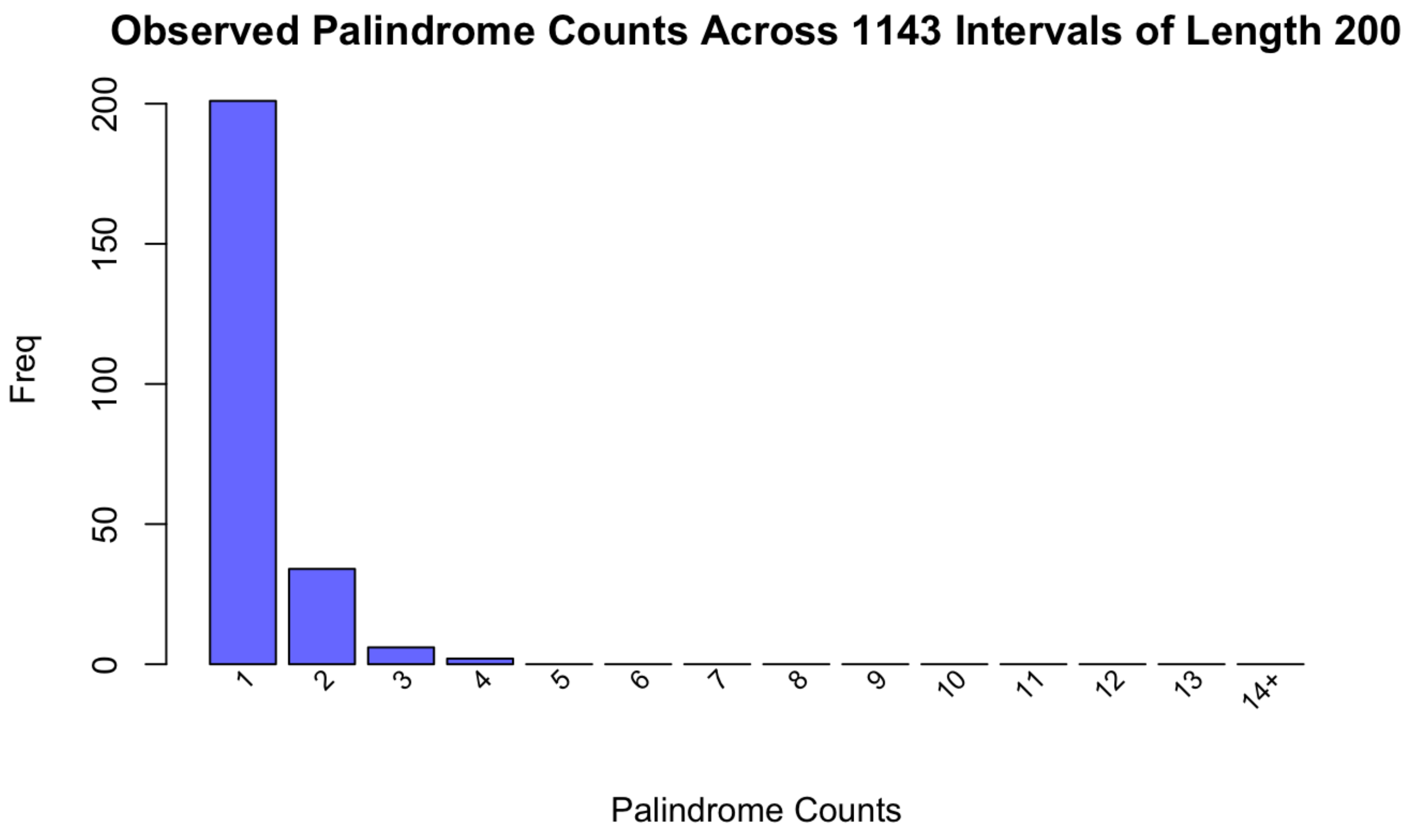
Hide

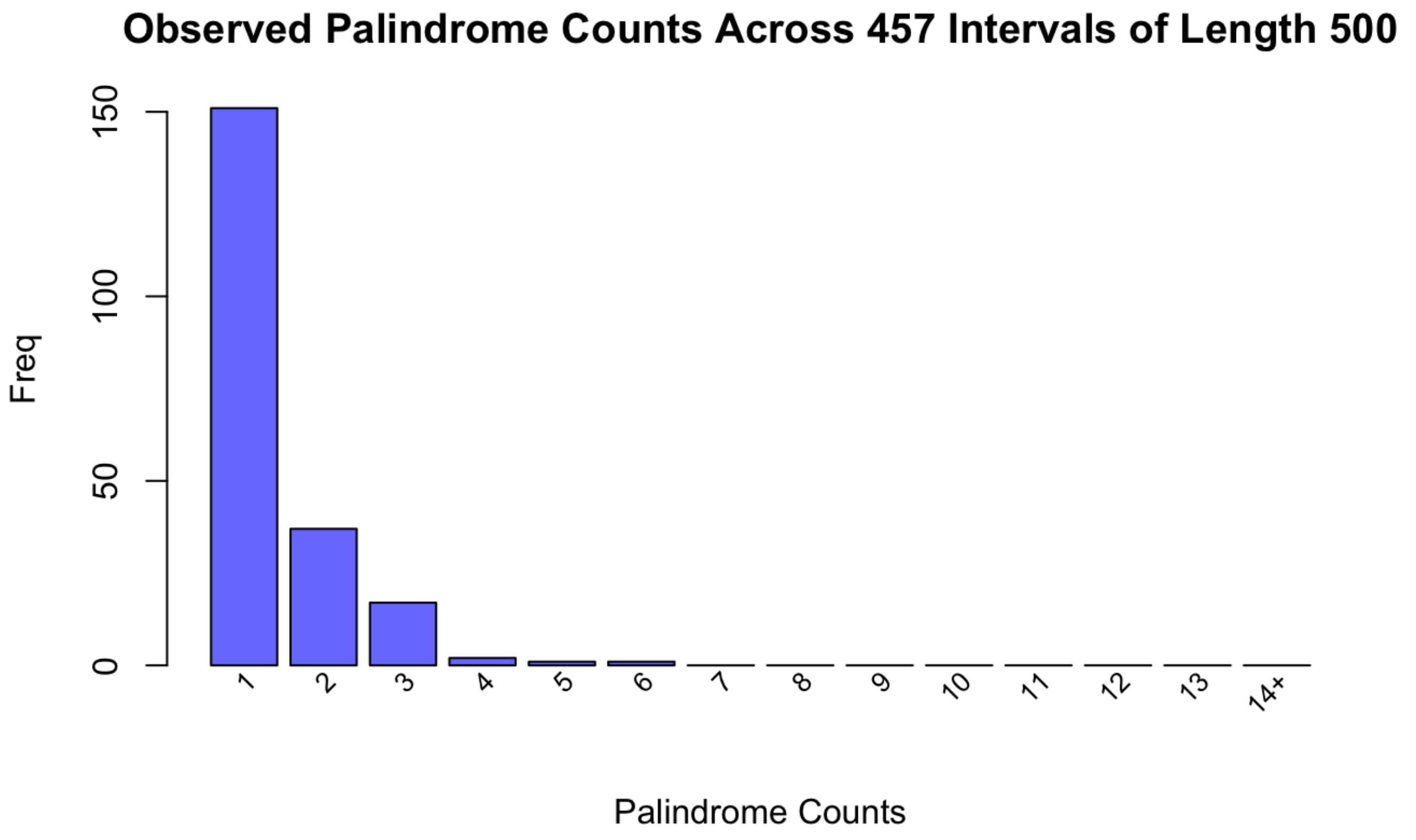
```
r_locs = round(runif(num_locs, min_loc, max_loc))
```

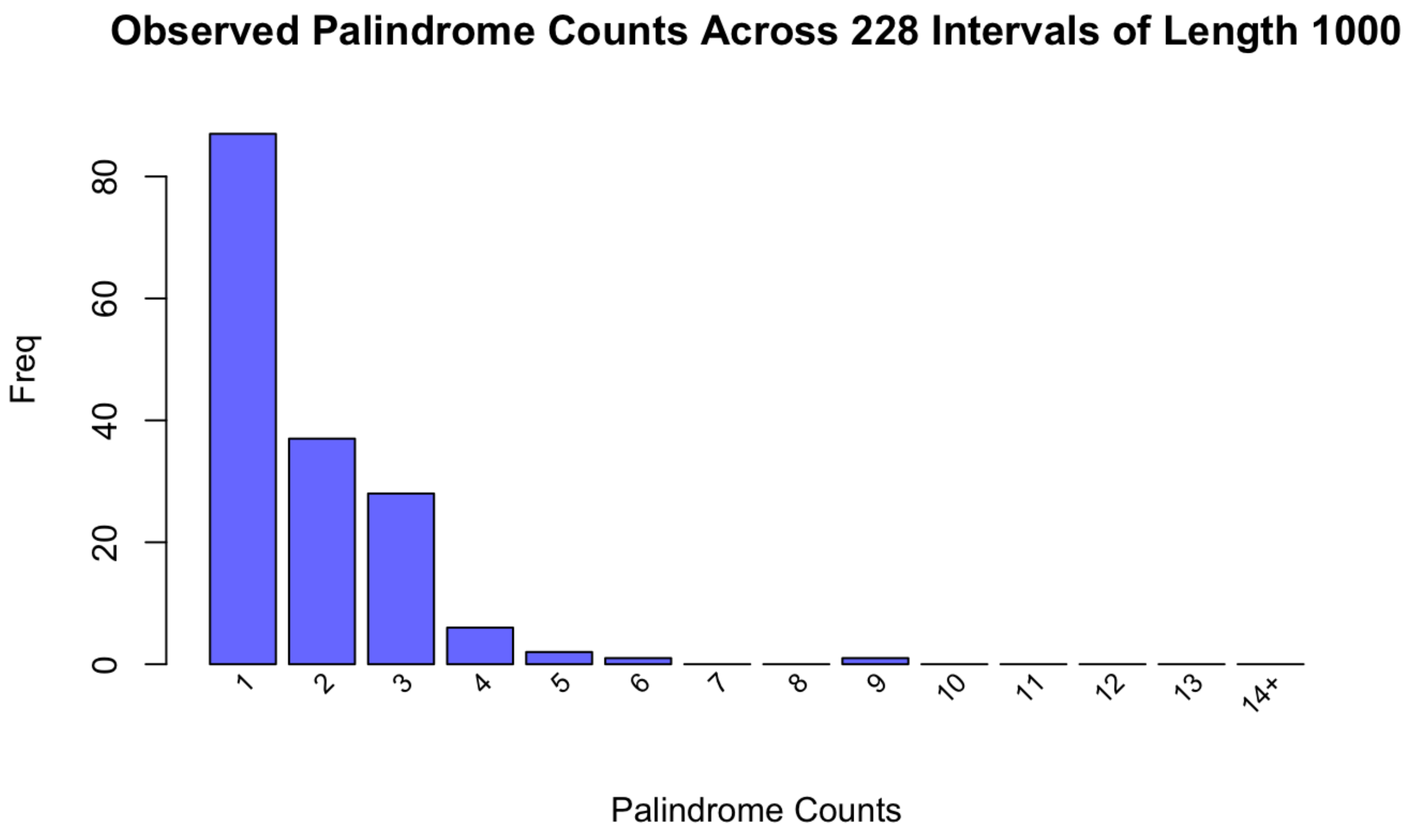
Classifying Regions According to Counts

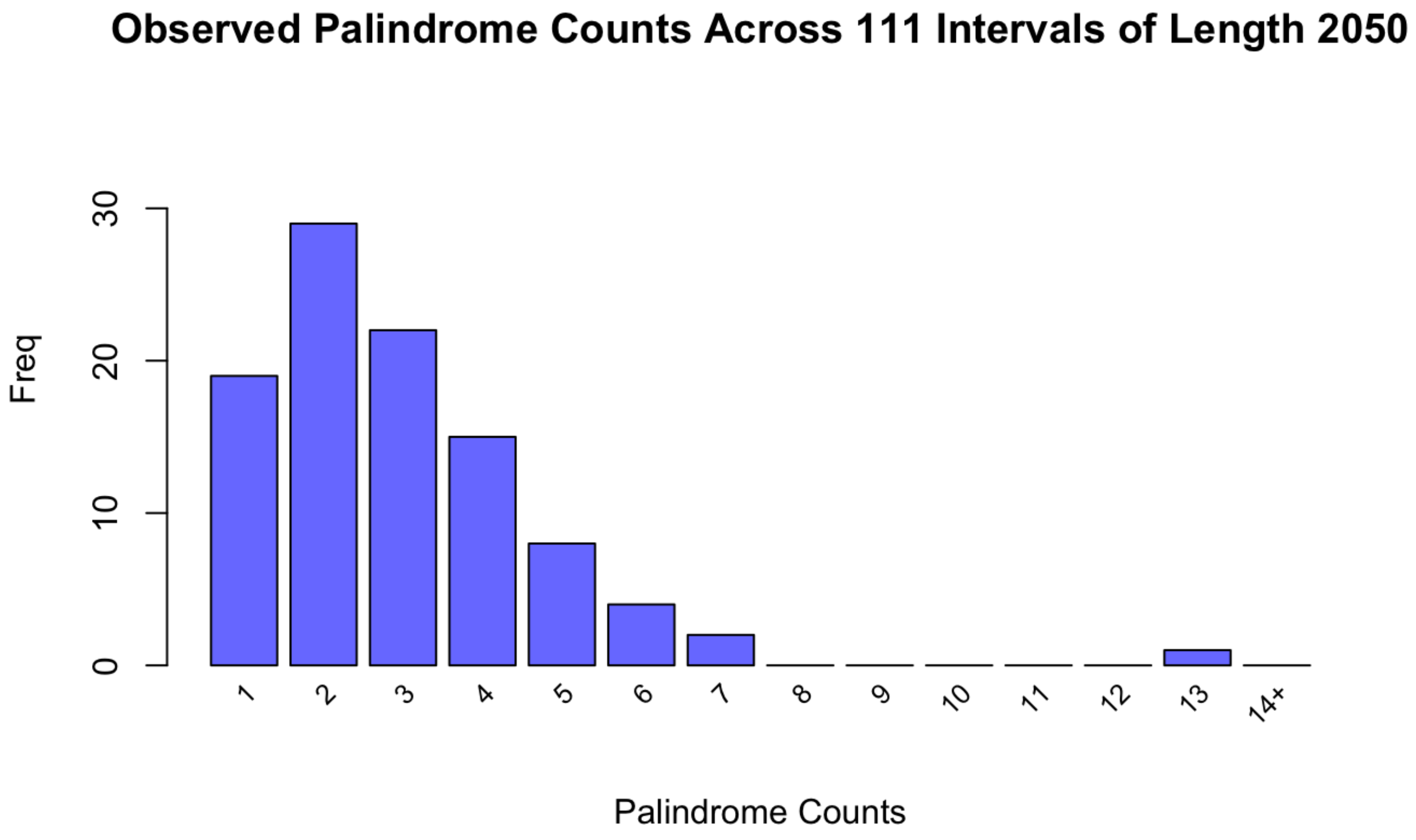
Hide

```
tmp_i_size <- c(200, 500, 1000, 2050)
r_locs <- round(runif(locs.len, 0, DNA.len))
tmp.counts <- integer()
for (i_size in tmp_i_size) {
  tmp.counts <- integer()
  tmp <- split(locs, cut(locs, seq(min(locs), max(locs), by=i_size),
                             include.lowest=TRUE, drop=FALSE))

  tmp.len <- length(tmp)
  tmp.counts <- integer(14)

  i = 1
  for (p in tmp) {
    if(length(p) >= 14 ){
      tmp.counts[14] = tmp.counts[14] + 1
    } else {
      tmp.counts[length(p)] = tmp.counts[length(p)] + 1
    }
    i = i + 1
  }

  title = paste('Observed Palindrome Counts Across', tmp.len, 'Intervals of Length', i_siz
e)
  bp <- barplot(tmp.counts, col=rgb(0,0,1, 0.6), main=title, ylim=c(0,max(tmp.counts) + 10
), xaxt='n', xlab="Palindrome Counts", ylab="Freq")

  labels <- (1:14)
  labels[14] <- "14+"
  text(x=bp[,1], y=-1, adj=c(1, 1), labels, cex=0.8, srt=45, xpd=TRUE)
}
```


Observed Palindrome Counts Across 1143 Intervals of Length 200
Freq
0
50
100
150
200
1
2
3
4
5
6
7
8
9
10
11
12
13
14+
Palindrome Counts


Observed Palindrome Counts Across 457 Intervals of Length 500
Freq
0
50
100
150
1
2
3
4
5
6
7
8
9
10
11
12
13
14+
Palindrome Counts





2 Spacing Between Palindromes

2.1 Compute and Generate Spacings

Hide

```
# Calculate observed spacings
pair_spacings <- diff(locs)
trip_spacings <- numeric(length(pair_spacings) - 1)
for(i in 1:(length(pair_spacings) - 1)) {
  trip_spacings[i] <- pair_spacings[i] + pair_spacings[i+1]
}
# Compute lambda estimate
exp_lambda.est <- 1 / mean(pair_spacings)
# Generate spacings
r_pair_spacings <- rexp(length(pair_spacings), exp_lambda.est)
r_trip_spacings <- rgamma(length(trip_spacings), 2, exp_lambda.est)
```

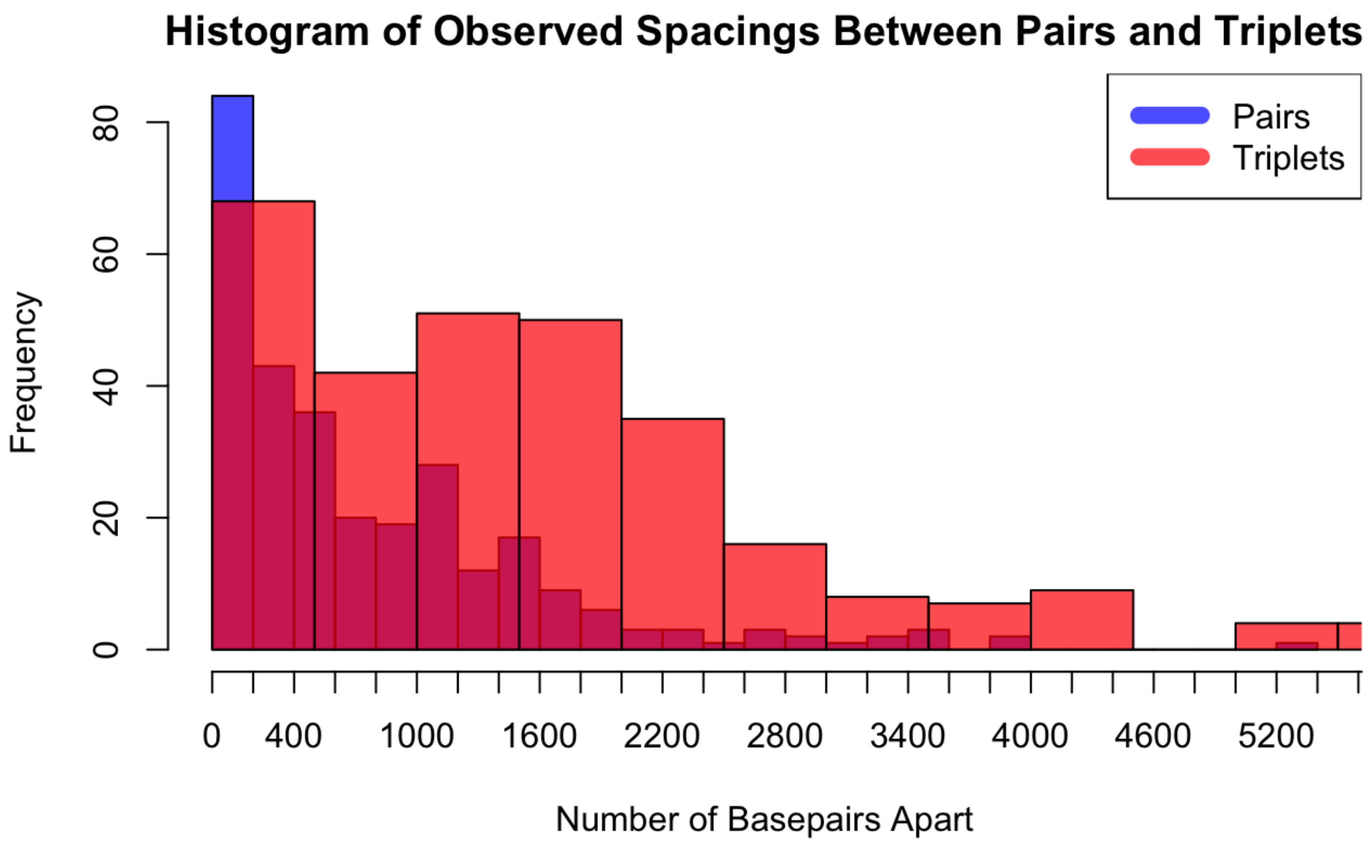
2.2 Plot Spacings Between Palindromes

Hide

```
# Histogram of Observed Spacings
hist(pair_spacings, breaks = 30, xaxt='n',
     main="Histogram of Observed Spacings Between Pairs and Triplets",
     col=rgb(0,0,1,0.7), xlab ="Number of Basepairs Apart")
hist(trip_spacings, col=rgb(1,0,0,0.7),  xaxt='n', add=T)
```

Hide

```
axis(1, seq(0, max(trip_spacings), 200))
legend("topright", c("Pairs", "Triplets"), col=c(rgb(0,0,1,0.7), rgb(1,0,0,0.7)), lwd=8)
```

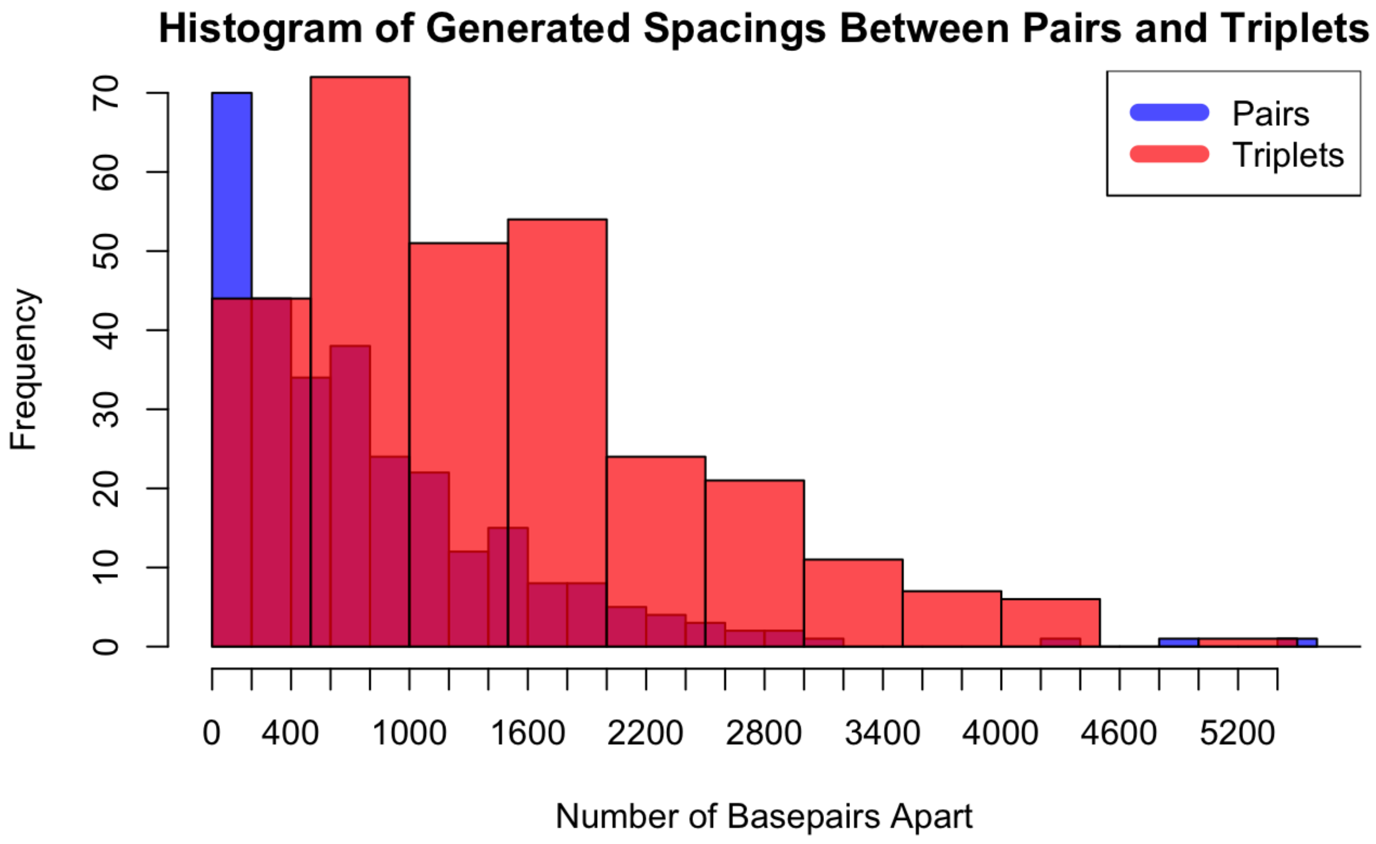


Hide

```
# Histogram of Generated Spacings
hist(r_pair_spacings, breaks = 30, xaxt='n',
     main="Histogram of Generated Spacings Between Pairs and Triplets",
     col=rgb(0,0,1,0.7), xlab ="Number of Basepairs Apart", xaxt='n')
hist(r_trip_spacings, col=rgb(1,0,0,0.7), add=T)
```

Hide

```
axis(1, seq(0, max(r_pair_spacings), 200))
legend("topright", c("Pairs", "Triplets"), col=c(rgb(0,0,1,0.7), rgb(1,0,0,0.7)), lwd=8)
```

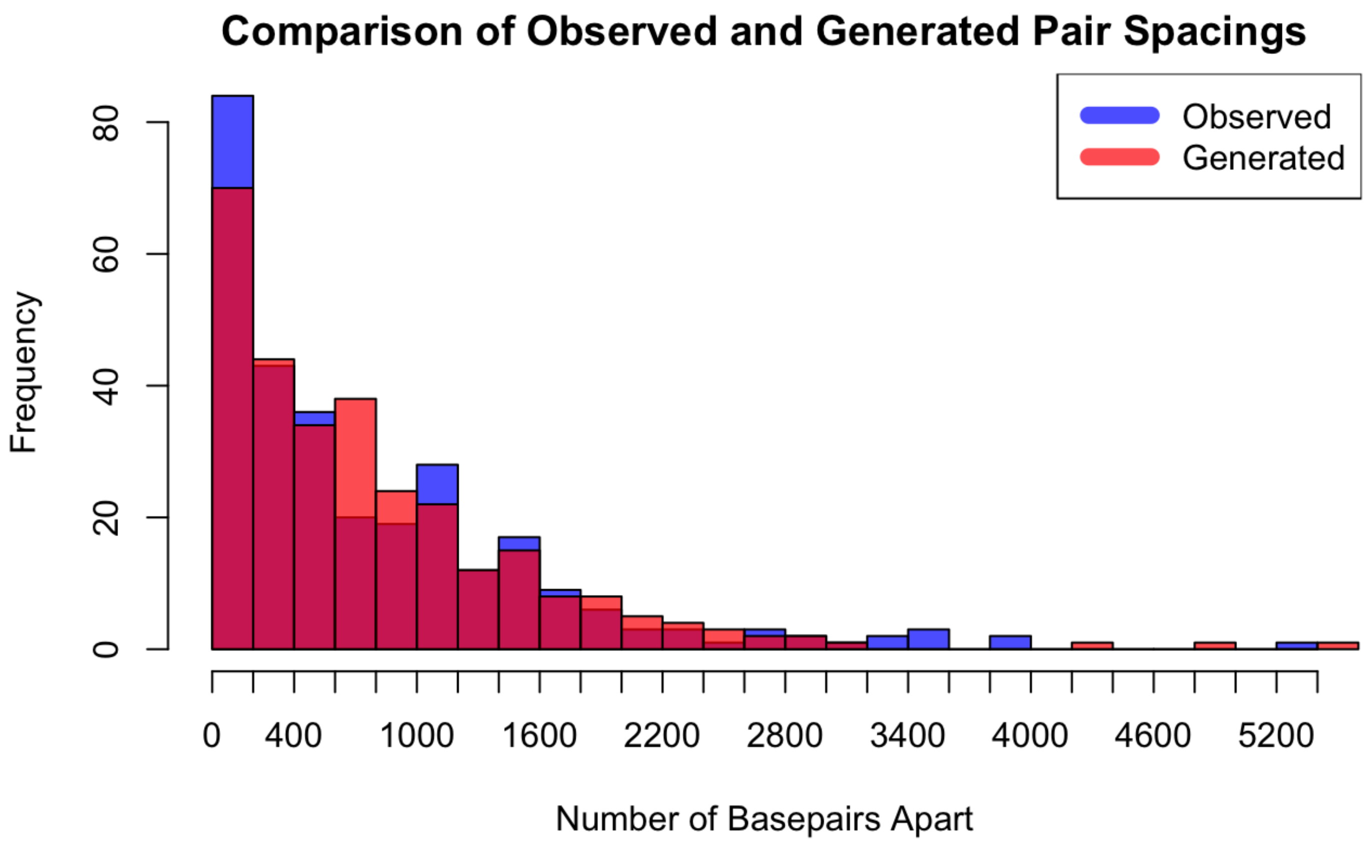


Hide

```
# Histogram Comparison of Observed and Generated Pair Spacings
hist(pair_spacings, breaks = 30, xaxt='n',
     main="Comparison of Observed and Generated Pair Spacings",
     col=rgb(0,0,1,0.7), xlab ="Number of Basepairs Apart")
hist(r_pair_spacings, breaks = 30, xaxt='n',
     col=rgb(1,0,0,0.7), add=T)
```

Hide

```
axis(1, seq(0, max(c(pair_spacings, r_pair_spacings)), 200))
legend("topright", c("Observed", "Generated"), col=c(rgb(0,0,1,0.7), rgb(1,0,0,0.7)), lwd=
8)
```



Hide

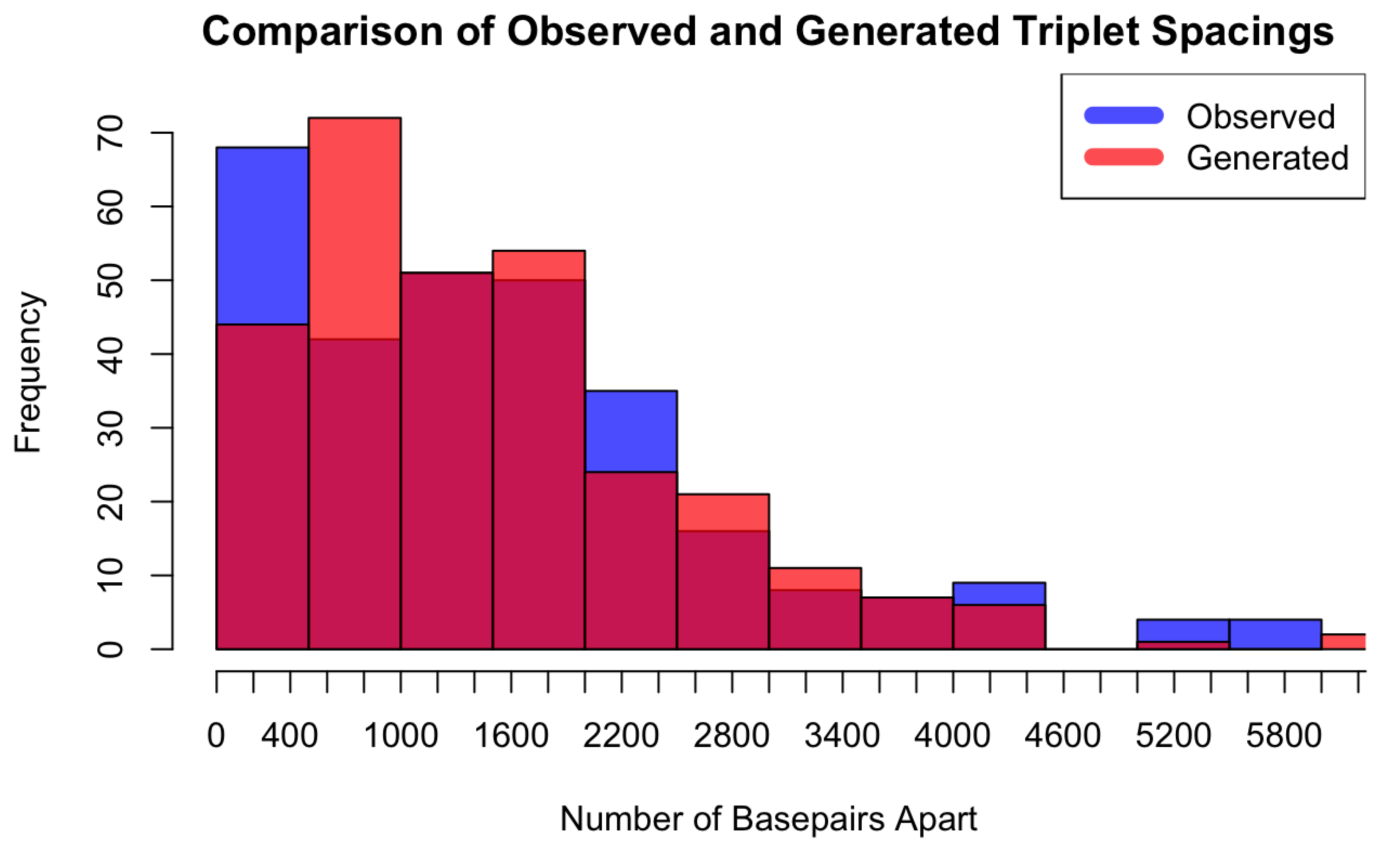
```
# Histogram Comparison of Observed and Generated Triplet Spacings
hist(trip_spacings, main="Comparison of Observed and Generated Triplet Spacings",
     col=rgb(0,0,1,0.7), xlab ="Number of Basepairs Apart", yaxt='n',  xaxt='n', ylim=c(0,
75))
hist(r_trip_spacings, xaxt='n', col=rgb(1,0,0,0.7), xlab ="Number of Basepairs Apart", yax
t='n', add=T )
```

Hide

```
axis(1, seq(0, max(c(trip_spacings, r_trip_spacings)) + 200, 200))
axis(2, seq(0, 70, 10))
```

Hide

```
legend("topright", c("Observed", "Generated"), col=c(rgb(0,0,1,0.7), rgb(1,0,0,0.7)), lwd=
8)
```



2.3 Create Contigency Table for Spacings

Hide

```
sizes <- 75
len <- 750
num_len_above <- sum((pair_spacings > len))
a <- split(pair_spacings, cut(pair_spacings, seq(0, len, by=sizes), include.lowest=TRUE, d
rop=FALSE))
a.count <- c(unlist(map(a, length), use.names=FALSE), num_len_above)
exp_spaces <- numeric(length(a.count))
j = 1
for(i in seq(0, len, by=sizes)) {
  if (i != len) {
    exp_spaces[j] <- pexp(i + sizes, exp_lambda.est, lower=T) - pexp(i, exp_lambda.est, lo
wer=T)
  } else {
    exp_spaces[j] <- pexp(len, exp_lambda.est, lower=F)
  }
  j = j + 1
}
exp_spaces = exp_spaces*length(pair_spacings)
```

Hide

```
space_cont_table <- data.frame("Bucket"=seq(0, len, by=sizes),
                               "Obs counts"=a.count,
                               "Exp Counts"=exp_spaces)
space_cont_table
```

Bucket <dbl>	Obs.counts <int>	Exp.Counts <dbl>
0	46	27.19341
75	26	24.68669
150	20	22.41105
225	20	20.34517
300	12	18.46973
375	14	16.76717
450	12	15.22156
525	13	13.81842
600	11	12.54462
675	5	11.38825

1-10 of 11 rows Previous 1 2 Next

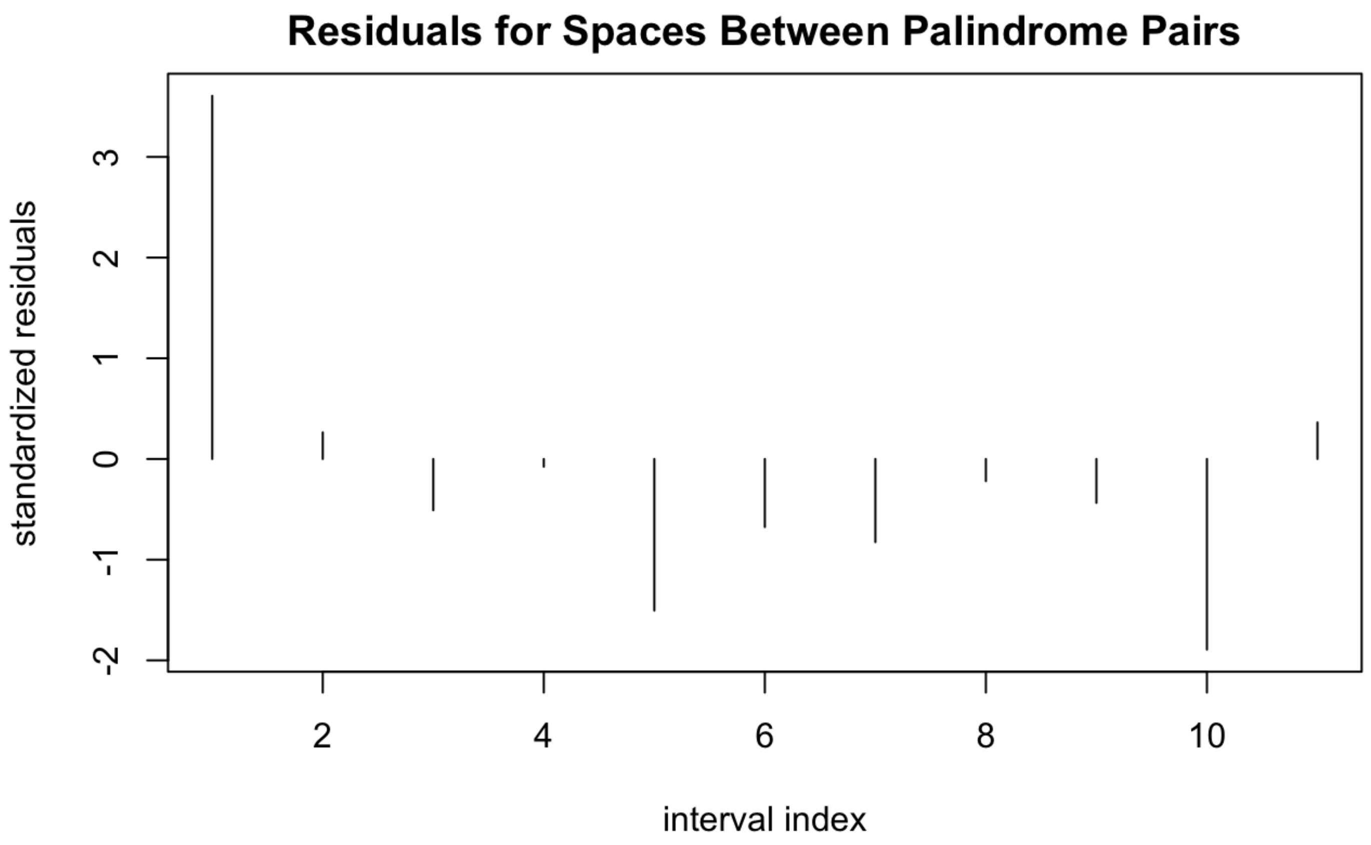
Hide

```
chi_sqrd(a.count, exp_spaces, 2)
```

```
[1] 9
[1] 20.70031691  0.01404961
```

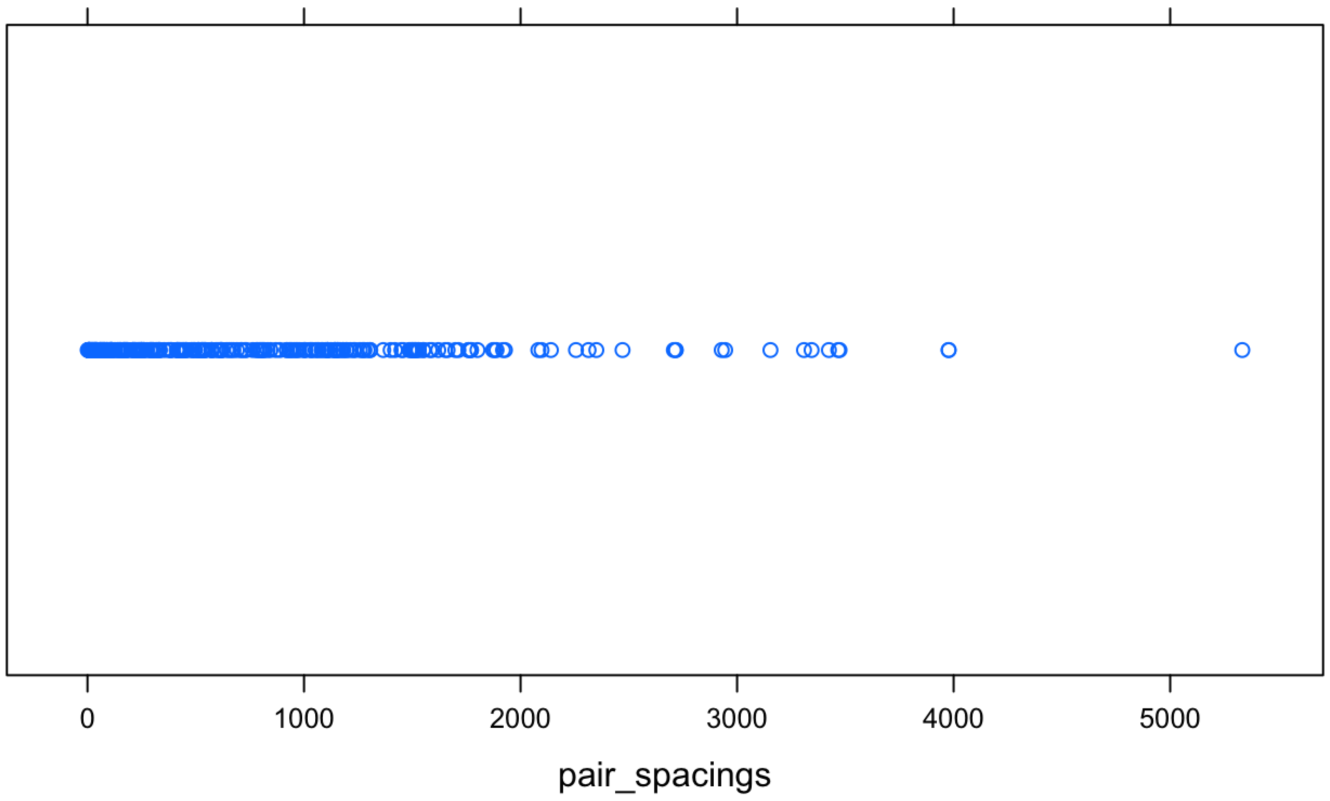
Hide

```
Residuals <- (a.count - exp_spaces) / sqrt(exp_spaces)
plot(Residuals, type = 'h', main="Residuals for Spaces Between Palindrome Pairs", ylab =
"standardized residuals", xlab = "interval index")
```



Hide

```
stripplot(pair_spacings)
```



2.4 Create Contigency Table for Triplet Spacings

Hide

```
sizes <- 150
len <- 1500
num_len_above <- sum((trip_spacings > len))
b <- split(trip_spacings, cut(trip_spacings, seq(0, len, by=sizes), include.lowest=TRUE, d
rop=FALSE))
b.count <- c(unlist(map(b, length), use.names=FALSE), num_len_above)
exp_trip_spaces <- numeric(length(b.count))
j = 1
for(i in seq(0, len, by=sizes)) {
  if (i != len) {
    exp_trip_spaces[j] <- pgamma(i + sizes, 2, exp_lambda.est, lower=T) - pgamma(i,2, exp_
lambda.est, lower=T)
  } else {
    exp_trip_spaces[j] <- pgamma(len, 2, exp_lambda.est, lower=F)
  }
  j = j + 1
}
exp_trip_spaces = exp_trip_spaces * length(trip_spacings)
```

Hide

```
space_cont_table <- data.frame("Bucket"=seq(0, len, by=sizes),
                               "Obs counts"=b.count,
                               "Exp Counts"=exp_trip_spaces)
space_cont_table
```

Bucket <dbl>	Obs.counts <int>	Exp.Counts <dbl>
0	22	4.839239
150	16	12.230090
300	21	16.871690
450	18	19.502445
600	19	20.686071
750	10	20.850202
900	9	20.316815
1050	15	19.326172
1200	15	18.055601
1350	16	16.634200

1-10 of 11 rows Previous 1 2 Next

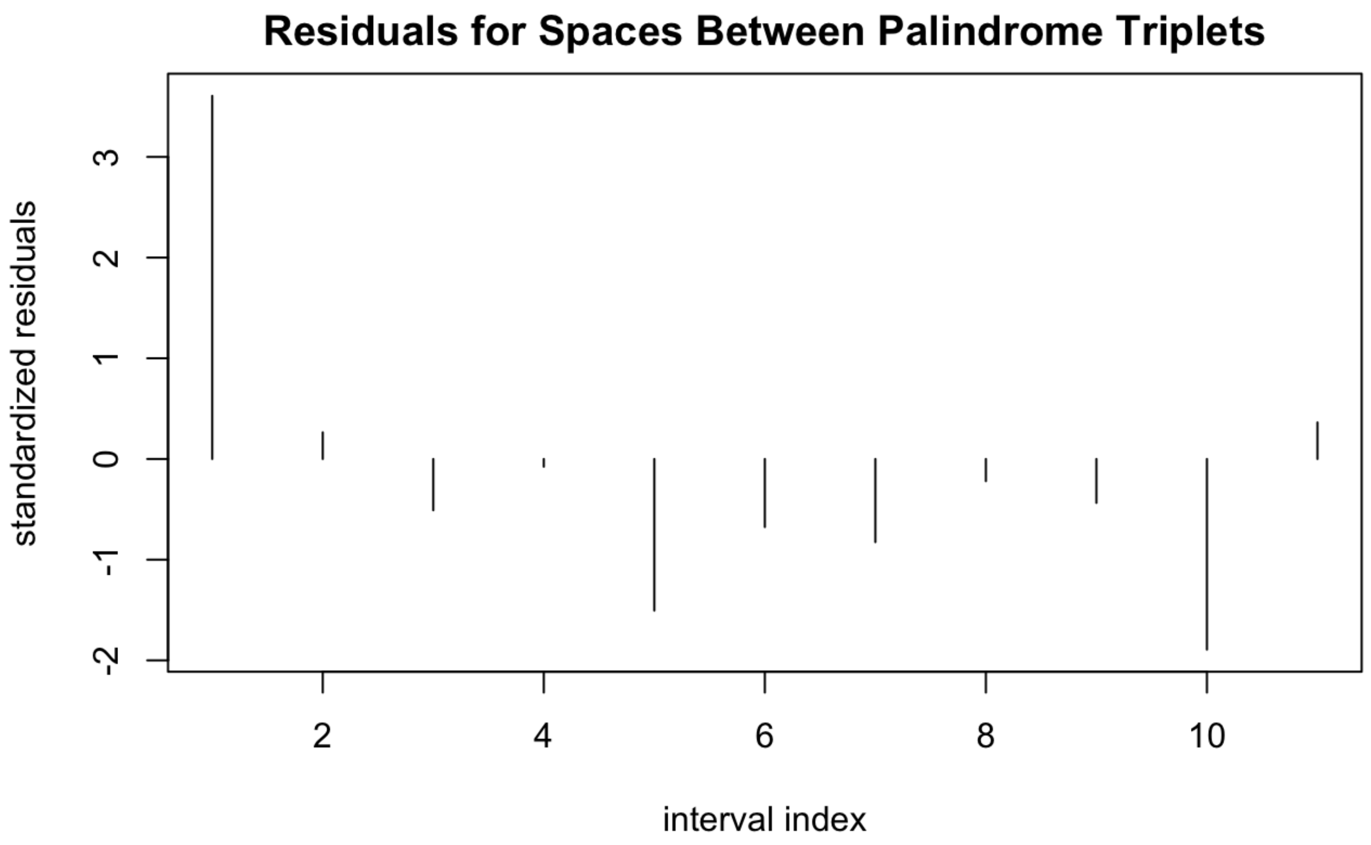
Hide

```
print(chi_sqrd(b.count, (exp_trip_spaces), 2))
```

```
[1] 9
[1] 7.729420e+01 5.560457e-13
```

Hide

```
Residuals <- (a.count - exp_spaces) / sqrt(exp_spaces)
plot(Residuals, type = 'h', main="Residuals for Spaces Between Palindrome Triplets", ylab
 = "standardized residuals", xlab = "interval index")
```



3 Locations and Uniform Distribution

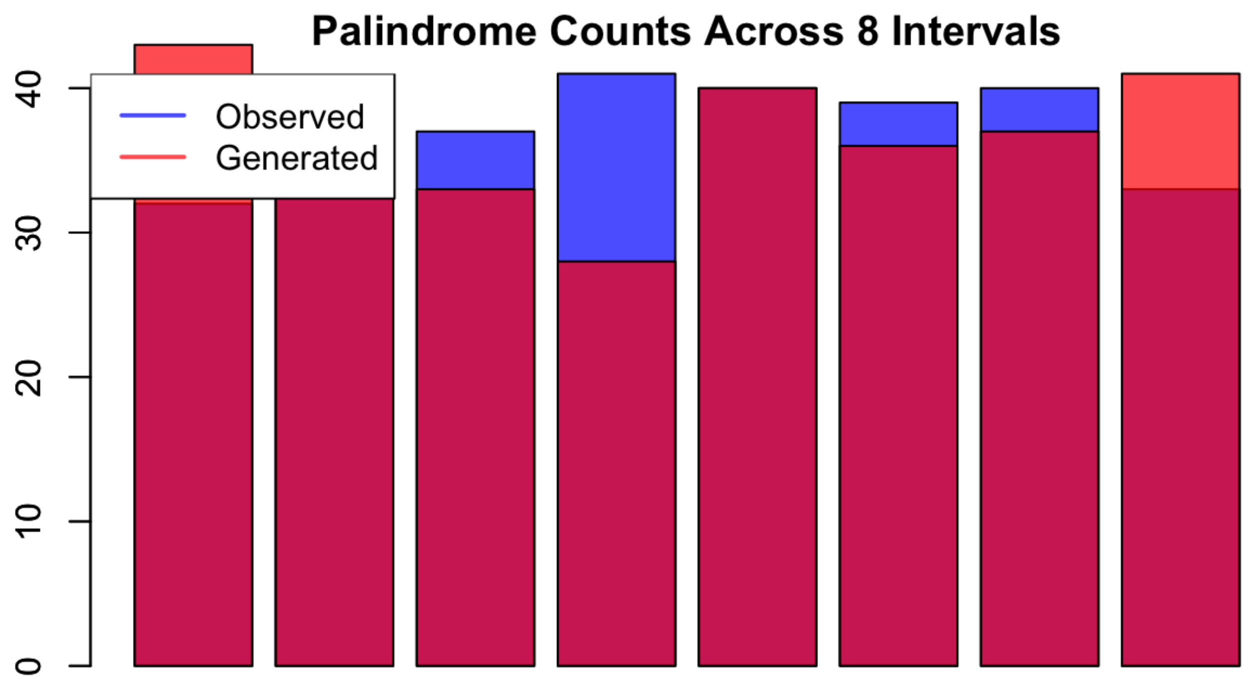
3.1 Compute Observed and Generated Count of Palindromes Across Buckets

Hide

```
num_buckets <- 8
r_locs <- round(runif(locs.len, 0, DNA.len))
split_locs <- split(locs, cut(locs, seq(0, DNA.len, by=(DNA.len/num_buckets)), include.low
est=TRUE, drop=FALSE))
split_locs.count <- unlist(map(split_locs, length), use.names=FALSE)
r_split_locs <- split(r_locs, cut(r_locs, seq(0, DNA.len, by=(DNA.len/num_buckets)), inclu
de.lowest=TRUE, drop=FALSE))
r_split_locs.count <- unlist(map(r_split_locs, length), use.names=FALSE)
barplot(split_locs.count,  main=paste("Palindrome Counts Across",num_buckets, "Intervals"
), col=rgb(0,0,1,0.7))
barplot(r_split_locs.count, col=rgb(1,0,0,0.7), add=T)
```

Hide

```
legend("topleft", c("Observed", "Generated"), col=c(rgb(0,0,1,0.7), rgb(1,0,0,0.7)), lwd=2
)
```



3.2 Contigency Table of Palindrome Locations

Hide

```
exp_counts <- replicate(num_buckets, locs.len/num_buckets)
obs_counts <- split_locs.count
contigency_table <- data.frame("Bucket #"=1:num_buckets, "Obs counts"=obs_counts, "Exp Cou
nts"=exp_counts)
contigency_table
```

Bucket.. <int>	Obs.counts <int>	Exp.Counts <dbl>
1	32	37
2	34	37
3	37	37
4	41	37
5	40	37
6	39	37
7	40	37
8	33	37

8 rows

3.3 P-Value of Chi-Squared Test for Palindrome Location Counts

Hide

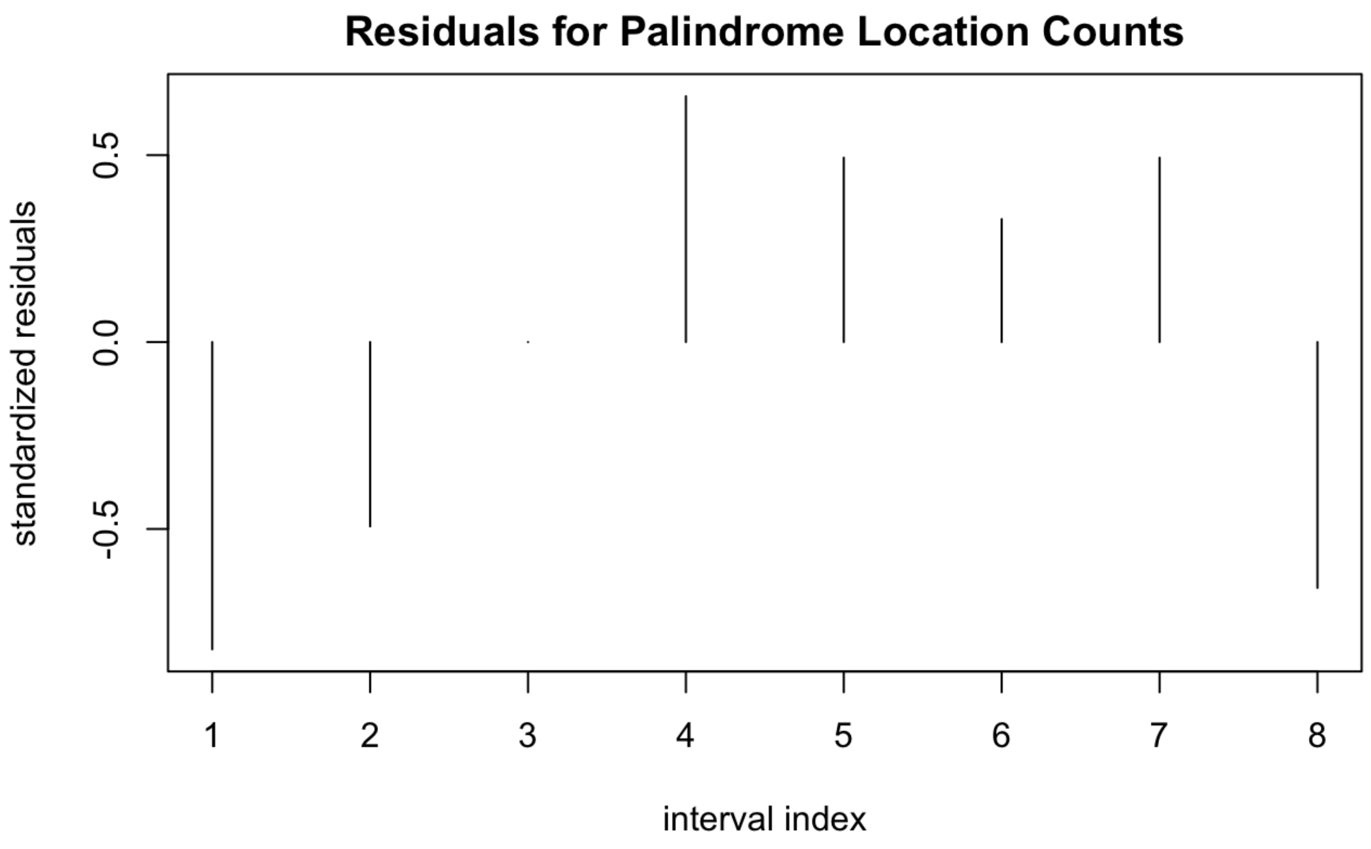
```
print(chi_sqrd(obs_counts, exp_counts, 2))
```

```
[1] 6
[1] 2.378378 0.881823
```

3.4 Residuals of Palindrome Location Counts

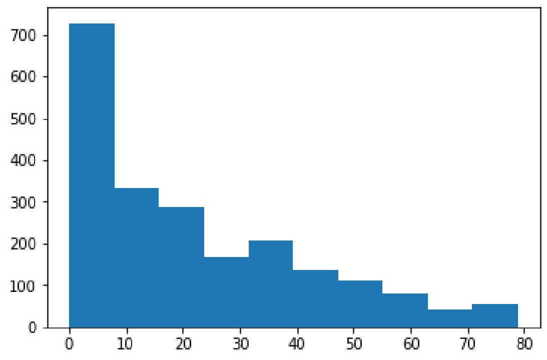
Hide

```
Residuals <- (obs_counts - exp_counts) / sqrt(exp_counts)
plot(Residuals, type = 'h', main="Residuals for Palindrome Location Counts", ylab = "stand
ardized residuals", xlab = "interval index")
```



In [30]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [39]:
```
df = pd.read_excel('RAW_DATA.xls')

age_groups = [18, 35, 50, 80, 2048]

ages_below_80 = np.array(df[df['age_yrs'] != '80+']['age_yrs'])
ages_below_80 = [float(age) for age in ages_below_80]
plt.hist(ages_below_80)
plt.show()
```



In [48]:
```
ages_below_18 = df[(df['age_yrs'] != '80+')]
ages_below_18 = ages_below_18[(ages_below_18['age_yrs'] <= 18)]

ages_18_35 = df[(df['age_yrs'] != '80+')]
ages_18_35 = ages_18_35[(18 < ages_18_35['age_yrs']) & (ages_18_35['age_yrs'] <= 35)]

ages_35_50 = df[(df['age_yrs'] != '80+')]
ages_35_50 = ages_35_50[(35 < ages_35_50['age_yrs']) & (ages_35_50['age_yrs'] <= 50)]

ages_50_80 = df[(df['age_yrs'] != '80+')]
ages_50_80 = ages_50_80[(50 < ages_50_80['age_yrs']) & (ages_50_80['age_yrs'] <= 80)]

ages_80_above = df[(df['age_yrs'] == '80+')]
```

In [156]:
```
def get_contable(df):
    cmvpos = len(df[df['cmvstatus'] == 'positive'])
    cmvneg = len(df[df['cmvstatus'] == 'negative'])
    hivpos = len(df[df['hiv'] == 'positive'])
    hivneg = len(df[df['hiv'] == 'negative'])

    numeric_codes = {
        'positive': 1,
        'negative': 0,
        'unknown': 2
    }

    cmv_codes = [numeric_codes[s] for s in df['cmvstatus']]
    hiv_codes = [numeric_codes[s] for s in df['hiv']]

    if (cmvpos == 0) or (cmvneg == 0) or (hivpos == 0) or (hivneg == 0):
        print("Pearson correlation:", "undefined")
    else:
        print("Pearson correlation:", np.corrcoef(cmv_codes, hiv_codes)[1,0])

    return pd.DataFrame({
        'CMV': [cmvpos, cmvneg],
        'HIV': [hivpos, hivneg]
    }, index=['Positive', 'Negative'])
```

In [157]:
```
print("CMV and HIV comparison for ages below 18")
get_contable(ages_below_18)
```

```
CMV and HIV comparison for ages below 18
Pearson correlation: -0.04935370648455273
```

Out[157]:

	CMV	HIV
Positive	1046	10

Negative	133	1129

In [158]:
```
print("CMV and HIV comparison for ages 19 to 35")
get_contable(ages_18_35)
```

```
CMV and HIV comparison for ages 19 to 35
Pearson correlation: 0.04287055851192768
```

Out[158]:

	CMV	**HIV**
Positive	422	42
Negative	24	404

In [159]:
```
print("CMV and HIV comparison for ages 36 to 50")
get_contable(ages_35_50)
```

```
CMV and HIV comparison for ages 36 to 50
Pearson correlation: -0.0017037378225520808
```

Out[159]:

	CMV	**HIV**
Positive	264	39
Negative	14	239

In [160]:
```
print("CMV and HIV comparison for ages 51 to 79")
get_contable(ages_50_80)
```

```
CMV and HIV comparison for ages 51 to 79
Pearson correlation: 0.047870405587604754
```

Out[160]:

	CMV	**HIV**
Positive	232	9
Negative	14	237

In [161]:
```
print("CMV and HIV comparison for ages 80+")
get_contable(ages_80_above)
```

```
CMV and HIV comparison for ages 80+
Pearson correlation: undefined
```

Out[161]:

	CMV	**HIV**
Positive	24	0
Negative	1	25

In [ ]:

In [ ]:

In [ ]:

```
p_value <- array(dim=c(500,1))
interval <- array(dim=c(500,1))
lambda <- array(dim=c(500,1))
for (k in c(40,50,60)){
  # cut the palindrome lists into designated interval
  tab <- table(cut(locs, breaks=seq(0, N, length.out=k+1), include.lowest=TRUE))
  # calculate parameter lambda
  lambda[k,] <-sum(as.vector(tab))/k

  matrix = 0
  interval[k,] <- N/k
  for (i in 0:(max(tab)-1)){
    # sum up poisson value of each i
    matrix = matrix + ((lambda[k]^i)*exp(-lambda[k])/factorial(i))
  }
  p_value[k,] <- 1 - matrix^k
}
matrix <- data.frame(lambda, interval, p_value)
# Display Table containing the probability of a Poisson Distribution having e greates
# t number of hits at least k for each sub-interval divisions
matrix[c(40,50,60),]
```