INTRODUCTION TO R: PRACTICAL

(1) Sometimes R does things you might not expect.

(a) (i) Type `1:6 + 1:6`. What happens?

(ii) What do you think will happen if you type `1:6 + 1:3` ? Try it. Explain what happened.

(iii) What do you think will happen if you type `1:6 + 1:4` ? Try it. Explain what happened.

```
> 1:6+1:6
[1]  2  4  6  8 10 12
> 1:6+1:3
[1] 2 4 6 5 7 9
> 1:6+1:4
[1] 2 4 6 8 6 8
Warning message:
In 1:6 + 1:4 :
  longer object length is not a multiple of shorter object length
```

.

When an operation involves two vectors of different lengths, the shorter one is *recycled*. If the lengths are not integer multiples of each other — that is, if the recycling ends with part of a vector being reused — there is an error message.

(b) (i)

```
> x=c(-4,2,3,-5,-1)
> y=1:5
> x < -1
[1]  TRUE FALSE FALSE  TRUE FALSE
> x < y
[1]  TRUE FALSE FALSE  TRUE  TRUE
```

.

(ii)

```
> x<-1
> x
[1] 1
```

.

The expression '`<-`' is an alternative assignment operator. So '`x<-1`' is not a Boolean operation, but rather reassigns to the vector `x` the value 1.

(2) Create the following vectors in R using `seq()` and `rep()`.

(a) $1, 1.5, 2, \ldots, 12$.

(b) $1, 8, 27, \ldots, 1000$.

(c) $1, -\frac{1}{2}, \frac{1}{3}, -\frac{1}{4}, \ldots, -\frac{1}{100}$.

(d) $1, 0, 3, 0, 5, 0, \ldots, 49$.

(e) $1, 9, 36, 100, \ldots, \sum_{i=1}^{k} i^3, \ldots, 44100$. [look up `?cumsum`]

(f) $1, 3, 3, 5, 5, 5, \ldots, \underbrace{19, \ldots, 19}_{10 \text{ times}}$.

```
#1
seq(1,12,.5)
#2
seq(1,10)^{3}
#3
x=seq(1,100)
(-1)^x/x
```

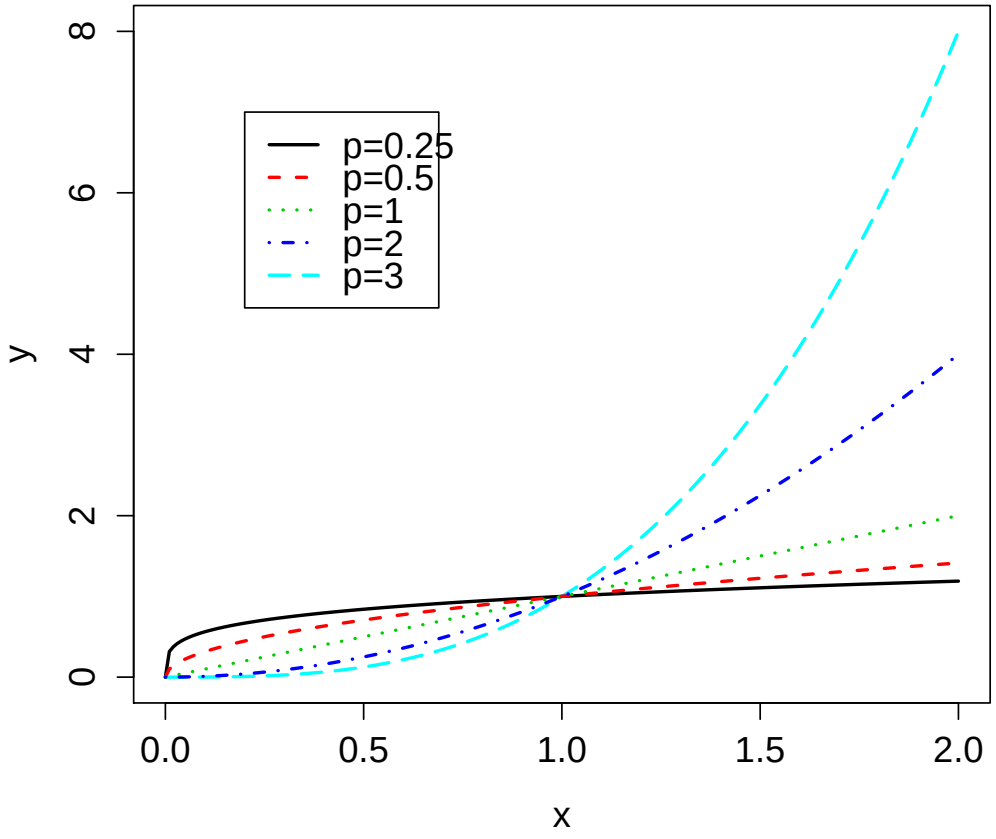
```r
#4
seq(1,49)*rep(c(1,0),length.out=49)
#5
cumsum(seq(1,20)^{3})
#6
rep(seq(1,19,2),seq(1,10))
```

.

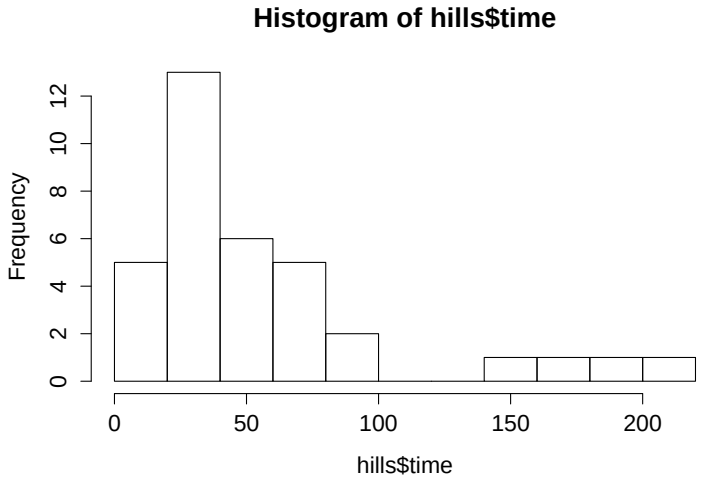
(3) Make a plot of the graphs of the functions $y = x^p$ for $p = \frac{1}{4}, \frac{1}{2}, 1, 2, 3$, and $x \in [0, 2]$, all on the same set of axes, with different colours and line types. For an extra challenge use the function `legend` to add a legend that explains the plot.

```r
x=seq(0,2,.01)
p=c(.25,.5,1,2,3)

plot(x,x^3,type='l',ylab='y',lwd=2,col=5,lty=5)
for(i in 1:4){
  lines(x,x^p[i],lwd=2,col=i,lty=i)
}
legend(.2,7,c('p=0.25','p=0.5','p=1','p=2','p=3'),lwd=2,col=1:5,lty=1:5)
```



(4) Load the package `MASS` with the command `require(MASS)`. The data frame `hills` gives record times in 1984 for 35 Scottish hill races.
  (a) Look at the help file for this data set.
  (b) Try applying commands like `head`, `summary`, `dim`, `attributes`.
  (c) What happens when you plot `hills`?
  (d) Make a histogram of the `time` variable.
  `hist(hills$time)`



  (e) Compute the mean and SD of the times for those races where the climb was above the median, and those where it was below the median.

```
> median(hills$climb)
[1] 1000
> t1=hills$time[hills$climb>1000]
> t2=hills$time[hills$climb<1000]
> mean(t1)
[1] 85.56965
> sd(t1)
[1] 59.12813
> mean(t2)
[1] 32.56171
> sd(t2)
[1] 15.06571
```

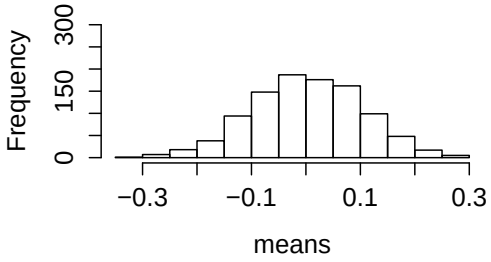
.

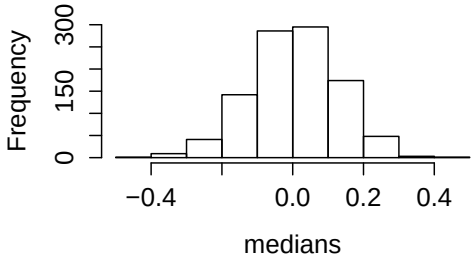
(5) Write a function that takes two integers m and n as inputs, produces m random samples of n independent standard normal random variables, and outputs a list with elements `means`, `medians`, `SDs`, and `IQRs`, each being a vector of length m containing all the means, medians, SDs, and interquartile ranges, respectively, of the samples. For $m = 1000$ and $n = 100$ investigate the results, using histograms and QQ plots. Are all of these summary statistics approximately normally distributed?

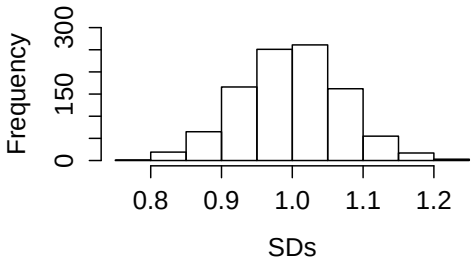
```
iqr=function(x){
  quantile(x,.75)-quantile(x,.25)
}
testfun=function(m,n){
  samples=matrix(rnorm(m*n),m,n) #Each row is a sample of n
  list(means=apply(samples,1,mean),medians=apply(samples,1,median),
       SDs=apply(samples,1,sd),IQRs=apply(samples,1,iqr))
}
```

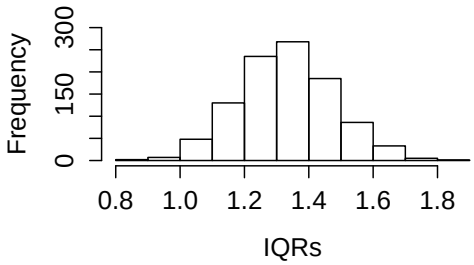
```
9
10 xmmsi=testfun(1000,100)
11
12 par(mfrow=c(2,2))
13 for(i in 1:4){
14   hist(xmmsi[[i]],main='',xlab=names(xmmsi)[i],ylim=c(0,300))
15 }
16 for(i in 1:4){
17   qqnorm(xmmsi[[i]],main=paste('Normal QQ plot for ',names(xmmsi)[i]))
18   qqline(xmmsi[[i]],lwd=2,col=2)
19 }
```

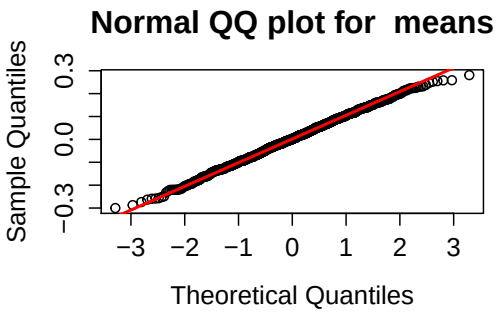
.

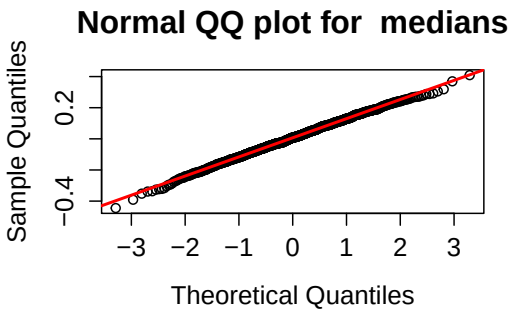

Frequency
0
150
300
−0.3
−0.1
0.1
0.3
means

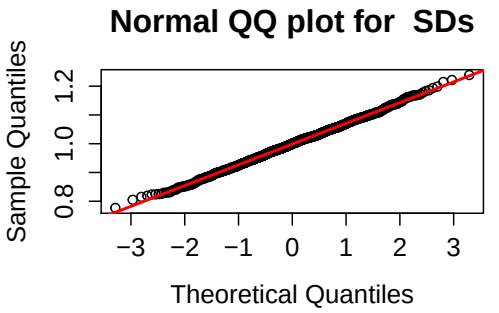

Frequency
0
150
300
−0.4
0.0
0.2
0.4
medians

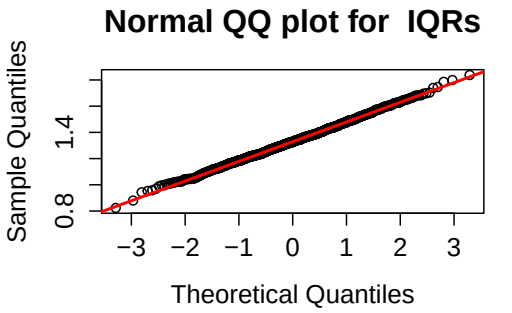

Frequency
0
150
300
0.8
0.9
1.0
1.1
1.2
SDs


Frequency
0
150
300
0.8
1.0
1.2
1.4
1.6
1.8
IQRs


Normal QQ plot for means
Sample Quantiles
−0.3
0.0
0.3
−3
−2
−1
0
1
2
3
Theoretical Quantiles


Normal QQ plot for medians
Sample Quantiles
−0.4
0.2
−3
−2
−1
0
1
2
3
Theoretical Quantiles


Normal QQ plot for SDs
Sample Quantiles
0.8
1.0
1.2
−3
−2
−1
0
1
2
3
Theoretical Quantiles


Normal QQ plot for IQRs
Sample Quantiles
0.8
1.4
−3
−2
−1
0
1
2
3
Theoretical Quantiles

(6) Surveys are often made easier by performing *cluster samples*: Instead of trying to survey n people uniformly chosen from the whole population, we choose a few geographic sites, and sample only from those sites. Suppose we have a country with T districts, where $T > 10$. A certain variable of interest is normally distributed within each district, with mean μ_D and variance σ^2. The district means are $\{0, 1/(T-1), 2/(T-1), \ldots, 1\}$. We randomly choose 10 districts, and then survey 100 individuals from each of those districts.

(a) Write an R function that takes arguments (T, σ) and outputs a simulation of the 1000 samples produced by this surveying procedure. You may wish to look up the `sample` command.

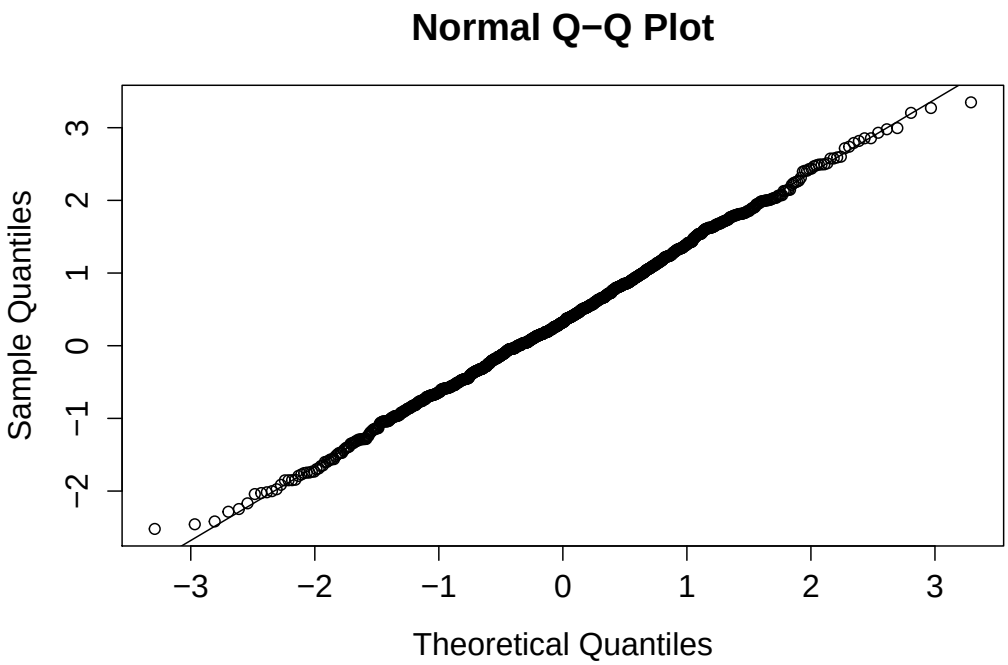
```
cluster=function(T,sigma){
  distmeans=seq(0,1,1/(T-1))
  distmeansamp=sample(distmeans,10)
  indivsamples=rnorm(1000,0,sigma)+distmeansamp
}
```

.

(b) Would you expect these 1000 numbers to be normally distributed? Test this graphically with the functions `qqnorm` and `qqline`.

```
T=20
c=cluster(T,1)
qqnorm(c)
qqline(c)
```

.



It looks very much like a normal distribution.

(c) Write a function that produces a vector of length `m`, each of which is the mean of 1000 samples produced by the above surveying procedure.

```
clustermeans=function(m,T,sigma){
  sapply(1:m,function(i) mean(cluster(T,sigma)))
}
```

.

(d) Estimate the variance of the sample means. (If you can, try to compute the variance theoretically.) How does it compare to the variance you would have had if the 1000 samples were uniform over the population? Let $Y_1, \ldots, Y_{10}$ be the 10 district means. Then the mean of the 1000 samples is just the mean of 1000 i.i.d. normal random variables with expectation 0 and variance σ^2, and the mean of the 10 district means. They are independent of each other, so the variance of the sum is the sum of the variance. The first component has variance $\sigma^2/1000$. The variance of the second component is

$$\frac{s^2}{10} frac T - 10 T,$$

where s^2 is the variance of the distribution of district means. Since these are uniform on $0, 1/(T-1), \ldots, 1$, we have

$$s^2 = \frac{T^2-1}{12} \cdot \frac{1}{(T-1)^2} = \frac{1}{12} \cdot \frac{T+1}{T-1}.$$

So the variance is

$$\frac{\sigma^2}{1000} + \frac{1}{120} \cdot frac (T+1)(T-10)T(T-1).$$

(e) Would you expect the sample means to be normally distributed? Test this.

(7) (a) Count the number of Adenines, Guanines, Cytosines and Thymines (As, Gs, Cs and Ts). Then there is the dinucleotide composition, that is, the number of AAs, AGs etc. (along one strand). There further is the trinucleotide composition, that is, the number of AAAs, AAGs etc.

```
> table(ecp)
ecp
      A       C       G       T
1142742 1180091 1177437 1141382
>
> table(rbind(ecp[-n],ecp[-1]))

> table(ecp[-n],ecp[-1])

         A      C      G      T
  A 338006 256773 238013 309950
  C 325327 271821 346793 236149
  G 267384 384102 270252 255699
  T 212024 267395 322379 339584

> table(ecp[-c(n-1,n)],ecp[-c(1,n)],ecp[-c(1,2)])
, ,  = A


         A      C      G      T
  A 108964  58664  56659  63721
  C  76654  86491  70971  26770
  G  83530  96071  56222  52688
  T  68858  84101  83532  68845

, ,  = C


         A      C      G      T
```

```
  A  82616  74935  80909  86523
  C  66782  47807 115734  42746
  G  54764  93028  92189  54247
  T  52611  56051  95270  83879

, ,  = G


      A      C      G      T
  A 63405  73288  50653  76282
  C 104850 87076  86904 102957
  G 42503 114670  47515  66142
  T 27254  71759  85180  76998

, ,  = T


      A      C      G      T
  A 83021  49886  49792  83424
  C 77041  50447  73184  63676
  G 86587  80333  74326  82622
  T 63301  55483  58397 109862
```

.

(i) If you had to assign a probability to observing an A at a stated position on the *E. coli* genome, what figure would you use and why? Under what assumptions does this seem appropriate?
Assuming A's are evenly spread through the genome, $1142742/4641652 = 0.246193$.

(ii) Your are told there is an A at a position along the *E. coli* genome. What probability would you assign to it being followed by a G, and why? The fraction of A's that are followed by G's is $238013/1142742 = 0.2082824$.

(iii) Does the *E. coli* composition data suggest that the event we observe a G at one site is independent (in some suitable sense) of the previous two bases? Explain fully, illustrating with appropriate data.

```
> load(url('http://steinsaltz.me.uk/DTC/ecoli.rda'))
> n=length(ecp)
> table(ecp)
ecp
      A       C       G       T
1142742 1180091 1177437 1141382
> table(ecp[-c(n-1,n)],ecp[-c(1,n)],ecp[-c(1,2)])
, ,  = A


      A      C      G      T
  A 108964  58664  56659  63721
  C 76654  86491  70971  26770
  G 83530  96071  56222  52688
  T 68858  84101  83532  68845

, ,  = C


      A      C      G      T
  A 82616  74935  80909  86523
  C 66782  47807 115734  42746
```

```
  G  54764  93028  92189  54247
  T  52611  56051  95270  83879

, ,  = G


       A      C      G      T
  A  63405  73288  50653  76282
  C 104850  87076  86904 102957
  G  42503 114670  47515  66142
  T  27254  71759  85180  76998

, ,  = T


       A      C      G      T
  A  83021  49886  49792  83424
  C  77041  50447  73184  63676
  G  86587  80333  74326  82622
  T  63301  55483  58397 109862

> ft=table(ecp[-c(n-1,n)],ecp[-c(1,n)],ecp[-c(1,2)])
> dim(ft)
[1] 4 4 4
> for(i in 1:4){
+   ft[,,i]=ft[,,i]/sum(ft[,,i])
+ }
> ft
, ,  = A


           A          C          G          T
  A 0.09535319 0.05133622 0.04958166 0.05576154
  C 0.06707907 0.07568732 0.06210594 0.02342613
  G 0.07309618 0.08407067 0.04919925 0.04610669
  T 0.06025687 0.07359585 0.07309793 0.06024550

, ,  = C


           A          C          G          T
  A 0.07000816 0.06349934 0.06856166 0.07331892
  C 0.05659055 0.04051128 0.09807210 0.03622263
  G 0.04640659 0.07883121 0.07812025 0.04596849
  T 0.04458216 0.04749718 0.08073106 0.07107842

, ,  = G


           A          C          G          T
  A 0.05385006 0.06224372 0.04301975 0.06478654
  C 0.08904943 0.07395391 0.07380783 0.08744170
  G 0.03609793 0.09738958 0.04035463 0.05617460
  T 0.02314691 0.06094514 0.07234363 0.06539464

, ,  = T


           A          C          G          T
  A 0.07273726 0.04370666 0.04362431 0.07309034
  C 0.06749800 0.04419817 0.06411876 0.05578851
```

```
  G 0.07586154 0.07038222 0.06511930 0.07238768
  T 0.05545996 0.04861037 0.05116341 0.09625349

> ft=table(ecp[-c(n-1,n)],ecp[-c(1,n)],ecp[-c(1,2)])
> t2/sum(t2)
 [1] 0.02905434 0.06078956 0.08237753 0.06569789 0.05374411
     0.07180936 0.07972736 0.14208318 0.05883309 0.02881407
[11] 0.03239641 0.03375130 0.06220407 0.05160699 0.03784669
     0.05850069 0.05076335
> t2=table(ecp[-n],ecp[-1])
> t2/sum(t2)

            A           C           G           T
  A 0.07282021 0.05531932 0.05127766 0.06677581
  C 0.07008864 0.05856127 0.07471329 0.05087608
  G 0.05760536 0.08275116 0.05822325 0.05508794
  T 0.04567857 0.05760773 0.06945352 0.07316018
```

.

At least to the naked eye it appears that the proportions of the 16 possible pairs preceding a G (the first array) are almost the same as the lower array, which has the overall proportions of those pairs.

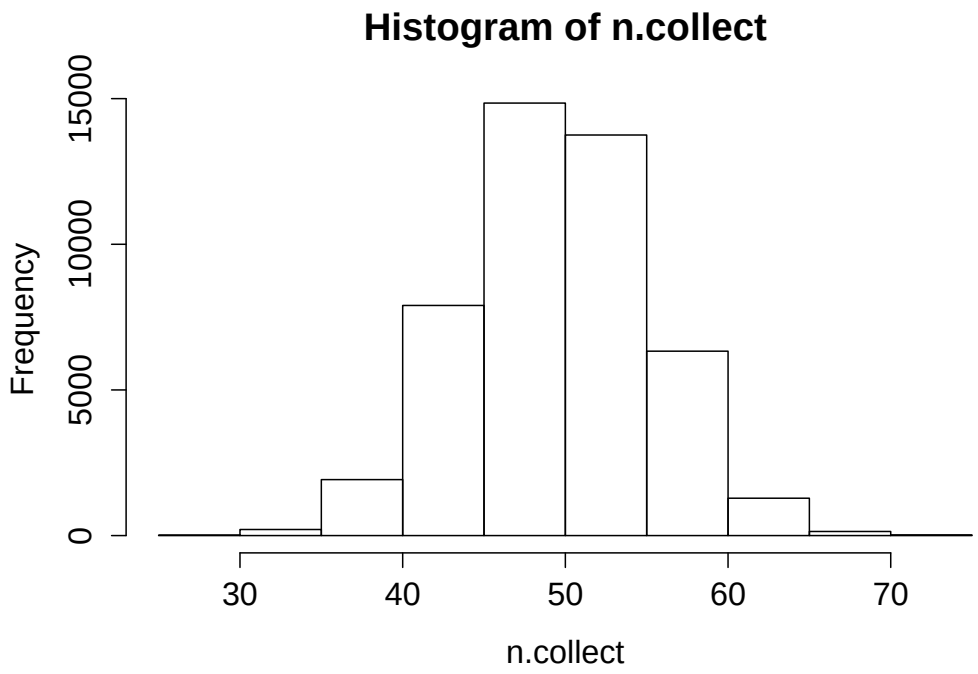
(iv) **Purine counts.**

Divide up the data into about 46,400 blocks of 100 base pairs (bp). Count the number of purines (i.e. A or G). Do the same for the 4,640 blocks of 1,000 bp, and 464 blocks of 10,000 bp.

(A) For each set of counts, calculate the mean and standard deviation of the number of purines per block, and draw histograms of these numbers.

```
> k=100
> n=length(ecp)%/%k -1
> n.collect=NULL
> for (i in 0:n){
+   a=ecp[i*k+(1:100)]
+   n.collect=c(n.collect,sum(a=='A'|a=='G'))
+ }
> sd(n.collect)
[1] 5.678099
> mean(n.collect)
[1] 49.986
> hist(n.collect)
```

.



(B) Compare the results of (i) across the different block sizes and comment.

```
 > k=500
> n=length(ecp)%/%k -1
> n.collect2=NULL
> for (i in 0:n){
+    a=ecp[i*k+(1:100)]
+    n.collect2=c(n.collect2,sum(a=='A'|a=='G'))
+ }
>
> sd(n.collect2)
[1] 5.673332
> mean(n.collect2)
[1] 50.01077
```

.

(C) For each block size, calculate the fraction of the counts within 1, 2 and 3 standard deviations of the mean.

```
> sapply(1:3,function(i) sum(abs(n.collect-mean(n.collect))/sd(n.
    collect)<i))/length(n.collect)
[1] 0.6647492 0.9588719 0.9978456
> #Block size 500
> sapply(1:3,function(i) sum(abs(n.collect2-mean(n.collect2))/sd(
    n.collect2)<i))/length(n.collect2)
[1] 0.6656253 0.9599267 0.9974146
```

.

(D) Repeat (i), (ii) and (iii) for *proportions* (rather than counts) of purines in each block.

Just change `sum` to `mean` in defining `n.collect` and `n.collect2`.

(v) Compute counts of TATAAT in blocks of 5,000 bp. Assuming that these counts follow a Poisson distribution, estimate the parameter of this distribution and

obtain an estimate of the standard error of your parameter estimate. This can be done by either a formula or by simulation.

```
> count.pattern=function(source=ecp,target=c('T','A','T','A','A','T')){
+   k=length(target)
+   n=length(source)-k+1
+   sum(sapply(1:n,function(i) prod(source[i:(i+k-1)]==target)))
+ }
> g=sapply(1:u,function(i) count.pattern(ecp[((i-1)*n+1):(i*n)]))
> table(g)
g
  0   1   2   3   4   5   6   7
611 213  56  25  15   5   2   1
> mean(g) # is the estimate of the Poisson parameter
[1] 0.5431034
# This is the proportion of each count number among the blocks
> round(table(g)/length(g),3)
g
    0     1     2     3     4     5     6     7
0.658 0.230 0.060 0.027 0.016 0.005 0.002 0.001
> round(dpois(0:7,.5431034),3)
# This is what the proportions of each count would be if Poisson
[1] 0.581 0.316 0.086 0.016 0.002 0.000 0.000 0.000
```

.

If the distribution of TATAAT were Poisson — that is, if they were scattered among the 5000-base blocks like independent trials with small probability of success at every point, the Poisson parameter would be the mean number of "successes" per block, which is 0.5431. In fact, over the 928 blocks the distribution of different count numbers is very different from Poisson.