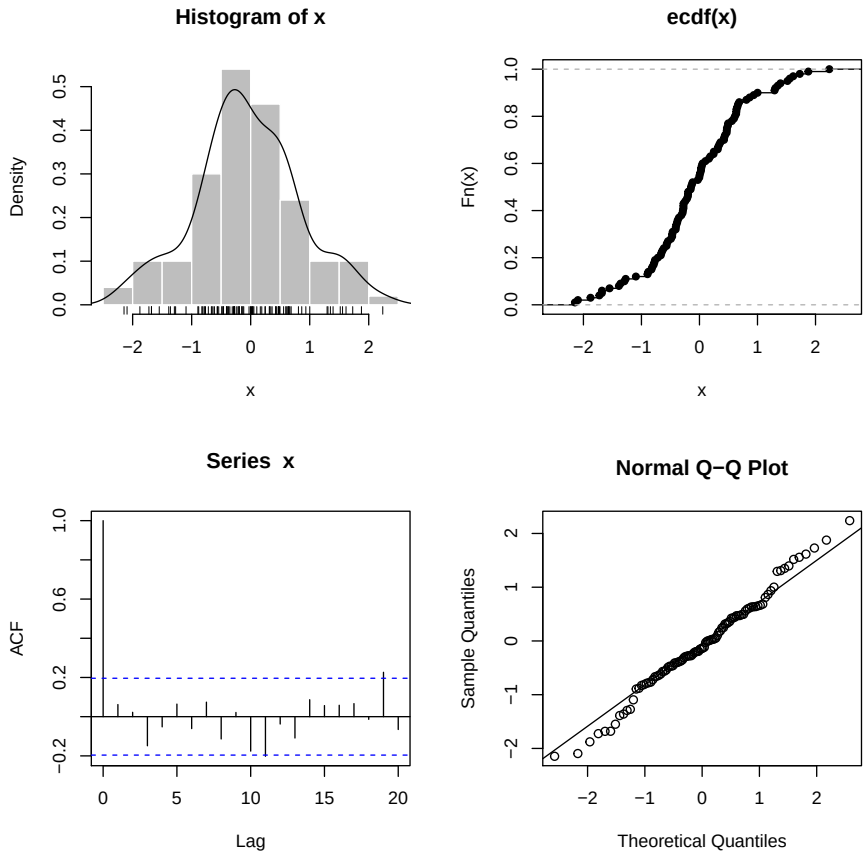
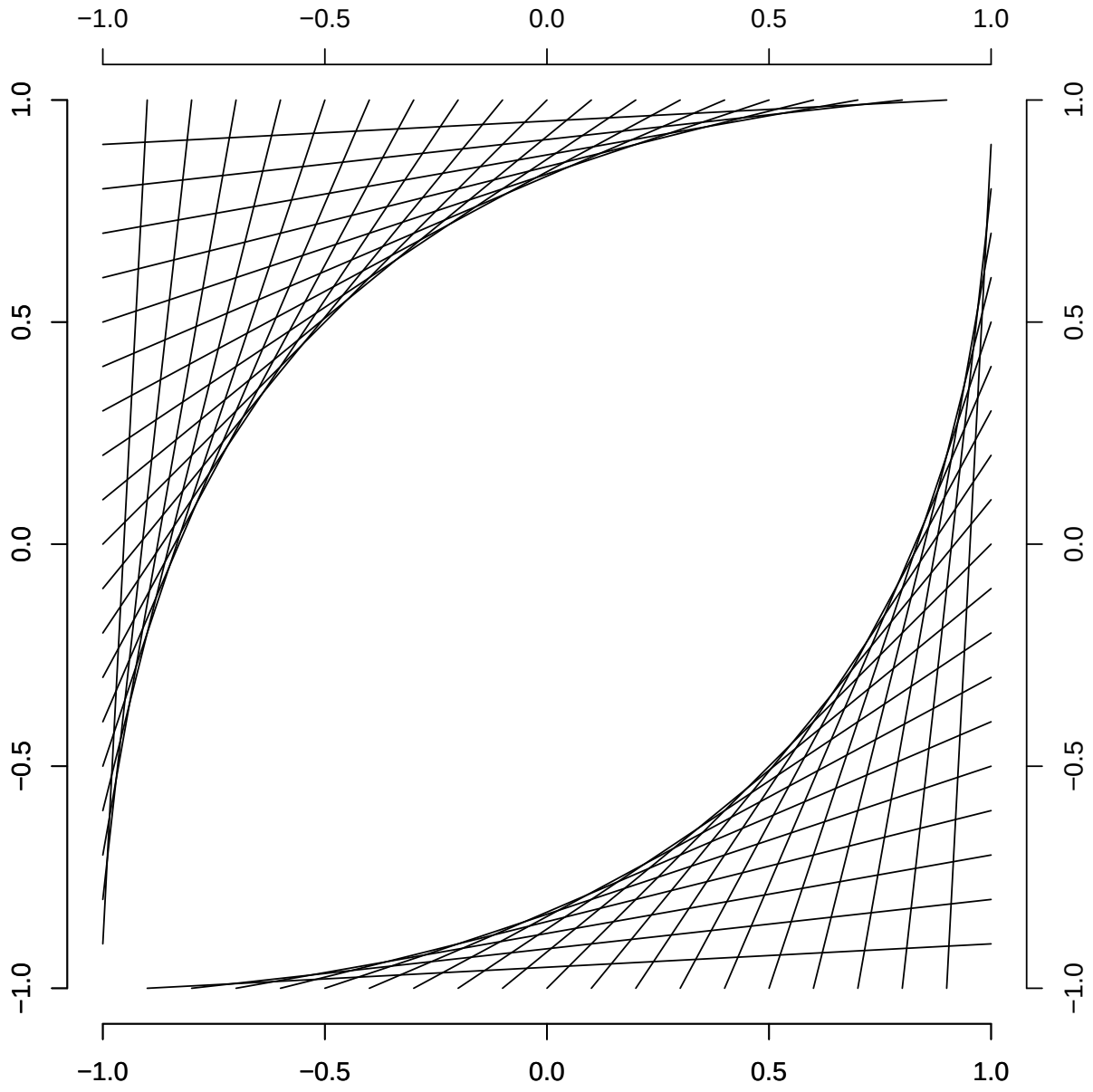
You have a sample of data values $x_i, i = 1, 2, \ldots, n$. Write `R` code to produce the following 2×2 graphical summary of this sample. The four panels of this graphical summary consist of

1. a probability density histogram of the sample, overlaid with the empirical density (`density()` in `R`), and a `rug` representation of the sample;
2. the empirical CDF (`ecdf()` in `R`) of the sample;
3. the autocorrelation function of the sample; and
4. a quantile-quantile plot against standard normal quantiles.

See an example of such a 2×2 summary below. Test your code using a randomly-generated $N(0, 1)$ sample of size 100. Also create a `pdf` image of the graphical summary using an appropriate `R` graphics device (screenshot are not allowed).



The following figure was made using only straight lines and standard `R` graphics functions. Construct `R` code that will produce this plot exactly. Create a `pdf` file of this plot (screenshot not allowed).



Which of the in-built functions `mean()` and `median()` is faster? To find out, repeat the following two steps M times:

1. Generate N random numbers $x_1, \ldots, x_N$ with the uniform density $U(0, 1)$.
2. Record initial time T_0, calculate `mean`, record final time T_1, and calculate the time difference $\delta_i = |T_1 - T_0|$.

Finally, compute the average time difference $\bar{\delta}_N = M^{-1} \sum_{i=1}^{M} \delta_i$. Take $N = 10, 100, 1000, 10000$ and $M = 1000$. Plot $\bar{\delta}_N$ vs. N.

Repeat the same by replacing `mean` with `median`. What is your conclusion from this exercise?

Below are approximate pairwise distances (in km) between five cities in Maharashtra:

	Nasik	Kopargaon	Shrirampur	Sangamner	Ahmednagar
Nasik	0				
Kopargaon	74	0			
Shrirampur	100	32	0		
Sangamner	66	44	46	0	
Ahmednagar	140	90	60	75	0

A salesperson wishes to visit each of these cities on a route that begins and ends in Nasik. Moreover, (s)he does not wish to visit the same city more than once.

An example of such a "circular" route is: Nasik $\rightarrow$ Ahmednagar $\rightarrow$ Shrirampur $\rightarrow$ Sangamner $\rightarrow$ Kopargaon $\rightarrow$ Nasik, and the total distance traveled by the salesperson on this route is $140 + 60 + 46 + 44 + 74 = 364$.

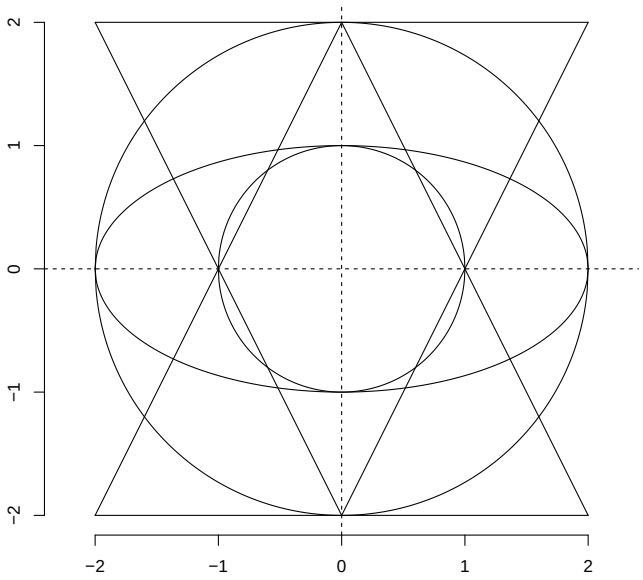
Write `R` code to enumerate all such "circular" routes available to the salesperson, together with the total distance traveled along each route. Which is the shortest route?

Implement the following `R` functions:

1. `circle <- function( radius = 1, center = c(0,0) ) { your implementation }`
2. `ellipse <- function( a = 1, b = 1, center = c(0,0) ) { your implementation }`
3. `triangle <- function( x1, y1, x2, y2, x3, y3 ) { your implementation }`

These respectively add a circle $(x-x_0)^2+(y-y_0)^2 = r^2$, an ellipse $((x-x_0)/a)^2+((y-y_0)/b)^2 = 1$, and a triangle to an existing plot. The argument `center` to the first two functions contains center coordinates (x_0, y_0). Arguments to the `triangle` function are the coordinates of the triangle's vertices. After implementing these functions, recreate the figure below. Produce a `pdf` file of your figure (screenshot not allowed). You may create an empty plot window as follows.

```
plot.new( ); plot.window( xlim = c( -2, 2 ), ylim = c( -2, 2 ), asp = 1 )
axis( 1 ); axis( 2 ); abline( v = 0, h = 0, lty = 2 )
```



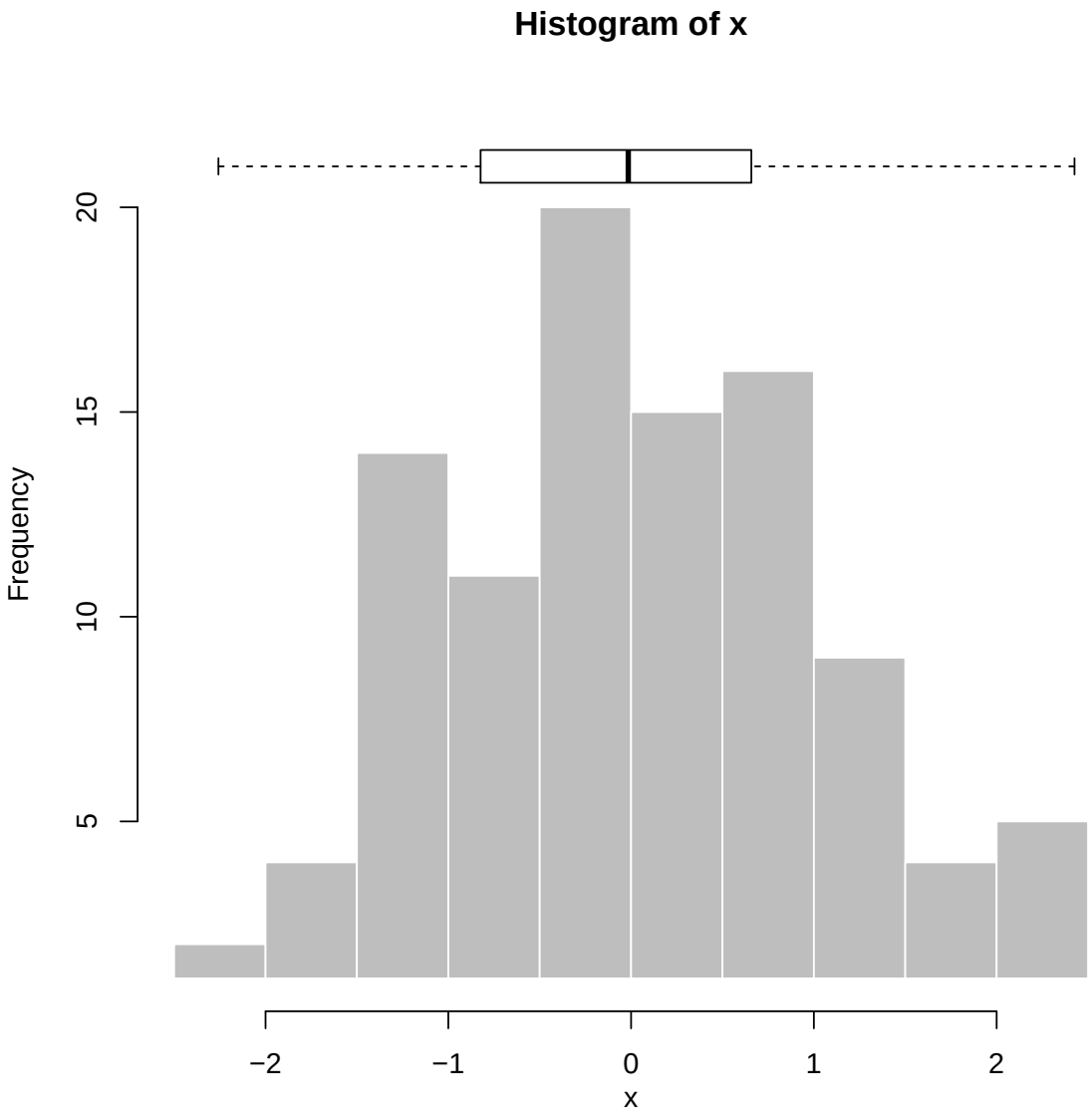
Consider integrals of the form $I = \int_0^1 f(x)dx$. For example, $I = 1/2$ for $f(x) = 1 - x$ and $I = \pi/4$ for $f(x) = \sqrt{1 - x^2}$, etc.

We can *estimate* the above integral as follows: Generate a uniform random sample $x_1, \ldots, x_n$ from the interval $[0, 1]$. An estimator $\widehat{I}_N$ for the above integral is the mean of $f(x_1), \ldots, f(x_N)$; i.e., $\widehat{I}_n = (1/n) \sum_{i=1}^n f(x_i)$.

For the two functions above, compare your estimate with the exact value I of the integral. Plot $\widehat{I}_n$ as a function of n and make a `pdf` of this plot through `R` (screenshot not allowed).

Write `R` code that computes the first $n+1$ Fibonacci numbers $f_0, \ldots, f_n$. Fibonacci numbers are defined recursively through $f_{i+1} = f_i + f_{i-1}$ with seed values $f_0 = 0, f_1 = 1$. Make a plot of f_{n+1}/f_n as function of n upto a sufficiently large value of n. The limit of the ratio sequence $\phi_n = f_{n+1}/f_n$ as $n \to \infty$ is called the *golden ratio* ϕ. Report your best approximation for the golden ratio to full 15-digit accuracy. Compare your values ϕ_n with the exact value $\phi = (1 + \sqrt{5})/2$, and make a plot of the relative error as function of n.

Write `R` code to add a horizontal boxplot on top of a histogram, as in the figure below. The width of the boxplot should look the same irrespective of the range of histogram counts. Test your code using a randomly-generated $N(0, 1)$ sample of size 100. Create a `pdf` file of this plot (screenshot not allowed).



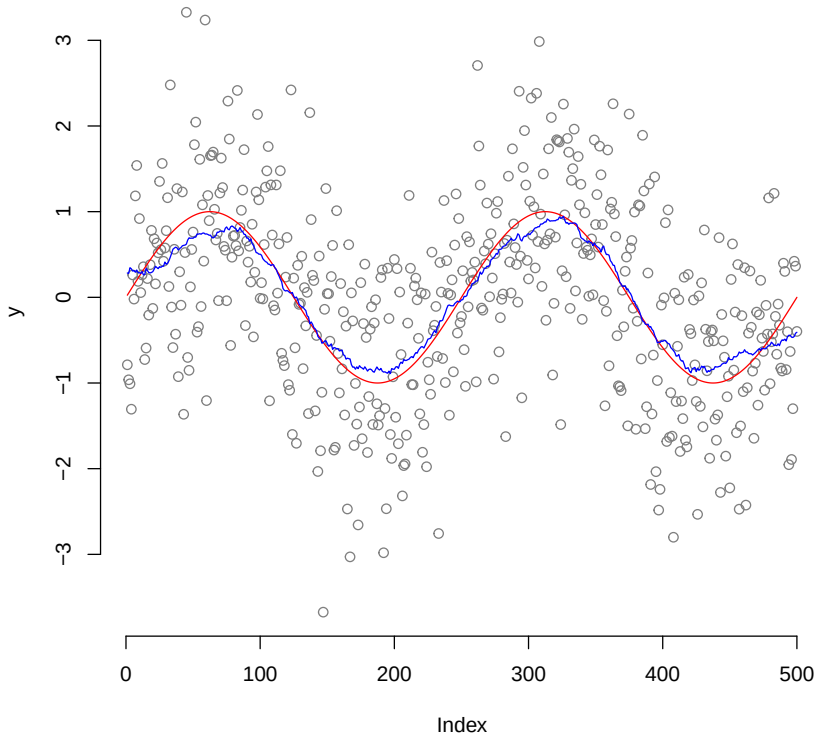
The *running mean* of a data vector $y_1, \ldots, y_n$ is defined as

$$\bar{y}_i := \sum_{j=\max(1,i-w)}^{\min(n,i+w)} y_j, \quad 1 \leq i \leq n,$$

where w is the *window size*; $0 \leq w \leq n$. Write `R` code to compute the running-mean series $\bar{y}$ given a vector of values y and a window size w. Generate a vector y as follows:

```
n      <- 500
signal <- sin( 4 * pi * ( 1:n ) / n )
noise  <- rnorm( n, sd = 1 )
y      <- signal + noise
w      <- 30
```

Run your running-mean code on this `y` with window size $w = 30$. Create a plot with y represented by points, `signal` represented by a red line, and $\bar{y}$ represented with a blue line. See the example plot below.

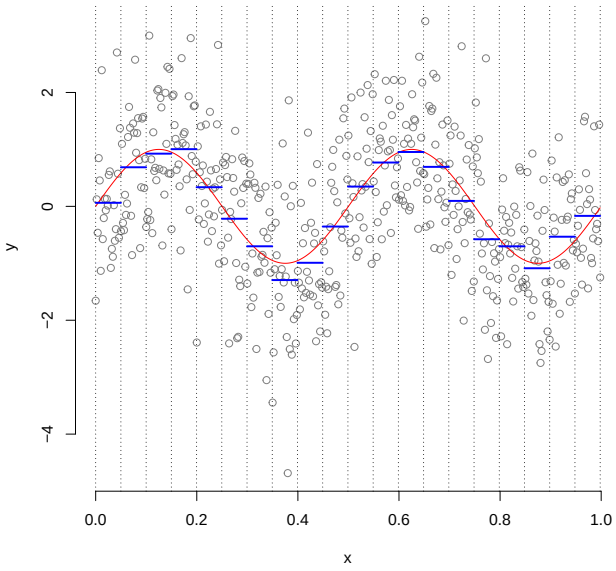


We are given data of the form $(x_1, y_1), \ldots, (x_n, y_n)$. Possibly, there is some relationship between x and y. A *regressogram* $r(x)$ is a way of guessing this relationship. A regressogram is computed as follows: Divide the range of x values in m bins. Therefore, the bins are $[b_0, b_1), [b_1, b_2), \ldots, [b_{m-1}, b_m]$, where $b_0 = \min(x), b_1 = h, b_2 = 2h, \ldots, b_{m-1} = \max(x) - h, b_m = \max(x)$, and $h = (\max(x) - \min(x))/m$. The regressogram $r(x)$ at x is the mean of all y values belonging to the bin that x belongs to. When plotted, it is a piecewise flat function (see example below).

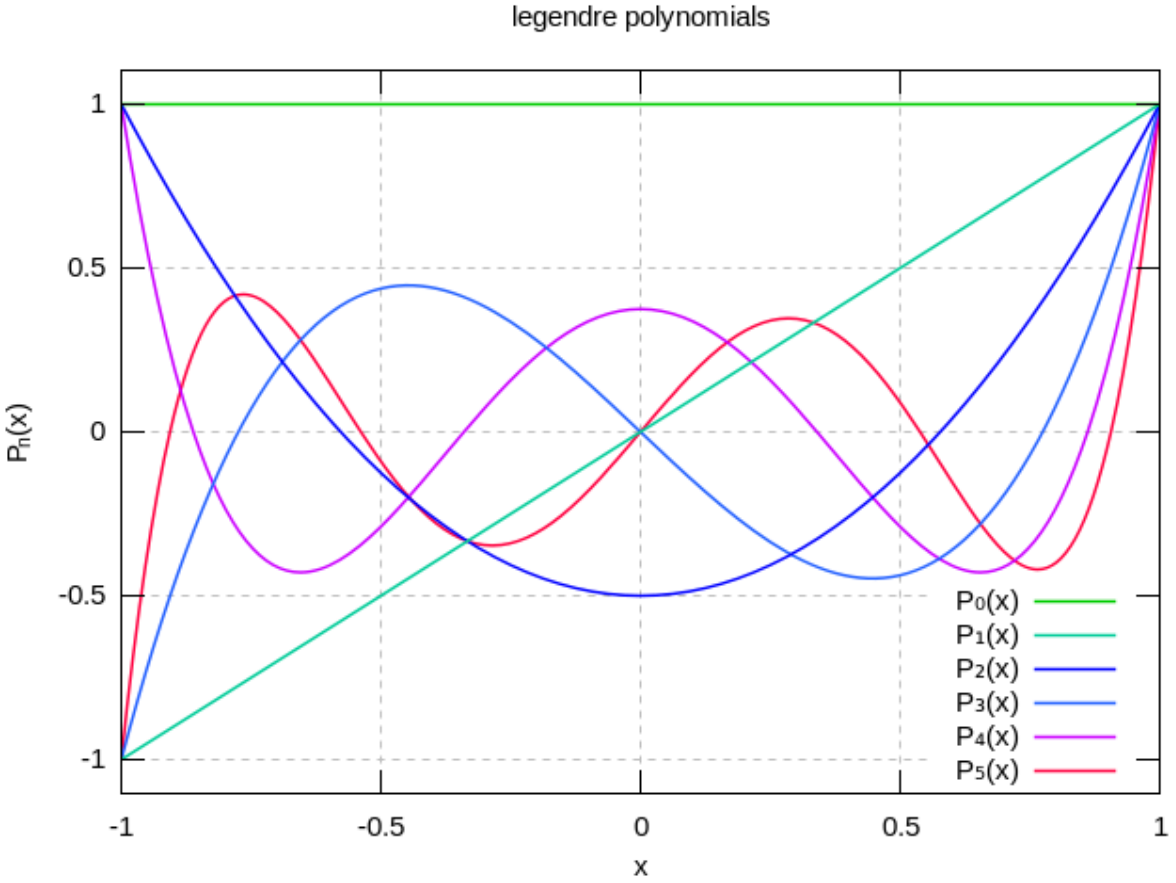
Write `R` code to compute regressogram, given vectors x, y and bin count m. Generate vectors x, y as follows:

```
n      <- 500
x      <- ( 0:( n - 1 ) ) / n
signal <- sin( 4 * pi * x )
noise  <- rnorm( n, sd = 1 )
y      <- signal + noise
m      <- 20
```

Compute the regressogram of this data (`x` and `y`) using your code, for a bin count $m = 20$. Create a plot with y represented by points, `signal` represented by a red line, and $r(x)$ represented with flat blue lines. as in the example plot below.



The *Legendre polynomials* are a family of polynomials defined over the interval $[-1, +1]$. $P_0(x)$, which is the Legendre polynomial of order 0, is 1 over this entire interval (i.e., $P_0(x) = 1$). $P_1(x)$, which is the Legendre polynomial of order 1, equals x over this entire interval (i.e., $P_1(x) = x$). The figure below plots P_0 through P_5.

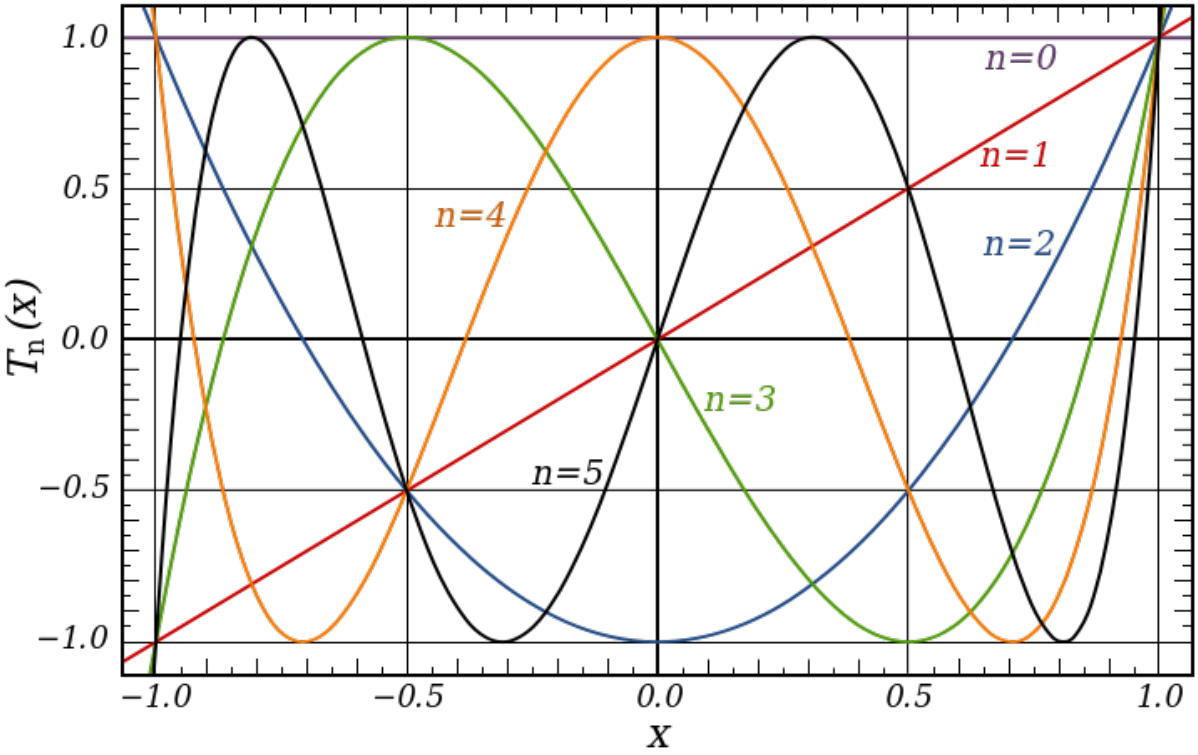


$P_n(x)$, which is the Legendre polynomial of order n, is defined recursively as

$$P_n(x) = \frac{1}{n}\left[(2n-1)xP_{n-1}(x) + (n-1)P_{n-2}(x)\right].$$

Write `R` code to compute $P_n(x)$ given a vector of x values and order n. Using your code, create a plot similar to the one above.

The *Chebyshev polynomials* are a family of polynomials defined over the interval $[-1, +1]$. $T_0(x)$, which is the Chebyshev polynomial of order 0, is 1 over this entire interval (i.e., $T_0(x) = 1$). $T_1(x)$, which is the Chebyshev polynomial of order 1, equals x over this entire interval (i.e., $T_1(x) = x$). The figure below plots T_0 through T_5.

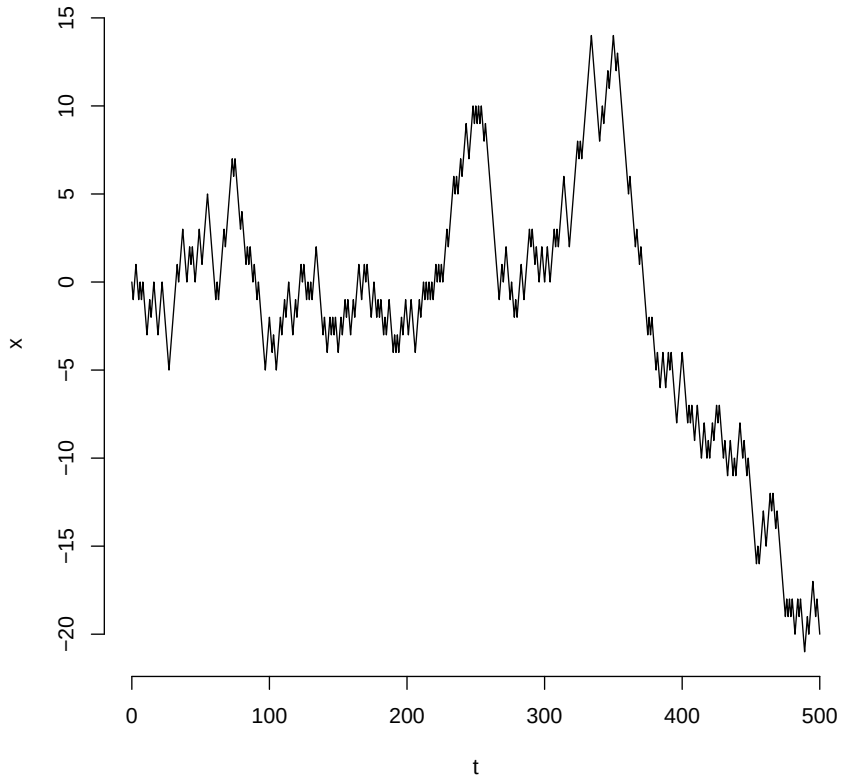


$T_n(x)$, which is the Chebyshev polynomial of order n, is defined recursively as

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x).$$

Write `R` code to compute $T_n(x)$ given a vector of x values and order n. Using your code, create a plot similar to the one above.

A 1-dimensional symmetric random walk is a series of random steps of length 1, either to the right or to the left of the current position of the random walker, with equal probability. In other words, starting at x_0 at time $t = 0$, the position x_1 of the random walker at time $t = 1$ is either $x_1 = x_0 + 1$ or $x_1 = x_0 - 1$ with probability 0.5 each. See the example below.

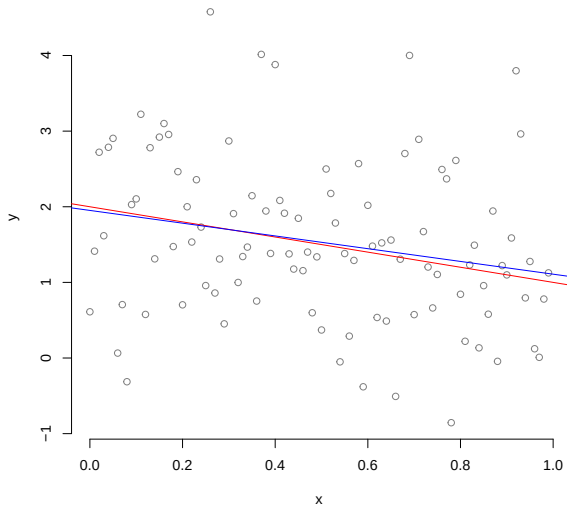


Write `R` code to simulate such a random walk of n steps. At each time, you will have to choose a displacement of $+1$ or -1 randomly with probability 0.5 each: This can be done in a number of ways in `R`. Starting from $x_0 = 0$, generate a realization of this random walk for $n = 500$ steps. Make a plot of this random walk, and save it as `pdf` (screenshot not allowed). Your particular plot will in general look very different from the one shown here.

We are given data of the form $(x_1, y_1), \ldots, (x_n, y_n)$. The relationship between y and x is known to be linear; i.e., of the form $y = mx + c$, where m and c are the unknowns. A way of estimating the parameters m and c is the least-squares way, where one minimizes $\sum_{i=1}^{n}(y_i - (mx_i + c))^2$ with respect to m and c. The resulting estimates $\widehat{m}$ and $\widehat{c}$ of m and c have the form

$$\begin{aligned} \widehat{m} &= \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \\ \widehat{c} &= \bar{y} - \widehat{m}\bar{x}, \end{aligned}$$

where $\bar{x} \equiv$ average value of $x_1, \ldots, x_n$, and $\bar{y} \equiv$ average value of $y_1, \ldots, y_n$. See the figure below, where the red line is the true relationship, and the blue line is the least-squares fit.



Write `R` code to compute these estimates given vectors x and y of length n. Generate vectors x, y as follows:

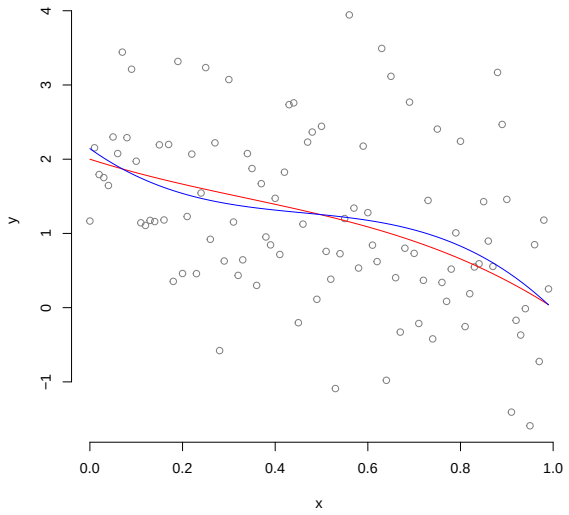
```
n      <- 100
x      <- ( 0:( n - 1 ) ) / n
m      <- -1
c      <-  2
signal <- m * x + c
noise  <- rnorm( n, sd = 1 )
y      <- signal + noise
```

Estimate the parameters m and c from this data using your code. Create a plot with `y` represented by points, `signal` represented by a red line, and the fit $\widehat{y} = \widehat{m}x + \widehat{c}$ represented with a blue line.

Given data of the form $(x_1, y_1), \ldots, (x_n, y_n)$, and a relationship between y and x of the form $y = a_0 + a_1 x + \ldots + a_k x^k$, a way of estimating the parameters $a_0, \ldots, a_k$ is the least-squares way, where one minimizes $\sum_{i=1}^{n} (y_i - (a_0 + a_1 x_i + \ldots + a_k x_i^k))^2$ with respect to $a_0, \ldots, a_k$. This leads to the solution

$$\widehat{a} \equiv \begin{bmatrix} \widehat{a}_0 \\ \vdots \\ \widehat{a}_k \end{bmatrix} = (X^T X)^{-1} X^T \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad \text{where } X = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^k \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^k \end{bmatrix}.$$

Notice that X is a $n \times (k+1)$ matrix. This solution involves forming the matrix X, taking its transpose X^T, inverting a matrix, several matrix multiplications, and a matrix-vector product. In the figure below, the red line is the true relationship (which is cubic polynomial), and the blue line is the least-squares fit $\widehat{y} = \widehat{a}_0 + \widehat{a}_1 x + \ldots + \widehat{a}_k x^k$ using the estimates $\widehat{a}$ obtained as above.



Write `R` code to obtain estimates $\widehat{a}$ as described above, given vectors x and y of length n, and the degree k of the fitting polynomial. Generate vectors x, y as follows:

```
n      <- 100
x      <- ( 0:( n - 1 ) ) / n
a      <- c( 2, -2, 2, -2 )
k      <- length( a )
signal <- a[1] + sapply( x, function( t ) { sum( a[-1] * t^( 1:(k-1) ) ) } )
noise  <- rnorm( n, sd = 1 )
y      <- signal + noise
```

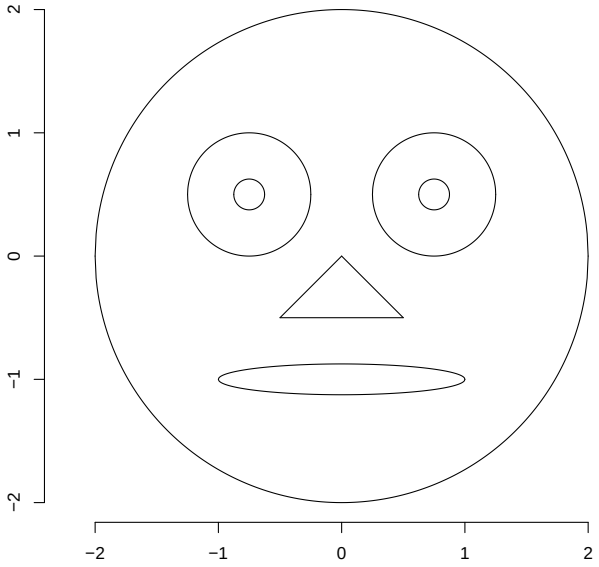
Estimate the parameters a from this data using your code. Create a plot with `y` represented by points, `signal` represented by a red line, and the fit $\widehat{y}$ represented with a blue line.

Implement the following `R` functions:

1. `circle <- function( radius = 1, center = c(0,0) ) { your implementation }`
2. `ellipse <- function( a = 1, b = 1, center = c(0,0) ) { your implementation }`
3. `triangle <- function( x1, y1, x2, y2, x3, y3 ) { your implementation }`

These respectively add a circle $(x-x_0)^2+(y-y_0)^2 = r^2$, an ellipse $((x-x_0)/a)^2+((y-y_0)/b)^2 = 1$, and a triangle to an existing plot. The argument `center` to the first two functions contains center coordinates (x_0, y_0). Arguments to the `triangle` function are the coordinates of the triangle's vertices. After implementing these functions, recreate the figure below. Produce a `pdf` file of your figure (screenshot not allowed). You may create an empty plot window as follows.

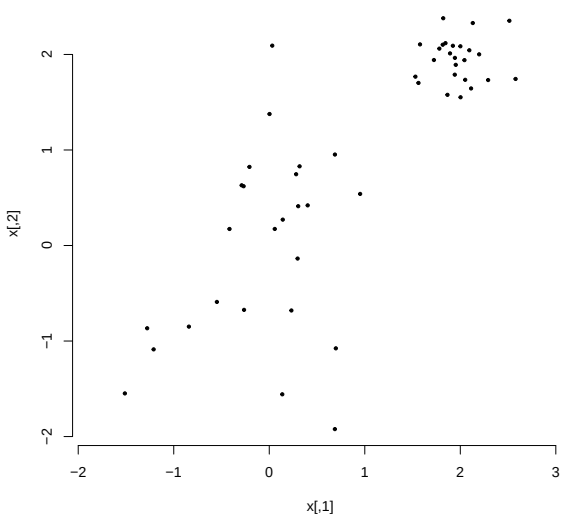
```
plot.new( ); plot.window( xlim = c( -2, 2 ), ylim = c( -2, 2 ), asp = 1 )
axis( 1 ); axis( 2 )
```

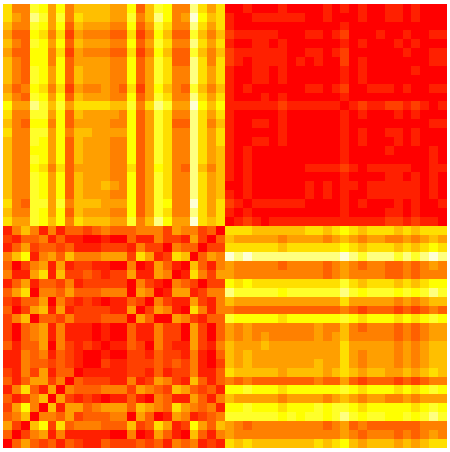


Consider a collection of n points in the x–y plane, such as that in the left-hand figure below. In the R code below, this collection is represented as a $n \times 2$ matrix representing the x (column 1) and y (column 2) coordinates.

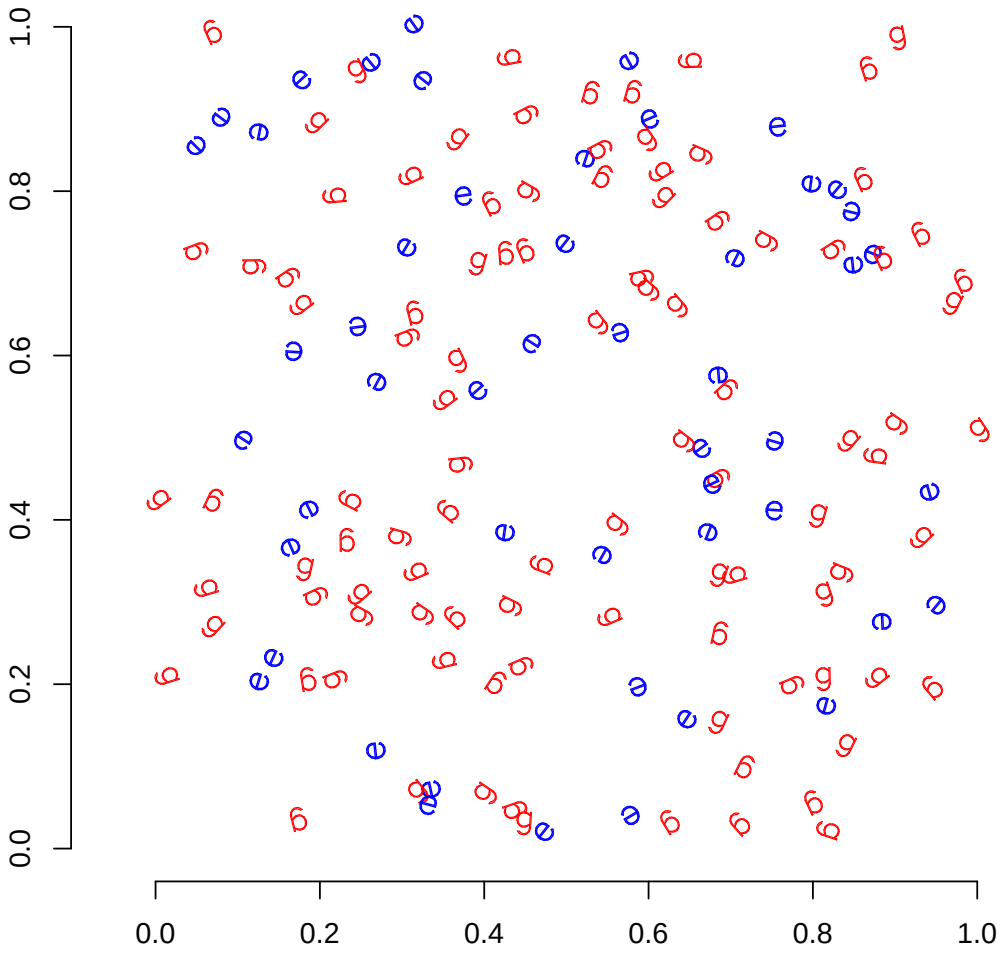
```
k <- 25
x <-           cbind( rnorm( k, mean = 0, sd = 0.75 ), rnorm( k, mean = 0, sd = 0.75 ) )
x <- rbind( x, cbind( rnorm( k, mean = 2, sd = 0.25 ), rnorm( k, mean = 2, sd = 0.25 ) ) )
n <- 2 * k
```

Write R code to compute the pairwise distances for this collection of n points, and store them as a $n \times n$ matrix. Further, create a color image of this matrix using your favorite color scheme; see the right-hand figure below for an example. For your collection these n points, produce plots similar to the ones below.





Write R code to produce a "scrambled eggs" plot similar to the one below. This plot consists of letters 'e' and 'g' placed with random orientation at random locations in the unit square, and the proportion of 'g' is twice that of 'e'.



Hint: Consider R functions that allow adding text to a plot.

Consider a text passage such as this one below:

Long years ago, we made a tryst with destiny, and now the time comes when we shall redeem our pledge, not wholly or in full measure, but very substantially. At the stroke of the midnight hour, when the world sleeps, India will awake to life and freedom. A moment comes, which comes but rarely in history, when we step out from the old to the new, when an age ends, and when the soul of a nation, long suppressed, finds utterance. It is fitting that at this solemn moment we take the pledge of dedication to the service of India and her people and to the still larger cause of humanity.

Suppose this or a similar piece of text is saved in a file without any punctuation marks. Read this file in `R` using `scan()` with `what=character(0)`. Write `R` code that will find the set of unique words in this array, ignoring case. Further, find the count of each unique word. Report the unique words (in alphabetical order) and their counts.

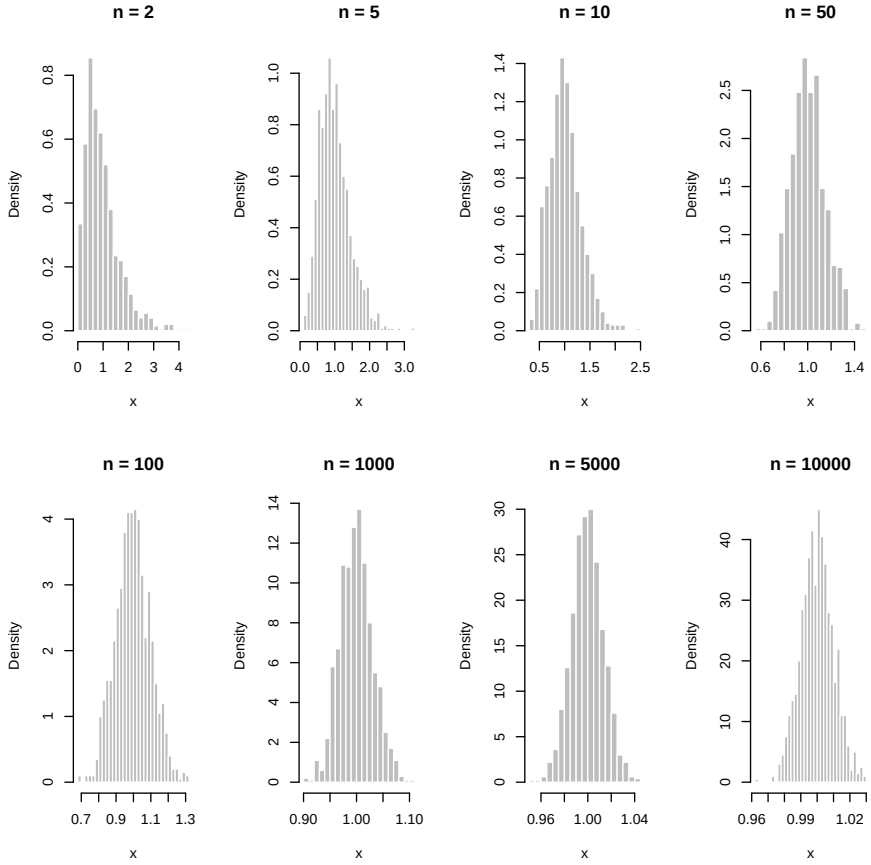
For example, the unique words (in alphabetical order) and their counts in the passage above are

```
a 3 age 1 ago 1 an 1 and 5 at 2 awake 1 but 2 cause 1 comes 3 dedication 1 destiny 1
ends 1 finds 1 fitting 1 freedom 1 from 1 full 1 her 1 history 1 hour 1 humanity 1
in 2 india 2 is 1 it 1 larger 1 life 1 long 2 made 1 measure 1 midnight 1 moment 2
nation 1 new 1 not 1 now 1 of 5 old 1 or 1 our 1 out 1 people 1 pledge 2 rarely 1 redeem 1
service 1 shall 1 sleeps 1 solemn 1 soul 1 step 1 still 1 stroke 1 substantially 1 suppressed 1
take 1 that 1 the 10 this 1 time 1 to 4 tryst 1 utterance 1 very 1
we 4 when 5 which 1 wholly 1 will 1 with 1 world 1 years 1
```

How can we illustrate the central limit theorem? Here is a recipe. Fix sample size to some integer $n > 0$. Generate a sample $x_1, \ldots, x_n$ from any distribution, say an exponential distribution. Compute the sample average

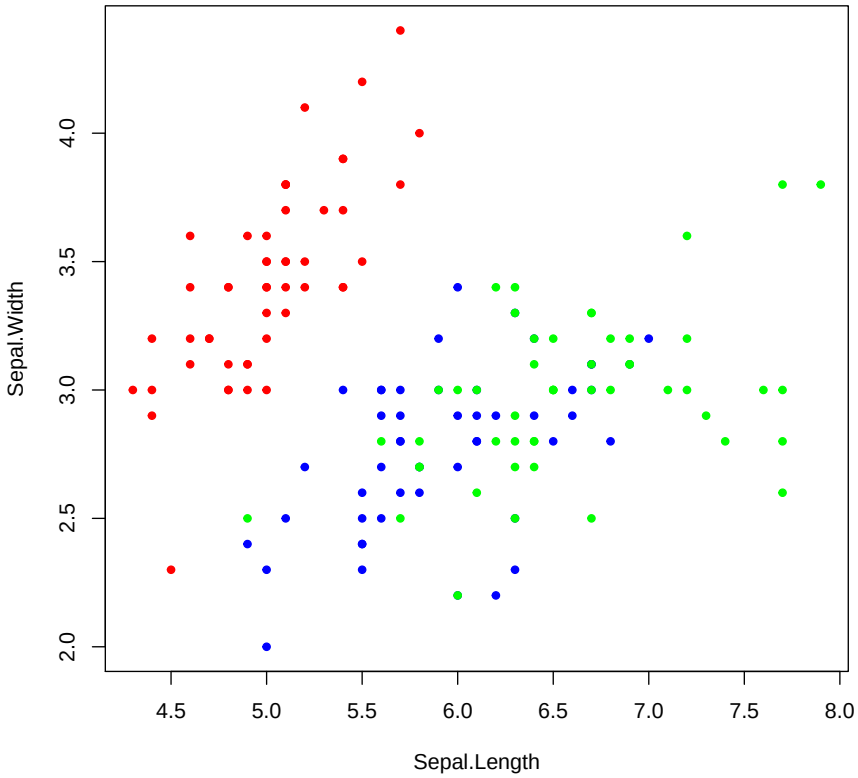
$$\bar{x}_n = n^{-1} \sum_{i=1}^{n} x_i$$

for this sample. Repeat this a larger number of times (say, $m = 1000$). You now have a collection of m sample averages. Make a histogram of this collection of sample averages. Repeat the above for a number of increasing n values, say $n = 2, 5, 10, 50, 100, 1000, 5000, 10000$. Present all these histograms together. With increasing n, the histograms should start looking more and more like the normal distribution. This should be even more apparent if you can overlay a normal density curve on top of each histogram. See an example below.



Consider the `iris` data set available in `R`.

1. Make a pair plot of the four numeric variables of this data set.
2. For the four numeric variables of this data set, make boxplots stacked side by side on a common vertical scale.
3. Take any one pair of numeric variables in this data set. Make a scatterplot such that the color of a point represents the species. See the example below.



Consider the iterative process

$$x_{k+1} = f(x_k),$$

where $f(x) = \lambda x(1-x)$, and the iteration starts at some x_0 in the interval [0,1]. Write `R` code to iterate this process n times.

For $\lambda = 0.5, 1, 2,$ and 4, iterate this process $n = 499$ times. For each λ, plot x_i against i $(0 \leq i \leq n)$, and make a histogram of $x_0, \ldots, x_n$. Stack these plots in the form of a 2×4 array.

We are given data of the form $(x_1, y_1), \ldots, (x_n, y_n)$, where no two x_is are identical. For these data, we wish to compute the *Lagrange interpolating polynomial* which has the form

$$L(x) = \sum_{i=1}^{n} y_i l_i(x),$$

where

$$l_i(x) = \frac{(x - x_1)}{(x_i - x_1)} \cdots \frac{(x - x_{i-1})}{(x_i - x_{i-1})} \frac{(x - x_{i+1})}{(x_i - x_{i+1})} \cdots \frac{(x - x_n)}{(x_i - x_n)}.$$

Write `R` code to compute the Lagrange polynomial for given data and plot it over the range of x values in the data. Generate the (x, y) as follows:

```
n <- 10
x <- ( 0:( n - 1 ) ) / n
y <- dnorm( x, 0.5, 0.2 )
```

Write `R` code that returns all primes numbers $\leq$ some positive integer n using the Sieve of Eratosthenes in its simplest form. To find all the prime numbers less than or equal to a given integer n by Eratosthenes' method:

1. Create a list of consecutive integers from 2 through n: (2, 3, 4, ..., n).
2. Initially, let p equal 2, the first prime number.
3. Starting from p, enumerate its multiples by counting to n in increments of p, and mark them in the list (these will be $2p$, $3p$, $4p$, etc.; the p itself should not be marked).
4. Find the first number greater than p in the list that is not marked. If there was no such number, stop. Otherwise, let p now equal this new number (which is the next prime), and repeat from step 3.

When the algorithm terminates, all the numbers in the list that are not marked are prime. Illustrate your code for $n = 100$.

Source for this description of the method: `http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes`