Exercises for Alnarp R Group

February 14, 2018

Contents

1 Introduction

This is a collection of instructions and exercises for R, sometimes stolen shamelessly from sources unknown, intended to serve as introductory material for an R discussion group at SLU-Alnarp.

2 Data Input

2.1 Assign

Assign data to an object using `<-`. Print by writing the object name.

```
a <- 4
a

## [1] 4
```

Create a vector (multiple data points) using `c` as a function. Possible to reassign using the newly created object. A sequence of natural numbers can be created using `:` .

```
a <- c(4, 2)
a

## [1] 4 2

b <- c(a, a)
b

## [1] 4 2 4 2

1:4

## [1] 1 2 3 4
```

The class `data.frame` can contain multiple vectors. The usual format for data is variables as columns, observations as rows (the old VAC-OAR). The class of an object can be retrieved using the function `class`.

```
c <- c("A", "B", "C", "D")
d <- data.frame("Name" = c, "Value" = b)
d

##   Name Value
## 1    A     4
## 2    B     2
## 3    C     4
## 4    D     2
```

```r
class(d)

## [1] "data.frame"
```

An alternative to `<-` is `assign`. This allows to create an object based on a name (a string of characters). The `get` function allows to print an object using the name.

```r
assign("a", 4)
get("a")

## [1] 4
```

Exercise 2.1 *Create a vector of four numbers.*

Exercise 2.2 *Create a data frame with four columns of four numbers each.*

2.2 Data import

Data can be imported from other sources in multiple ways. Simple `.txt` or `.csv` files can be read using `read.table`. Excel can be imported using the `readxl` package. The `foreign` package provides functions to import data from files associated with SAS, SPSS, Minitab and more.

```r
#Read from a csv-file with column names in a header
read.csv("C:\\Users\\Bingo\\Documents\\data1.csv", header = T)

#Load readxl package and read Sheet 1 in excel file data1
library(readxl) #Load package readxl
read_excel("C:\\Users\\Bingo\\Documents\\data1.xlsx",
           col_names= TRUE, sheet = "Sheet 1")
```

2.3 Subsetting

Objects can be subset using indices in square bracket `[]`.

```r
a <- 3:7
a

## [1] 3 4 5 6 7

a[2]

## [1] 4

a[2:3]

## [1] 4 5
```

Data frames require specifying a row and a column, separated by a comma. Vectors can be used as indices to extract multiple cells

```
b <- data.frame(var1 = 1:5, var2 = 4:8)
b
```

```
##   var1 var2
## 1    1    4
## 2    2    5
## 3    3    6
## 4    4    7
## 5    5    8
```

```
b[2, 1] #Second row, first column
```

```
## [1] 2
```

```
b[2, ] #Entire second row
```

```
##   var1 var2
## 2    2    5
```

```
b[1:3, 1] #First three elements of first column
```

```
## [1] 1 2 3
```

```
b$var1 #First column using the dollar sign operator
```

```
## [1] 1 2 3 4 5
```

Exercise 2.3 *Select the fourth row of the third column from the data frame created in exercise 2.2.*

Exercise 2.4 *Load the CO2 dataset using* `data("CO2")`*. Print rows* 10 *to* 20 *for all columns. Print the column named Treatment.*

3 Basic Arithmetic

3.1 Vector arithmetic

Single data points follow the standard rules of arithmetic.

```
a <- 4
a + 1
```

```
## [1] 5
```

```
a * 2
```

```
## [1] 8
```

```
a ^ 2
```

```
## [1] 16
```

```
a / 8
```

```
## [1] 0.5
```

Operators on a vector and a single data point calculates the operation for each point in the vector

```
a <- c(4, 7)
a + 1
```

```
## [1] 5 8
```

```
a * 2
```

```
## [1]  8 14
```

```
a ^ 2
```

```
## [1] 16 49
```

```
a / 8
```

```
## [1] 0.500 0.875
```

Operators on vectors calculates the pairwise operation, if the vectors are of equal length.

```
b <- c(1, 2)
a + b
```

```
## [1] 5 9
```

```
a * b
```

```
## [1]  4 14
```

```
a ^ b
```

```
## [1]  4 49
```

```
a / b
```

```
## [1] 4.0 3.5
```

Exercise 3.1 *Create two numerical vectors of length eight. Add the vectors together. Multiply the vectors together.*

Exercise 3.2 *Create two numerical vectors, one of length three and one of length four. Add them together - what happens?*

Exercise 3.3 *Create two numerical vectors, one of length three and one of length nine. Add them together - what happens?*

Exercise 3.4 *What does `1/0` return? And `1/Inf`? What about `0/0`? `Inf/Inf`?*

3.2 Logical operators

Logical operators create a vector of `TRUE` or `FALSE`. Equality is given by `==`, comparisons of magnitude by `>` and `<`, inclusion in another set by `%in%` . Logical vectors can be negated (`TRUE` turns to `FALSE` and vice versa) by `!`.

```
a <- c(4,4,8,9)
a == 4

## [1]  TRUE  TRUE FALSE FALSE

a > 4

## [1] FALSE FALSE  TRUE  TRUE

a %in% c(4,9)

## [1]  TRUE  TRUE FALSE  TRUE

b <- a == 4
b

## [1]  TRUE  TRUE FALSE FALSE

!b

## [1] FALSE FALSE  TRUE  TRUE
```

Logical vectors can be used to extract subsets.

```
b <- data.frame(var1 = c("A", "A", "B", "C"),
                var2 = a)
b

##   var1 var2
## 1    A    4
## 2    A    4
```

```
## 3    B    8
## 4    C    9

b[b$var1 == "A", ]

##   var1 var2
## 1    A    4
## 2    A    4
```

The `sum` function can be used to calculate the number of `TRUE` values. `any` returns `TRUE` if at least one value in a vector is `TRUE` and `all` returns `TRUE` if all values in a vector are `TRUE` The function `which` returns the order numbers of the values which are `TRUE`.

```
a <- c(4,4,8,9)
sum(a == 4)

## [1] 2

any(a >= 9) #>= for greater than or equal

## [1] TRUE

any(a > 9)

## [1] FALSE

all(a < 9)

## [1] FALSE

all(a <= 9) #<= for less than or equal

## [1] TRUE

b <- a %in% c(4,9)
b

## [1]  TRUE  TRUE FALSE  TRUE

which(b)

## [1] 1 2 4
```

Exercise 3.5 *Load the CO2 dataset used in exercise 2.4. Use a logical operator to print rows where the variable Type is Quebec. Use a logical operator to find all rows where the variable uptake is greater than* 40.

Exercise 3.6 *Create a vector of the first* 300 *multiples of* 3*, i.e.* $(3, 6, 9, ..., 900)$*, and the* 180 *first multiples of* 5*, i.e.* $(5, 10, 15, ..., 900)$*. How many values in the first vector are also in the second vector? What are the order numbers of those values? How many in the second vector are also in the first? What are their order numbers?*

Exercise 3.7 *Redo exercise 3.6, but with the* 60 *first multiples of* 6 *and the* 90 *first multiples of* 4*. How often is a cell in the first vector also in the second vector? And vice versa?*

Exercise 3.8 *Take the name of an SLU department and turn it into a vector of letters. Which letter is the most common? At which positions is the letter B? Hint: the function* `substring` *can be used to split a string of letters into single cells, e.q.* `substring("SLU", 1:3, 1:3)`*.*

Exercise 3.9 *The single letters* `T` *and* `F` *are short-hand for* `TRUE` *and* `FALSE`*. What is the result of* `TRUE <- 5`*? What is the result of* `T <- 5`*? What about* `assign("TRUE", 5)` *followed by* `get("TRUE")`*?*

4 Plotting

Some examples of plotting functions, using randomly generated data (more on random numbers in section 8).

```
par(mfrow = c(3, 2)) #Print plots in a 3x2 grid
x <- rnorm(100) #100 random draws from standard normal
y <- cos(3 * x) #Creates y as cosine of 3 times x

#Scatterplot
plot(x, y)

#Line plot, x and y both in order of x
plot(x[order(x)], y[order(x)], type = "l")

#Histogram of x
hist(x)

#Probability histogram of x (note change of y-axis)
hist(x, prob = T)
#Added density function for normal distribution
lines(seq(-5, 5, 0.01), dnorm(seq(-5, 5, 0.01)))

#Create new data with x nominal, y continuous
a <- data.frame(x = sample(c("A","B","C"), 100, T),
                y = rnorm(100))
```

```r
table(a$x) #Gives the number of obs in each class

#Barplot
barplot(table(a$x), names.arg = names(table(a$x)),
        ylab = "Number of observations")

#Boxplot
boxplot(y ~ x, data = a, boxwex = 0.25)
```

Exercise 4.1 *Using the CO2 dataset from exercise 2.4, create a histogram of the variable uptake. Use the information from the help page,* `?hist`*, to change the main title and the labels for the x and y axes.*

Exercise 4.2 *Using the CO2 dataset, create a boxplot of uptake seperated by the variables Type and Treatment.*

Exercise 4.3 *Using the CO2 dataset, create a scatterplot with the variable uptake on the y axis and the variable conc on the x axis. Make the observations for Type Quebec a different color from those of Type Mississippi.*

5 Functions

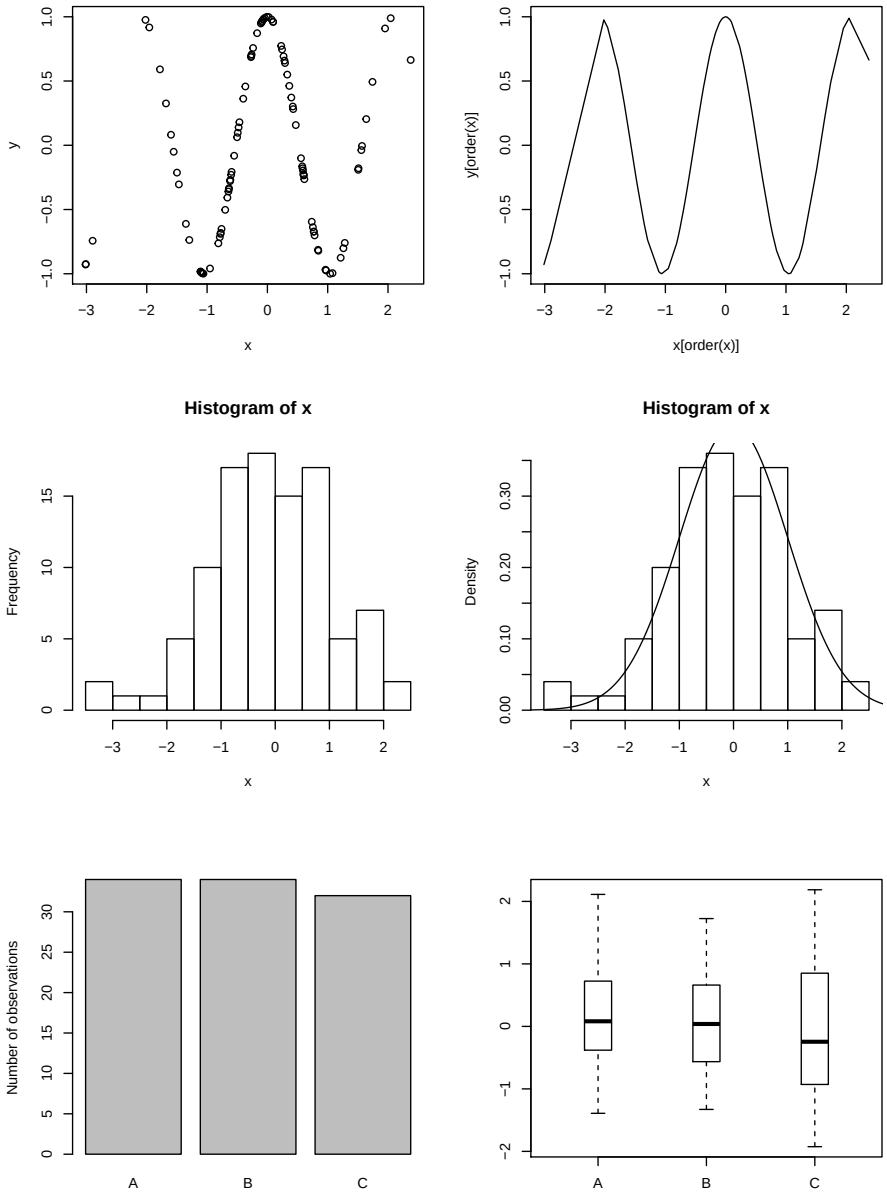
User-defined functions are constructed using the `function` function (*sigh*). The value of the final line of the function gives the output of the function.

```r
f <- function(x){
  y <- x + 2
  y
}

f(3)

## [1] 5

f(c(3, 4)) #The function f acts on both cells in the input vector.

## [1] 5 6
```

The last call, with a vector as an input, works because the operations in the specific function `f` work with a vector.

```r
c(3, 4) + 2

## [1] 5 6
```

Functions can have multiple objects as input and return any object.


y
x
y[order(x)]
x[order(x)]
Histogram of x
Frequency
Histogram of x
Density
Number of observations
A
B
C

```r
g <- function(x1, x2){
  y <- data.frame(var1 = c("A", "B"),
                  var2 = c(x1, x2))
  y
}

g(4, 6)

##   var1 var2
## 1    A    4
## 2    B    6
```

Exercise 5.1 *Write a function which takes a vector and returns the second-last element. Note: the* `length` *function returns the number of elements in a vector.*

Exercise 5.2 *Given two numerical vectors, u and v, of equal length n, the squared Euclidean distance is defined by*

$$d(v,u) = \sum_{i=1}^{n} (u_i - v_i)^2. \tag{1}$$

For example, the distance between the vectors $v = (1,3,5)$ and $u = (6,3,9)$ is given by

$$\begin{aligned} d(v,u) &= (1-6)^2 + (3-3)^2 + (5-9)^2 \\ &= 5^2 + 0^2 + 4^2 \\ &= 25 + 0 + 16 \\ &= 41. \end{aligned} \tag{2}$$

Write a function which takes two equal length vectors as input and returns the squared Euclidean distance.

Exercise 5.3 *A point on the plane can be described as two numbers in a vector of length two (x,y), e.g. $(4,2)$ is the point which is four on the x axis and two on the y axis. Create a function which takes two points in the plane and returns the point which is half-way between the two points. For example, the inputs $(4,2)$ and $(2,8)$ should return $(3,5)$.*

6 If-statements

Operations can be made conditional using an if-statement.

```r
f <- function(x){
  if(x > 4) {print("Greater than four")} else{
    print("Less than or equal to four")
  }
}

f(3)

## [1] "Less than or equal to four"

f(5)

## [1] "Greater than four"
```

Exercise 6.1 *Create a function which takes a single numerical value as input and returns* `"Even"` *if the input is even and* `"Odd"` *if the input is odd.*

7 For-loops

For-loops can be used to reproduce a set of code multiple times, cycling through the input.

```r
a <- c(3, 2, 5)
for(i in a){
  print(i + 1)
}

## [1] 4
## [1] 3
## [1] 6
```

The code in the loop does not have to depend on current value in the loop.

```r
for(i in a){
  print(1)
}

## [1] 1
## [1] 1
## [1] 1
```

For-loops can be used to calculate recursive sequences. Take for example the Fibonacchi sequence defined by

$$\begin{aligned} a_0 &= a_1 = 1 \\ a_i &= a_{i-1} + a_{i-2} \qquad i \geq 2, \end{aligned}$$

and beginning $(1, 1, 2, 3, 5, 8, 13, 21, 34, ...)$.

```
n <- 10 #The length of the sequence
a <- numeric() #Sets up an empty numerical vector
a[1] <- a[2] <- 1 #First two cells are 1
for(i in 3:n){
  a[i] <- a[i-1] + a[i-2] #Next cell is sum of preceeding two
}

a
```

```
##  [1]  1  1  2  3  5  8 13 21 34 55
```

Exercise 7.1 *Create a for-loop which loops over the letters of the alphabet (available as* `LETTERS`*) and for each step of the loop adds the new letter to the preceeding letters (so that, for example, the third step is* `"ABC"`*) and prints the string of letters. Hint:* `paste` *can be used to concatenate two character strings.*

Exercise 7.2 *Create a function which takes a natural number n as input and returns the nth Fibonacchi number.*

Exercise 7.3 *Find the smallest Fibonacchi number greater than* 1000000*.*

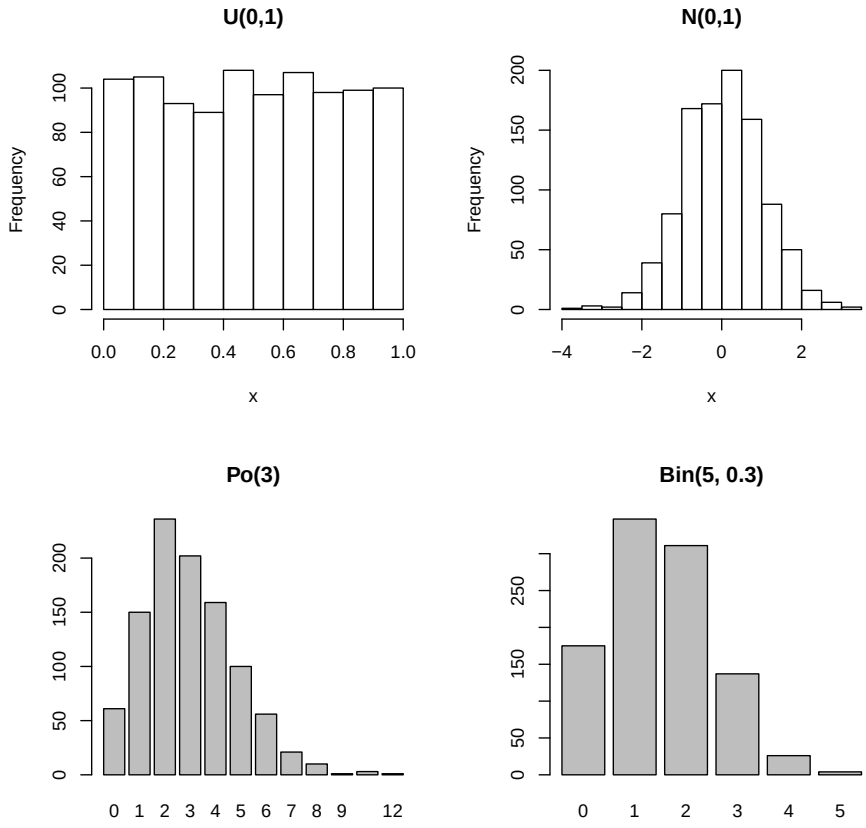
Exercise 7.4 *What happens to the ratio between two Fibonacchi number, a_i/a_{i-1}, as i gets larger? Does it converge to some specific number? If so, what is the square of the atomic number of the element associated with that number?*

8 Pseudo-random Number Generation

Pseudo-random numbers are numbers created using a deterministic process, intended to emulate completely random numbers. R includes functions to draw numbers from a large number of different distributions.

```
par(mfrow = c(2,2))

#Uniform distribution, a = 0, b = 1
x <- runif(1000)
hist(x, main = "U(0,1)")

#Normal distribution, mean = 0, sd = 1
x <- rnorm(1000)
hist(x, main = "N(0,1)")

#Poisson distribution, lambda = 3
x <- rpois(1000, lambda = 3)
```

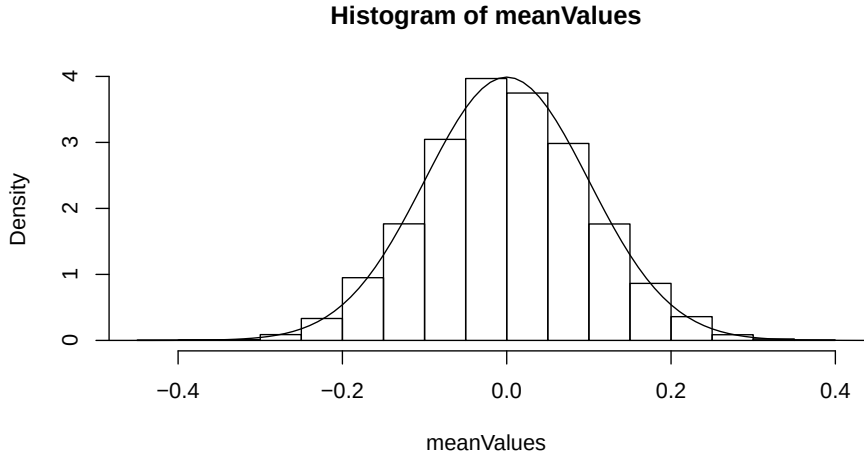
```r
barplot(table(x), main = "Po(3)")

#Binomial distribution, n = 5, p = 0.3
x <- rbinom(1000, size = 5, prob = 0.3)
barplot(table(x), main = "Bin(5, 0.3)")
```



Random numbers can be used to test if methods and functions work as intended. As an example, we investigate the distribution of a mean value. Based on theory, one expects the mean value of 100 observations from a standard normal distribution to be normal with mean $\mu = 0$ and standard deviation $\sigma = 0.1$. To test this, we simulate the situation multiple times and store the mean values.

```r
meanValues <- numeric()
for(i in 1:10000){
  x <- rnorm(100, mean = 0, sd = 1)
  meanValues[i] <- mean(x)
```

```
}

mean(meanValues)

## [1] -0.0007930347

sd(meanValues)

## [1] 0.09975179

par(mfrow = c(1,1))
hist(meanValues, prob = T)
lines(seq(-1, 1, 0.01), dnorm(seq(-1, 1, 0.01),
                              mean = 0, sd = 0.1))
```



The simulated result supports the theoretical calculation.

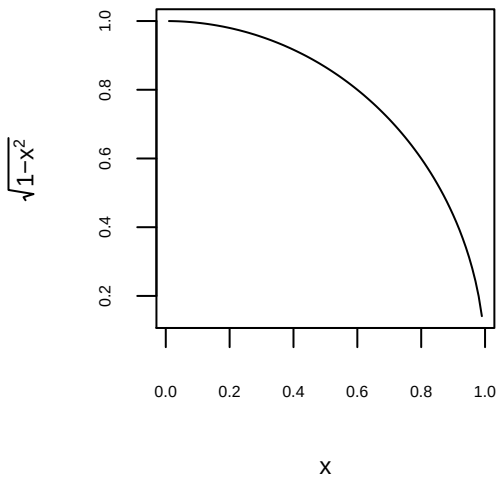
Another common use of random numbers is to evaluate integrals. Recall the intuitive understanding of an integral as an area under a curve. If the curve is contained in a rectangle, and random points simulated in the rectangle, the area under the curve can be estimated as the proportion of points that are below the curve times the area of the rectangle.

For example, take

$$f(x) = \sqrt{1 - x^2} \qquad x \in (0, 1), \tag{3}$$

the curve which gives the upper right part of the unit circle.

```
x <- seq(0.01, 0.99, 0.01)
plot(x, sqrt(1 - x^2), type = "l", cex.axis = 0.5,
     cex.lab = 0.75, ylab = expression(sqrt("1-x"^2)))
```



We simulate from the 1×1 rectangle with base $(0, 0)$ and check if the random number y if smaller than $\sqrt{1 - x^2}$ for the random number x. The area of the rectangle is 1, so the estimate of the integral is given directly by the proportion of points below the curve.

```
underTheCurve <- numeric()
for(i in 1:10000){
  x <- runif(1)
  y <- runif(1)
  yUnderCurve <- y < sqrt(1 - x^2)
  underTheCurve[i] <- yUnderCurve
}
integralEstimate <- mean(underTheCurve)
integralEstimate

## [1] 0.7936
```

Note that the mean of a logical vector returns the proportion of cells with value `TRUE`.

Since our integral is one fourth of the unit circle, and the unit circle should have an area given by π, four times our integral estimate gives an estimate of the constant $\pi \approx 3.14$.

```
4 * integralEstimate
```

```
## [1] 3.1744
```

```
pi
```

```
## [1] 3.141593
```

Exercise 8.1 *Simulate* 10 *numbers from a normal distribution with mean* 0 *and standard deviation* 1*. What is the maximum value? Repeat with* 100 *numbers,* 1000 *numbers, and so on, until you are bored.*

Exercise 8.2 *Use the simulation approach to evaluate the area under the curve* $\sqrt{x+1}$ *for* x *in the interval* $(0, 2)$*, i.e.* x *between* 0 *and* 2*. The solution should be close to* 2.80*.*

Exercise 8.3 *Use the simulation approach to evaluate the area under the curve*

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \tag{4}$$

for x *in the interval* $(-1.96, 1.96)$*. Note that the curve is the density function of a standard normal distribution - what is the probability of being between* -1.96 *and* 1.96*?*

Exercise 8.4 *Draw* 1000 *numbers from a Poisson distribution with* $\lambda = 3$ *and* 1000 *numbers from a binomial distribution with* $n = 100$ *and* $p = 0.03$*. Plot the random draws using two bar charts. Are the plots similar?*

Exercise 8.5 *Theory states that a binomial distribution with some* n *and a probability of* $3/n$ *approaches a Poisson distribution with* $\lambda = 3$ *as* n *increases. Examine if this is true by simulating* 1000 *random numbers from a binomial with* n *equal to* 10*,* 100*,* 1000 *and* 10000 *and comparing to a distribution of random numbers from the Poisson distribution.*

Exercise 8.6 *Create a vector of points in the two-dimensional plane using the following scheme.*

1. *Let* $a = (0, 0)$*,* $b = (1, 1)$ *and* $c = (2, 0)$*.*
2. *Let* $d_0 = (1, 0.5)$*.*
3. *Given* d_{i-1}*, calculate* d_i *as the point half-way between* d_{i-1} *and a random choice of* a*,* b *and* c*.*
4. *Repeat the previous step until you reach* d_{1000}*.*
5. *Plot the set of points.*

Hint: in exercise 5.3 you were asked to write a function which takes two points in the plane and returns the point between the two points.