2019 Class Test 2: Honours

Task A: Formula 1

Formula One (F1) is the highest class of single-seater auto racing sanctioned by the Federation Internationale de l'Automobile (FIA). Drivers race for teams knowns as 'constructors' at a series of races held around the globe. At each race drivers are awarded points according to the position they finish in (the driver finishing in position 1 receives the most points etc…).

Questions 1 to 5 relate to the dataframe `f1`. The dataframe `f1` contains one row for every driver who scored points in any race held during the period 1956 to 2017. It consists of the following columns;

f1	
`driver`	Name of the driver
`position`	Race position the driver finished in
`points`	Number of points the driver received
`year`	Year the race was held
`gpname`	Name of the race
`constructor`	Name of the constructors team the driver raced for in this race
`nationality`	Nationality of the driver

Set your working directory to the location where your files are stored and enter the line of code below to read the `f1` dataframe into `R`.

```r
load("f1.RData")
```

Q1. [**2 marks**] Sort the rows of the `f1` dataframe in ascending order according to the `year` column. The ordered dataframe should be called `f1`

```r
f1 <- f1[order(f1$year),]
```

Q2. [**2 marks**] Define a variable called `italian` which contains the number of times the `ferrari` Constructors team has come first (i.e. `position`==1) in any race.

```r
italian <- sum(f1$position==1&f1$constructors=="ferrari")
```

Q3. [**3 marks**] Create a column in the `f1` dataframe called `podium`. The `podium` column should contain the value `TRUE` if the value of `position` in the corresponding row is 1,2, or 3 and `FALSE` otherwise.

```r
f1 <- transform(f1, podium=ifelse(f1$position<=3, TRUE, FALSE))
```

Q4. [**2 marks**] Define a variable called `austria` which contains the average number of points per race for all drivers listed as having Austrian nationality.

```r
austria <- mean(f1$points[f1$nationality=="austrian"])
```

The question on the printed paper said Australian rather than Austrian, marks were awarded if the austrial nationality was used rather than Austrian.

Q5. [**5 marks**] Define the following two objects

- a vector called `champions` which contains the names of the Constructors teams who won the most points each year (from 1956 to 2017). In other words the first element of `champions` should contain the name of the Constructors team with the most points in 1956 etc …
- a vector of length 1 called `winner` which contains the name of the team who appears most often in the vector `champions`.

```
byyear <- split(f1, f1$year)
win <- function(x){
a <- which.max(unlist(by(x$points, x$constructor, sum)))
  names(a)
}
champions <- sapply(byyear, win)
winner <- names(which.max(table(champions)))
```

Task B

Q6. [**5 marks**] The general formula for the vertices of a t-sided regular polygon, (x_i, y_i), $i = 1, ..., t$, radius r,can be written as

$$x_i = r \times cos\left(\frac{2\pi i}{t}\right) + c_1$$

$$y_i = r \times sin\left(\frac{2\pi i}{t}\right) + c_2$$

where r is the radius of the polygon and (c_1, c_2) is the centre of the polygon.

The question originally had addition instead of multiplication signs; marks were awarded if either was used here Write a function `reg.polygon` which takes the arguments `t`, `r` and `centre` and both draws a regular polygon and returns the co-ordinates of the vertices of that polygon (as a `t` $\times$ 2 matrix) .

Your function should. . .

- have default values for the arguments that mean the function draws a hexagon with radius 2 and is centred at $(0, 0)$ by default.
- be able to accept additional arguments which will control the graphical parameters of the polygon drawn, such as the color and line width etc.
- produce a warning message `"t must be at least 3"` if the value of t supplied to the function is ≤ 3.

```
reg.polygon <- function(t=6, r=2, centre=c(0,0), ...)
  {
  if (t<3) warning("t must be greater than 3")
  x = r * cos(2*pi*((1:t)/t)) + centre[1]
  y = r * sin(2*pi*((1:t)/t)) + centre[2]
  mm <- rbind(cbind(x, y), c(x[1], y[1]))
  lines(mm, ...)
  return(cbind(x,y))
}
```

Q7. [**4 marks**] Use the following code to create a blank drawing canvas

```
plot(c(-2,2), c(-2,2), type="n")
```

Now use your function `reg.polygon` from Question 6 to draw two hexagons, both centred at $(0, 0)$, one of radius 1 and one of radius 2.

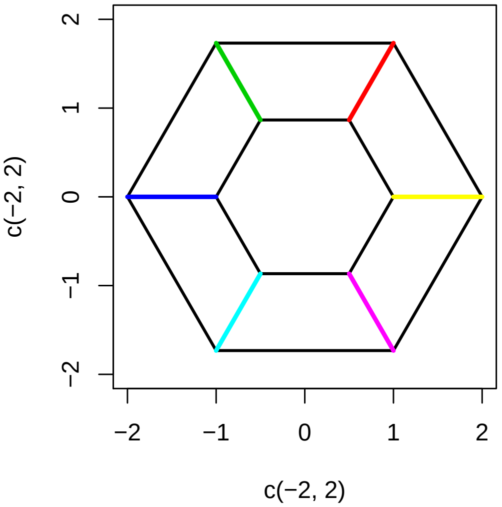
Next draw lines from all of the vertices of the outer hexagon to the corresponding vertices of the inner hexagon. Your plot should look like the one shown below. The line joining each pair of vertices should be drawn in a different colour.

Note: *If you have not successfully written the function* `reg.polygon` *then you can run the following line of code to read in two matrices containing the co-ordinates of the vertices of two hexagons* `hex1` *(co-ordinates of a hexagon with radius 1) and* `hex2` *(co-ordinates of a hexagon with radius 2);*

```
load("hexagons.RData")
```

This code assumes your working directory is set to where the file is stored.

```
par(pty="s")
plot(c(-2,2), c(-2,2), type="n")
ab <- reg.polygon(6,  lwd=2)
cd <- reg.polygon(6, lwd=2, r=1)
n <- nrow(ab)
for(i in 1:n){
      lines(c(ab[i,1], cd[i,1]), c(ab[i,2],cd[i,2]), lwd=3, col=i+1)
}
```



Task C

Q8. [**3 marks**] The unit disc is the circle with radius 1 and centre $(0, 0)$. Write a function `within` which takes as a single argument a vector of length 2, `u=c(u1, u2)`. Your function `within` should return `TRUE` if the vector `u =c(u1, u2)` represents a point which lies within the unit disc and `FALSE` otherwise.

Hint: a point (x, y) lies within the unit circle if $\sqrt{x^2 + y^2} < 1$.

Use the code below to define a 2000×2 matrix called `mat` where each row contains a pair (u_1, u_2) randomly drawn from a uniform distribution on the interval $[-1, 1]$.

```
set.seed(123)
mat <- matrix(runif(4000,-1,1),2000,2)
```

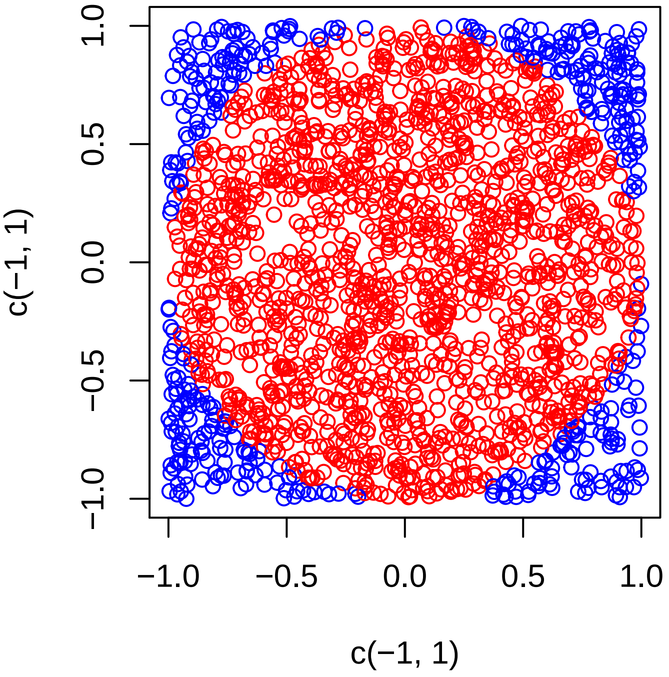
Q9 [**4 marks**] Use the following code to set up an empty plotting canvas;

```
plot(c(-1,1), c(-1,1), type="n")
```

Plot on this plotting canvas the points in `mat` (as defined above). Colour the points which lie within the unit disc in red and those outside the unit circle in blue.

Your plot should look similar to the one below.

```r
par(pty="s")
cols <- apply(mat, 1, within)
plot(c(-1,1), c(-1,1), type="n")
points(mat, col=c(4,2)[unclass(cols)+1])
```



In this task you will implement a method for drawing (pseudo-)random realisations from the normal distribution. No marks will be awarded for Question 10 or Question 11 if the function `rnorm` is used within them.

Q10 [**6 marks**] Write a function `marsaglia.polar` which takes no arguments and which returns a vector of length 2 containing a pair of numbers z_1 and z_2 which can be calculated as follows

1. Carry out steps (i) and (ii) until $s < 1$.
   (i) Use the `R` function `runif` to draw a pair of independent random numbers u_1 and u_2 from the uniform distribution on the interval $[-1, 1]$.
   (ii) Compute $s = \sqrt{u_1^2 + u_2^2}$
2. Compute $\rho = \sqrt{\frac{-2 \log s}{s}}$
3. Compute $z_1 = \rho u_1$ and $z_2 = \rho u_2$

Note that z_1 and z_2 are then random draws from a standard normal distribution.

```r
marsaglia.polar <- function() {
  while (TRUE) {
  u <- runif(2, -1, 1)
  s <- sum(u^2)
  test <- within(u)
  if (test) break}
    rho <- sqrt(-2*log(s)/s)
    rho*u
    }
```

Q11. [**3 marks**] One can show that if $Z \sim N(0,1)$ then

$$X = \mu + \sigma, \quad Z \sim N(\mu, \sigma^2)$$

We can exploit this to draw random numbers from a normal distribution with any mean and standard deviation. Write a function called `normal.pair` which takes μ and σ as arguments. Your function `normal.pair` should;

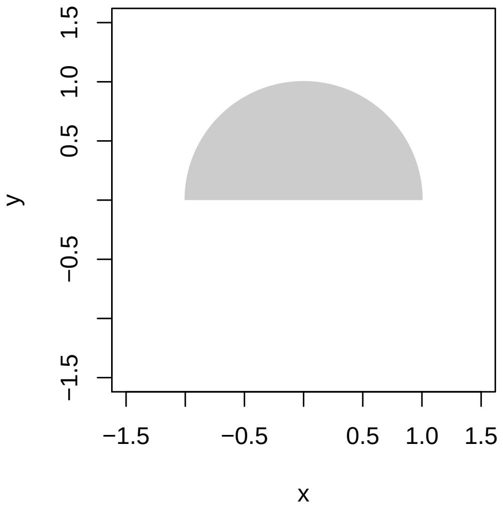
- draw a pair (z_1, z_2) from the standard normal distribution using the `marsaglia.polar` method you have written in Question 10,
- return a pair of numbers $(\mu + \sigma z_1, \mu + \sigma z_2)$ calculated using the formula above,
- check whether σ is negative and return an error message in that case.

Q12 [**3 marks**] Use your function `normal.pair` to draw 1000 normal pairs (x_i, y_i) $i = 1, ..., 1000$ where each pair comes from a normal distribution with $\mu = -0.25$ and $\sigma = 0.5$, i.e.

$$(x_i, y_i) \sim N(-0.25, 0.5)$$

Use the 1000 pairs you have simulated to define a variable `prob` which contains the probability that a pair from a $N(-0.25, 0.5)$ distribution respresents a point which lies within the upper half of the unit disc (as defined above).

The upper half of the unit disc is depicted by the grey shaded region in the plot below.



```
xy <- matrix(0,1000,2)
for (i in 1:1000) xy[i,] <- normal.pair(-0.25,0.5)
prob <- sum((sqrt(rowSums(xy^2)) < 1) & xy[,2]>0)/1000
```

Note: *If you have not managed to successfully complete Question 11 you may use the code below to define a function* `normal.alt` *which can be used as a substitute for* `normal.pair` *defined in Question 11.*

```r
normal.alt <- function(mu, sigma){
  rnorm(2, mean=mu, sd=sigma)
}
```