23847-Lab-3

23847

2024-03-19

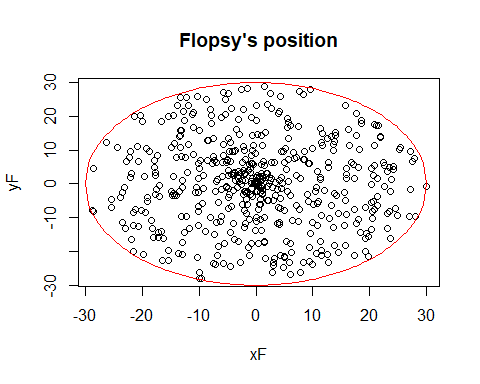
GenAI has not been used in the creation of any content within this document

## Problem 1

### a)

We know ΘF ∼ Uniform(0, 2π) and RF ∼ Uniform(0, 30). Firstly we shall create two vectors, one for ΘF (thetaF) and one for rF, using the runif distribution. Next we use the given formula to convert from polar to cartesian coordinates which can then be plotted using the plot function. Finally a perimeter circle can be plotted using the add\_circle function with radius 30, this being the maximum of the unif distribution.

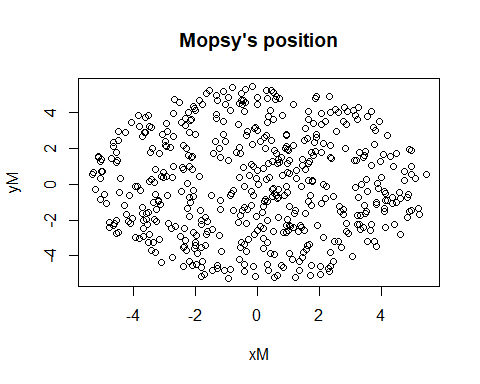
n = 500  
  
thetaF = runif(n, 0, 2\*pi)  
rF = runif(n, 0, 30)  
  
xF = rF \* cos(thetaF)  
yF = rF \* sin(thetaF)  
  
plot(xF, yF, main = "Flopsy's position")  
  
  
add\_circle=function(radius)  
{  
 xcoord=(radius/50)\*c(-50:50)  
 ycoord\_pos=sqrt(radius^2-xcoord^2)  
 ycoord\_neg=-sqrt(radius^2-xcoord^2)  
 points(xcoord,ycoord\_pos,type='l',col='red')  
 points(xcoord,ycoord\_neg,type='l',col='red')  
}  
add\_circle(30)



### b)

Defining xM and yM as two empty vectors, we can then execute the loop until the length of xM (or yM) reaches 500. as xM, yM ~ Unif(-30, 30) we can use the runif function to create a vector of these two realizations. Then we check if their distance from the origin is within the radius 30 circle. If they are we append the coordinates to the xM and yM vectors respectively, and if not nothing happens. Once the vectors are filled we can then plot the x and y coordinates respectively.

xM = c()  
yM = c()  
while(length(xM) < 500){  
 XY = runif(2, -30, 30)  
 if(XY[1]^2 + XY[2]^2 <= 30){  
 xM = append(xM, XY[1])  
 yM = append(yM, XY[2])  
 }  
}  
plot(xM, yM, main = "Mopsy's position")



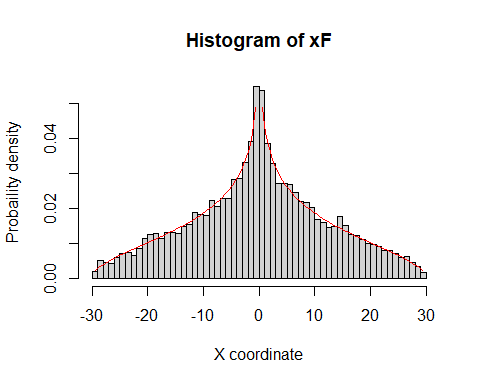
### c)

It is noticeable that Flopsy’s positions are more centered round the origin whereas Mopsy’s are more spread, having a lot more points near the boundary. This could be because Flopsy’s Cartesian coordinates are not independent, both relying on r and theta.

### d)

Using the code from 1a) we can change n to 10000 and generate 10000 realizations. Using the hist functions with 60 breaks we can see the density is much greater around x = 0. Overlaying the pdf\_f function onto the histogram we can see it agrees however, around the origin the curve disappears. This will be due to the fact it is undefined at x=0, from the log(abs(x)) component of the function.

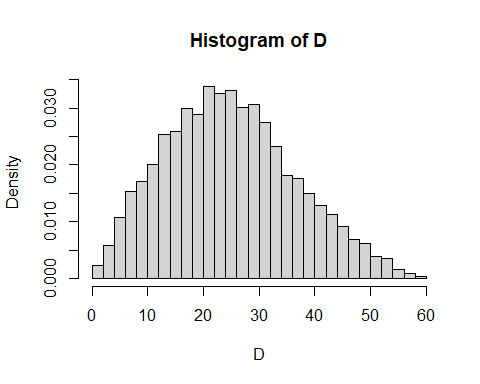
n = 10000  
thetaF = runif(n, 0, 2\*pi)  
rF = runif(n, 0, 30)  
  
xF = rF \* cos(thetaF)  
yF = rF \* sin(thetaF)  
  
hist(xF, breaks = 60, prob = TRUE, xlab = "X coordinate", ylab = "Probaility density")  
  
pdf\_f=function(x)  
{  
 fx= (1/pi)\*(1/sqrt(30^2-x^2)) +  
 (1/(30\*pi))\*log(30+sqrt(30^2-x^2)) +  
 (-1/(30\*pi))\*log(abs(x)) +  
 (-1/(30\*pi)) +  
 (-x^2/(30\*pi))\*(1/(30+sqrt(30^2-x^2)))\*  
 (1/sqrt(30^2-x^2))  
 fx  
}  
curve(pdf\_f, add = TRUE, col = 'red')



### e)

Albeit inefficient, we will once again copy code from a previous question, this time taking code from 1a) for Flopsy’s positions and 1b for Mopsy’s. Then we can use a for loop to calculate the distance between the two sample points using Pythagoras’s theorem, storing them in a vector, D. Finally the points can be plotted on a histogram and the estimated value est can be calculated as the mean of the distances, giving est ~= 15.5 to 3s.f

n = 10000  
# Flopsy  
thetaF = runif(n, 0, 2\*pi)  
rF = runif(n, 0, 30)  
  
xF = rF \* cos(thetaF)  
yF = rF \* sin(thetaF)  
  
# Mopsy  
xM = c()  
yM = c()  
while(length(xM) < n){  
 XY = runif(2, -30, 30)  
 if(XY[1]^2 + XY[2]^2 <= 30^2){  
 xM = append(xM, XY[1])  
 yM = append(yM, XY[2])  
 }  
}  
D = numeric(n)  
for(i in 1:n){  
 D[i] = sqrt((xF[i] - xM[i])^2 + (yF[i] - yM[i])^2)  
}  
  
hist(D, breaks = 30, prob = TRUE)



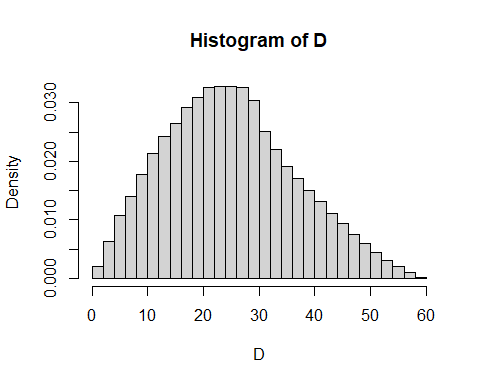
est = sum(D) / n

### f)

For i. we can create an empty vector, Z and then update the index if the statement, here being event E, is true. For ii. we once again take the sum of the vector over the number of trials to give an estimate for p\_E. For iii. we can use the sdphat function from exercise 1c) with p = p\_E from part ii) and n = 10000. For iv. assume

running the computation with this chosen value of n gives an estimation of 0.2177

# i.  
Z = rep(c(0), n)  
for(i in 1:n){  
 if(D[i] < 15){  
 Z[i] = 1  
 }  
}  
  
# ii.  
p\_E = sum(Z) / n  
  
# iii.  
sdphat = function(p, n){  
 return (sqrt(p \* (1 - p) / n))  
}  
  
sdev = sdphat(p\_E, n)  
  
# iv.  
n = 172000  
# Flopsy  
thetaF = runif(n, 0, 2\*pi)  
rF = runif(n, 0, 30)  
  
xF = rF \* cos(thetaF)  
yF = rF \* sin(thetaF)  
  
# Mopsy  
xM = c()  
yM = c()  
while(length(xM) < n){  
 XY = runif(2, -30, 30)  
 if(XY[1]^2 + XY[2]^2 <= 30^2){  
 xM = append(xM, XY[1])  
 yM = append(yM, XY[2])  
 }  
}  
D = numeric(n)  
for(i in 1:n){  
 D[i] = sqrt((xF[i] - xM[i])^2 + (yF[i] - yM[i])^2)  
}  
  
hist(D, breaks = 30, prob = TRUE)



est = sum(D) / n  
Z = rep(c(0), n)  
for(i in 1:n){  
 if(D[i] < 15){  
 Z[i] = 1  
 }  
}  
  
p\_E = sum(Z) / n