1. Simulate the following problem: choose \( n \) points on a circle and find the probability that they all lie in some semi-circle. Hint: draw \( n \) times from the uniform pdf \( \text{runif}(n, 0, 2\pi) \). Additionally, let \( X \) be the angle of the smallest arc containing all the points. Plot a histogram of \( X \) — that is, estimate the pdf \( f_X(x) \). (You may find the function \( \text{sort} \) useful).

```R
### TRIAL = n points on a circle
trial <- function(n){
  return (runif(n,0,2*pi))
}

### calculate the smallest angle subtending all points
min.angle <- function(t){
  # input: a trial of n points on circle
  # outputs the smallest sized sector (in radians) of the
  # circle that contains all the points.
  v <- sort(t)
  w <- c(v[-1], 2*pi + v[1])
  angs <- (w - v)
  m.angle <- 2*pi - max(angs)
  return(m.angle)
}

### in.circle = all points lie in a semi-circle
in.semi.circ <- function(t){
  # input: a trial of n points on circle
  # outputs TRUE if all points lie in a semicircle.
  return(min.angle(t) < pi)
}

## histogram / mean of minimum angle

distr.min.angle <- function(num pts, reps){
  v <- replicate(reps, min.angle(trial(num pts)))
  br <- seq(0,2*pi, .1)
  title <- paste("mean distance:", mean(v))
  hist(v, breaks=br, col="blue", main= title, xlab="min angles (Radians)", prob=TRUE)
}

## plots distributions for different numbers of sample points
plot.distrib <- function(reps){
  par(mfrow=c(3,3))
  v <- c(2,3,4,5,6,8,10,12,15)
  for(k in v){
    distr.min.angle(k, reps)
  }
}
```
## relative frequency of "n random points lying in a semi-circle"

rel.freq <- function(num pts, reps){
  # relative freq of n points sitting in a semicircle
  exper <- replicate(reps, in.semi.circ(trial(num pts)))
  return(mean(exper))
}

## Plot relative frequencies for different sample sizes
rf.plot <- function(reps){
  v <- numeric(20)
  for (k in 1:20) { v[k] <- rel.freq(k, reps) }
  par(mfrow=c(1,1))
  title <- "relative frequency of n points lying in a semicircle"
  xvar <- "number of sample points n"
  plot(v, main=title, xlab=xvar, ylab="probability")
  lines(v)
}

# The rest is dedicated to visually depicting the points on a circle

min.max <- function(t){
  # input a trial t
  # outputs the two consecutive points on circle that have max angle between them.
  v <- sort(t)
  w <- c(v[-1], 2*pi + v[1])
  angs <- (w - v)
  k <- which(angs == max(angs))
  if( k != length(t)) {
    mm <- c(v[k], v[k + 1])
  } else {
    mm <- c(v[k], v[1])
  }
  return(mm)
}

grtrial <- function(v){
  n <- length(v)
  if( in.semi.circ(v) == TRUE) { s <- "fits in a semicircle"} else { s <- "doesn't fit in a semicircle"}
  mina <- paste("angle = ", min.angle(v))
  plot(cos(v),sin(v),xlim=c(-1,1),ylim=c(-1,1),col="red", main=s, ylab="",
       xlab=mina)
  curve(sqrt(1-x^2),add=T)
  curve(-sqrt(1-x^2),add=T)
  for (k in 1:n){
    if( cos(v[k]) < 0 ){
      bmin<-cos(v[k])
      bmax<-0
    } else {
      bmin<-0
      bmax<-cos(v[k])
    }
    curve(x*sin(v[k])/cos(v[k]),from=bmin, to=bmax, add=T)
  }
}
### Attempt #2: using shapes package

```r
plot.trials <- function (num.pts)
    {par(mfrow=c(3,3))
     for(k in 1:9) { gtrial(trial(num.pts)) }
    }

plot.trials(4)
```

**Graphical output:** $n$-points in a semicircle.

The probability 4 points chosen on a circle is 50%, as predicted by problem #14. The red points are plotted from the simulation, while the curve is plotted using the exact answer in problem #14.
Below is a sample of 9 trials, drawing $n = 4$ points on a circle. The smallest angle containing all 4 points is listed below each circle. You can generate plots like this for different $n$ using the function `plot.trials(n)`.

Below is a sample of 9 trials, drawing $n = 4$ points on a circle. The smallest angle containing all 4 points is listed below each circle. You can generate plots like this for different $n$ using the function `plot.trials(n)`.
Plots of histograms for the minimum angle containing $n$ points picked on a circle. Notice as $n$ becomes large, the histogram concentrates itself nearer to $2\pi$. 
2. (a) Pick a few distributions (`rnorm()`, `rpois()`, `rexp()`...) with parameters of your choice. Sample two independent random variables from the same distribution and look at a histogram of their sum. (See the example below). Sketch a drawing of the histograms $X, Y, X + Y$ on your HW.

(b) Let $X$ and $Y$ be independent standard normal random variables. Plot $(X, Y)$ and look at the distances. Then, make a histogram of the distance from the origin: $\sqrt{X^2 + Y^2}$.
## For sums of two uniform RVs

## with parameters \([a,b]\) and \([c,d]\).

```R
unif.plot <- function(u){
x <- runif(10000,u[1],u[2])
m <- mean(x)
title <- paste("hist of unif \[", toString(u), "] \)
stitle <- paste("mean = ", toString(m))
hist(x,col="red",prob=TRUE, main=title, sub=stitle, xlab="")
}

unif.sum <- function(u1,u2){
  # u1 = \([a,b]\) is parameter of uniform rv1
  # u2 = \([c,d]\) is parameter of uniform rv2
  x <- runif(10000,u1[1],u1[2])
y <- runif(10000,u2[1],u2[2])
s <- x+y
  m <- mean(s)
title <- paste("sum of unif \[", toString(u1), "] and \[", toString(u2), "]\)
stitle <- paste("mean = ", toString(m))
hist(s,col="yellow",prob=TRUE, main=title, sub=stitle, xlab="")
}

sample.unif <- function(){
  par(mfrow=c(3,3))
  for( k in 1:3 ){
    u1=sort( sample(0:10,2) )
    u2=sort( sample(0:10,2) )
    unif.plot(u1)
    unif.plot(u2)
    unif.sum(u1,u2)
  }
}
```

## For sums of two poisson RVs with

## parameters \(\lambda_1,\lambda_2\).

```R
poisson.plot <-function(lambda){
x <- rpois(10000,lambda)
m <- mean(x)
title <- paste("hist of poisson", toString(lambda))
stitle <- paste("mean = ", toString(m))
hist(x,col="red",prob=TRUE, main=title, sub=stitle, xlab="")
}

poisson.sum <- function(lambda1,lambda2){
x <- rpois(10000,lambda1)
y <- rpois(10000,lambda2)
s<- x+y
```
```r
m <- mean(s)
title <- paste("sum of poissons", toString(lambda1), "and", toString(lambda2))
stitle <- paste("mean = ", toString(m))
hist(s,col="yellow",prob=TRUE, main=title, sub=stitle, xlab="")
}

sample.poisson <- function(){
  par(mfrow=c(3,3))
  for( k in 1:3 ){
    lambda1=sample(1:10,1)
    lambda2=sample(1:10,1)
    poisson.plot(lambda1)
    poisson.plot(lambda2)
    poisson.sum(lambda1,lambda2)
  }
}

sample.poisson()

## For sums of two exponential RVs with
## waiting times lambda1, lambda2.

exp.plot <- function(lambda){
  x <- rexp(10000,lambda)
  m <- mean(x)
  title <- paste("hist of exp", toString(lambda))
  stitle <- paste("mean = ", toString(m))
  hist(x,col="red",prob=TRUE, main=title, sub=stitle, xlab="")
}

exp.sum <- function(lambda1,lambda2){
  x <- rexp(10000,lambda1)
  y <- rexp(10000,lambda2)
  s <- x+y
  m <- mean(s)
  title <- paste("sum of exp", toString(lambda1), "and", toString(lambda2))
  stitle <- paste("mean = ", toString(m))
  hist(s,col="yellow",prob=TRUE, main=title, sub=stitle, xlab="")
}

sample.exp <- function(){
  par(mfrow=c(3,3))
  for( k in 1:3 ){
    lambda1=sample(1:20,1)
    lambda2=sample(1:20,1)
    exp.plot(lambda1)
    exp.plot(lambda2)
    exp.sum(lambda1,lambda2)
  }
}
```r
### For the RV Z = \sqrt{X^2+Y^2}, that's the
distance from the origin of a point
### (X,Y), where both are normally distributed.

```r
norm2D <- function()
{
  par(mfrow=c(2,2))
  x <- rnorm(10000) # x-coordinate
  y <- rnorm(10000) # y-coordinate
  thin.ind <- sample(1:10000,500) # a sample of the coordinates to plot
  plot(x[thin.ind],y[thin.ind], xlab="x coord",ylab="y coord",main="plot of bivariate std normal in plane")
  d <- sqrt(x^2+y^2)
  m <- mean(d)
  title <- "distance of bivariate std normal to (0,0)"
  stitle <- paste(" mean = ", toString(m) )
  hist(d,col="red",prob=TRUE, main=title, sub=stitle, xlab="")
}

norm2D()
```

Graphical Output: sums of random variables

First two columns are histograms of samples of uniform random variables \( X \) and \( Y \). The 3rd column is the histogram of the sum. The parameters are randomly generated using the function `sample.unif()` . The following pages are similar.
hist of exp 20

hist of exp 11

sum of exp 20 and 11

mean = 0.0504206106347654

mean = 0.0900047682177765

mean = 0.14020796874521

hist of exp 18

hist of exp 3

sum of exp 18 and 3

mean = 0.0562175411196509

mean = 0.326917693687261

mean = 0.387496342133429

hist of exp 13

hist of exp 15

sum of exp 13 and 15

mean = 0.0761693578140454

mean = 0.0675002953478136

mean = 0.1427386412722553
Given two standard normal RVs $X$ and $Y$, the first figure is a plot of pairs $(X,Y)$. The histogram counts the occurrences of values for the distance of $(X,Y)$ to the origin — that is, it’s a histogram of the random variable $\sqrt{X^2 + Y^2}$.

3. Simulate a binomial RV with $n = 100$ and $p = 1/5$. What is the normal pdf that approximates this random variable? Create a histogram of your simulation and graph the normal curve on a single plot.

Binomial_Normal.R

```r
## Approximate Binomial with Normal

trial <- function(){
  # generates 100 flips of a coin w/p=1/5
  # and outputs the number of Heads
  flips <- sample(c(0,0,0,0,1),100,replace=TRUE)
  return( sum( flips ) )
}

exper <- function(n){
  return( replicate(n, trial() ) )
}

norm.pdf <- function(x,mu,sigma){
  exp(-((x-mu)^2)/(2*sigma^2))/(sqrt(2*pi)*sigma)
}

mu = 20
sigma = 4

hist(exper(10000), breaks=seq(-.5,100.5), prob=TRUE, col="blue")
curve(norm.pdf(x,mu,sigma),add=TRUE, col="red")
```
4. Simulate a geometric RV and compare it to the pmf derived in class.

```r
# First, we simulate a geometric random variable below:

gem.trial <- function(p){
  # input likelihood a given trial is success
  # outputs the # of trials until success occurs
  num.trials <- 0
  x <- FALSE
  while(x == FALSE){
    # further code here...
  }
}
```
10 trial <- sample(c(0,1), 1, replace=TRUE, prob=c(1-p,p))
11 num.trials <- num.trials + 1
12 x <- (trial == 1)
13 }
14 return( num.trials )
15 }
16 # Now we will simulate this geometric RV 10000 times, and
17 # record "the number of trials it takes before success"
18 # for p=.2 and record our results in a histogram.
19 p<-.2  #probability trial results in success
20 L <- replicate(10000, geom.trial(p))  # repeated trials
21 br <- (0:max(L)+.5)  # bins for the histogram
22 hist( L ,prob=T, breaks=br, col="blue" )  # prob=T converts "of
23 # Now we will compare this with the formula
24 # for the pmf for a geometric random variable.
25 geom.pmf <- function(k){
26 return( p*(1-p)^(k-1) )
27 }
28 points( geom.pmf(1:max(L)),col="red" )
29 lines( geom.pmf(1:max(L)),col="red" )
5. (Courtesy of T Moore) The list below give the number of fumbles that occurred in every college football game during a given week earlier this season. Find a Poisson random variable $X$ that approximates this sample. Plot both the sample and the pmf $f_X(x)$ of $X$.

List of fumbles: 3 1 2 4 6 2 1 3 3 1 5 5 4 4 3 3 2 1 1 5 2 4 1 3 3 4 3 1 2 1 3 2 2 2 2 2 3 0 2 1 0 0 2 0 4 0 5 2 1 3 2 3 2 5 2 2 4 1 2 4 4 5 1 1 4 1 2 1 6 2 3 2 2 0 7 4 1 1 3 1 2 3 5 2 1 2 2 1 3 1 3 5 4 4 0 1 4 6 1 2 4 0 3 4 1 5 4 3 5

Fumbles.R

1
2

16
```r
fumbles <- c(3, 1, 2, 4, 6, 2, 1, 3, 3, 1, 5, 5, 4, 4, 3, 3, 2, 1, 1, 5, 2, 4, 1, 3, 3, 4, 3, 3, 1, 2, 1, 3, 2, 2, 2, 2, 3, 0, 2, 1, 0, 2, 0, 1, 0, 0, 1, 4, 1, 3, 1, 3, 5, 4, 4, 0, 1, 4, 6, 1, 2, 4, 0, 3, 4, 1, 5, 4, 3, 5)
M <- max(fumbles)
hist(fumbles, breaks=seq(-.5, M + 0.5), prob=TRUE, col="yellow")
poiss.pmf <- function(k, lambda){
  exp(-lambda)*lambda^k/factorial(k)
}
### note: lambda is the expected value of the Poisson RV
### and expected value is approximated by the mean of the sample
lambda <- mean(fumbles)
points(0:M, poiss.pmf(0:M, lambda), col="red")
```
6. Alex and Burt are playing a coin flipping game against one another. Both flip their coin repeatedly, recording whether they flip H or T after each flip. Whoever has the most heads after 2000 rounds of the game (a total of 2000 flips each) wins. However, they’d like to think of this as a race. After each flip, they check who is in the lead.

(a) For a single game of 2000 rounds, produce a plot that shows how far ahead/behind Alex is after each round.

(b) For a single game, calculate the percentage of time that Alex is in the lead.
(c) Run this game many times and (1) find the average proportion of
time that alex is in the lead, and (2) produce a histogram of these
percentages that occur in the simulation. Explain why (1) and (2)
don’t contradict each other.

(d) (BONUS) Simulate this game many times, recording the average
number of times the lead changes during such a game.

(e) Now, repeat the above when Alex begins with a lead of 5.

Remark: When a “fair competition” occurs, usually one expects the lead
to change back and forth many times (one might call this “an exciting
game”). Is this what you observe here?

fair_competition.R

```r
M <- 0

trial <- function()
{
  A <- sample(c(0,1),2000,replace = TRUE)
  B <- sample(c(0,1),2000,replace = TRUE)
  countA <- ( cumsum(A) + M )
  countB <- cumsum(B)
  return(countA - countB)
}

leadTime <- function( v ){
  s <- sum(v > 0)
  return(s/length(v))
}

mean(replicate(10000,leadTime(trial())))
hist(replicate(10000,leadTime(trial())))

leadChange <- function( v ){
```
```r
count <- 0
zeros <- which(v == 0)
if( length(zeros) != 0 ){
  noZeros <- v[-zeros]
  N <- length(noZeros)
  for(k in 1:(N-1)){
    if( noZeros[k] * noZeros[k+1] < 0) { count <- (count + 1) }
  }
} return( count )

hist(replicate(10000, leadChange(trial())))
```
7. A woman has \( n \) keys, of which one will open her door.

   (a) If she tries the keys at random, discarding those that do not work, what is the probability that she will open the door on her \( k \)th try? (You can use either chapter 2 or 3 to do this!) What if she does not discard previously tried keys?

   (b) Suppose you draw 13 cards from a standard deck. What is the probability at least one suit is missing from those 13 cards? (Hint: use inclusion/exclusion for 4 events).
we make key number 1 the correct key

```r
# we make key number 1 the correct key

# we make key number 1 the correct key

library(R.utils)

# Question 1

# we make key number 1 the correct key

trial1 <- function(n){ return( sample(1:n, n) )} #sample of n keys

question1 <- function(k, t){ return( t[k]==1 )} #whether she gets the right key on the kth try

exper1 <- function(k,n,r){ return( replicate(r, question1(k, trial1(n))) )}

prob1 <- function(k,n,r) { return( mean(exper1(k,n,r)) )} # probability that she opens the door on the kth try

# we plot the probability for all tries and see that it's 1/n
# for this we fix 20 keys!

L1 <- numeric(20)
for (k in 1:20) { L1[k] <- prob1(k,20,10000) }

plot(L1, ylim =c (0 ,1) , col ="blue",main =" without replacement (20 keys)", xlab ="kth try worked ", ylab ="probability key worked ")

curve(.05*(x/x), add =T, col ="red") # plot of exact probabilities

# Question 2

trial2 <- function(k,n){ return( sample(1:n,k, replace = TRUE ) )} # sample of n keys and k tries

question2 <- function(k, t ){ return( t[k]==1 & sum( t[1:k]==1 ) ==1 )} # whether she gets the right key on the kth try and not before!

exper2 <- function(k,n,r){ return( replicate(r, question2(k, trial2(k,n))) )}

prob2 <- function(k,n,r) { return( mean(exper2(k,n,r)) )} # probability that she opens the door on the kth try

# we plot the sample plot as before, this time w/ replacement:

L2 <-numeric(20)
for (k in 1:20) { L2[k] <- prob2(k,20,10000) }

plot(L2, ylim =c (0 ,1) , col ="blue",main ="with replacement (20 keys)", xlab ="kth try worked ", ylab ="probability key worked ")

curve((1 / 20) * (19 / 20) ^(x -1) , add =T, col ="red") # plot of exact probabilities

# Though, with replacement is more interesting when there are fewer keys
# and more tries than number of keys:

L3 <-numeric(20)
for (k in 1:20) { L3[k] <- prob2(k,3,10000) }

plot(L3, ylim =c (0 ,1) , col ="blue",main ="with replacement (3 keys)", xlab ="kth try worked ", ylab ="probability key worked ")
```
curve((1/3)*(2/3)^(x-1), add=T, col="red")

# Note: question2 is not the "most intuitive" way
# to define whether she gets the correct key / and not before.
# this is an alternate way of defining it.
# (you could also write a more direct for loop!)

first.elements<-function(k,v){
  # input a number k and a vector
  # outputs the first k elements of v
  ans<-numeric(k)  # start with a vector of k zeros
  for( i in 1:k ){ ans[i]<-v[i] }  # change ans to v
  return( ans )
}

question2<-function(k,t){
  iskey <- t[k]==1
  firstkeys <- first.elements(k-1,t)  # first k-1 tries to open
  not.open.before <- sum( firstkeys == 1 ) == 0
  return( sum( c(iskey,not.open.before) ) == 2 )
}

## Question 2
## How often is a bridge hand void in at least one suit?

hearts <- replicate(13,'H')
clubs <- replicate(13,'C')
spades <- replicate(13,'S')
diamonds <- replicate(13,'D')
deck <- c(hearts,clubs,spades,diamonds)

trial<- function(){
  return( sample(deck,13) )
}

question <- function( t ){return( length( unique( t ) ) < 4 )}

exper<-function( n ){return( replicate(n,question(trial())))}

rel.freq <- function( n ){return( mean(exper(n)) )}
```

8. (a) Set up the following trial: flip a coin $N = 2000$ times and record the "largest streak of heads" that occurs during those flips. Repeat this trial $k$ times and compute the average of your results. (As $k \to \infty$, this average converges to what's called "the expected value" of the length of the longest streak).
Given a number $n$, estimate (using simulation) the probability that a streak of $n$ heads appears in $N = 2000$ coin flips?

```r
N <- 2000
## trial = N coin flips
flips <- function(){
  # no input
  # outputs N coin flips
  # Heads = 1 / Tails = 0
  return( sample(c(0,1),N,replace=TRUE))
}

## now we count the streaks at each point in the a trial (given by flips()

streak <- function( trial ){
  # inputs a vector of 0/1 of length N
  # outputs a vector w/current streak count
  L <- numeric(N)
  for ( k in 2:N ) {
    if ( trial[k] == 1 ) { L[k] <- L[k-1]+1
    } else { L[k] <- 0
    }
  }
  return( L )
}

## take the max streak present in a trial:

max.streak<- function( trial ){
  # inputs a trial
  # outputs the max streak of HEADS
  return( max(streak( trial )))
}

## replicate the trial (N=2000 coin flips) and record the max streak each time
exper <- function( n ){
  return( replicate(n, max.streak(flips())) )
}

# now record the average length of the max streak # and view histogram of all the max streaks
mean(exper(100))
hist(exper(100),breaks=seq(.5,20.5),col='yellow')
```

## what is the probability there is
## a streak of length n in N=2000 coin flips?

## here choose K=100 trials to perform the N=2000 coin flips
## and ask each time: " was there a streak of length n?"

K<-100

hundred.trials <- function( n ){
  # input n = length of streak
  # outputs K=100 trials of N coin flips
  # TRUEs if a streak of length n occurs
  L<- replicate(K, max(streak(flips())) > n )
  return( L )
}

## count up the relatively frequencies
## and plot, for each n=1...20, the relative freqs.

rel.freq <- function( n ){
  return( sum( hundred.trials(n))/K )
}

L<-numeric(20)
for (k in 1:20) L[k] <- rel.freq(k)

plot(L,type='o',col='blue', main='Probability that a streak of length n
occurs in 2000 flips of a coin',xlab='streak length',ylab='probability')

# Bonus: how many flips does it take to hit
## a streak of n heads?

# Notice the 'while loop' in the solution!

str.time <- function( n ){
  # input length of streak
  # outputs number of flips until
  # a streak of length n is reached
  str.count <- 0
  flip.count <- 0
  while ( 0 < 1 ){
    flip <- sample( c(0,1), 1)
    flip.count <- flip.count + 1
    if ( flip == 1 ) { str.count <- str.count + 1
    } else { str.count <- 0 }
    if ( str.count >= n ) break
  }
  return( flip.count )
}
max number of streaks of heads
9. Read example 5m on page 39 of Ross (The matching hat problem). Write and run a simulation (for $N = 50, 100, etc$) of the problem and estimate the probability that none of the men select their own hat. Compare your answer to the exact answer in the book.

```r
N <- 35 # how many ppl at party
## Each person is a number between 1 and N
hat.shuffle <- function(){
```

![Graph showing probability of a streak in 2000 coin flips]
function hat_shuffle()
return( sample(1:N,N) )
}

##

own.hat <- function( hatconfig ){
  # inputs vector length N that
  # specifies a mixing of hats.
  # outputs a vector V[k]=TRUE if
  # kth person has own hat.
  return( hat.shuffle() == 1:N )
}

##

at.least.one <- function( V ){
  # input a T/F vector
  # outputs if at least one person
  # picked up their own hat.
  return( sum( V ) > 0 )
}

##

trial <- function(){
  # NO INPUT
  # outputs TRUE if at least on person
  # picked up own hat.
  return( at.least.one( own.hat( hat.shuffle() ) ) )
}

##

exper <- function( k ){
  return( replicate(k, trial() ) )
}

rel.freq <- function( k ){
  return( sum(exper(k))/k )
}

###

# Compare with the exact answer in Ross
# Section 2.5 example 5m on pg. 39
###

# Drunk airplane passenger in an
# exceedingly polite place:
# First passenger is drunk and just sits down in random seat.
# The rest of the passengers try to go to their assigned seat.
# If it’s free, they sit there, otherwise they pick a random
# seat. What is the expected number of people who
# are seated properly?
# (Note: prob last passenger gets her seat: 1/2)

# There are N passengers, who *should* be seated
# in order 1 through N.

N <- 100

## A function that removes a seat from the list of available seats:

rm.seat <- function ( avail.seats , seat ){
  # input a seat number
  # outputs an updated vector of
  # available seats.
  return ( avail.seats[ avail.seats != seat ] )
}

## A function that checks if a seat is available:

is.available <- function ( available.seats , seat ){
  # input seat number
  # outputs TRUE if seat is in available.seats
  return ( sum ( available.seats == seat ) > 0 )
}

##

fill.plane <- function (){
  # NO INPUT (random trial)
  # Outputs a seating chart that’s filled
  # according to procedure in question.
  # i.e. a vector v[k] = where the kth psgr is seated.
  available.seats <- 1:N  # all seats are initially available
  # seating.chart[k] will be the seat of the kth person.
  seating.chart <- numeric(N)  # To be filled with seat values.
  # Step 1: drunk person sits anywhere:
  rseat <- sample( available.seats , 1 )  # pick random seat
  seating.chart[1] <- rseat  # assign rseat to passenger 1
  available.seats <- rm.seat( available.seats , rseat )  # update avail
  seats.
  # Steps 2 N: other ppl sit down.
  for ( psgr in 2:N )
    if ( is.available(available.seats, psgr ) ){
      seating.chart[ psgr ] <- psgr  # psgr sits in own seat
      available.seats <- rm.seat( available.seats, psgr )
    } else {
      rseat <- sample( available.seats , 1)
      seating.chart[ psgr ] <- rseat
      available.seats <- rm.seat( available.seats, rseat )
    }
  return ( seating.chart )
}


10. Simulate the following “dartboard experiment.” Let $S$ be the square in \( \mathbb{R}^2 \) with vertices \((\pm 1, \pm 1)\), and let $C$ be the inscribed circle of radius 1 centered at \((0, 0)\). The experiment consists of throwing a dart randomly at the square. If the dart lands inside $C$, then you record “HIT,” otherwise you record “MISS.” Use this simulation to estimate the number $\pi$. Roughly how many simulations do you have to run to get $\pi$ correct to 4 digits? (We’ll answer this question precisely later in the course).

```
# HW #1 Problem 11

# Throwing darts to calculate pi:
# notice that, from in class, the probability of a dart hitting board is pi/4.
# So the rel.freq -> pi/4 as n-> infinity.

## First we define a throw of a dart.
throw <- function(){
  # NO input
  # outputs a vector of two random numbers thought of as (x,y), both between 0 and 1.
  return( runif(2,-1,1) )
}

## next we ask if (x,y) landed in the circle.
in.circle <- function( dart ){  
  # input a dart coordinate (x,y)
  # outputs TRUE if (x,y) is in circle
}

## next we define a trial:
trial <- function(){  
  # NO input
  # outputs TRUE if random dart throw lands in circle
  return( in.circle( throw() ) )
}
```
## The functions exper, rel.freq, etc...
## can be copied directly from the previous
## problem.

pi.approx <- function(k){
  # input number of trials
  # outputs approximation of pi
  return( 4*rel.freq(k) )
}

11. Show that with 24 rolls of a pair of dice, the probability of at least one
instance of a "double ace" is slightly less than .50, but with 25 rolls, the
probability is slightly greater than .50. For the experiment of 25 rolls,
plot the relative frequencies for a sequence of $n = 1000$ trials.

double.ace.R

## double.ace.R

# HW #1 Problem 10

# a trial consists of rolling 2 dice N=24 times,
# and checking if you rolled (1,1)
N <- 24 # number of rolls of the two dice

# first we define a roll of two dice:
roll <- function(){
  # takes NO input
  # outputs two dice rolls
  return( sample(1:6,2,replace=TRUE) )
}

# next we check if a roll is (1,1):
is.snake.eyes <- function(roll){
  # input a roll of the dice (2 numbers)
  # outputs TRUE if the roll is (1,1)
  return( sum( roll ) == 2 )
}

# now, we run a "trial", that is
# we roll 2 dice N = 24 times, and check if (1,1) appears
trial <- function(){
  # inputs NO input
  # outputs a vector of length N, TRUE if that roll was (1,1)
  v <- replicate( N, is.snake.eyes(roll()) ) # vector of T/Fs
  s <- sum( v ) # number of (1,1) rolled in 24 throws
  return( s > 0 ) # outputs T if s > 0
}

# now we run the trial k times:
exper <- function(k){

# input number of times to run the trial
# outputs a vector of T/F of length k
# with TRUE when a trial was a 'success.'
return ( replicate ( k, trial () ))
}

## relative freq. calculator
rel.freq <- function ( k ){
  # input number of trials
  # outputs the relative freq of rolling (1,1)
  return ( sum ( exper ( k ))/k )
}

## Now you can change N=25 and run the same functions.
## To do the plotting, you can take the rel.frequencies / rf.analysis
## functions DIRECTLY from the sample example.