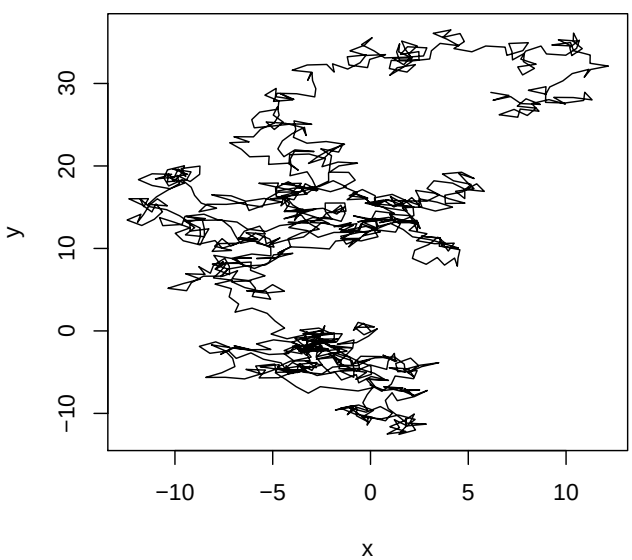
Additional exercises

1 Brownian motion

Write a function that simulates a random walk:

1. From a starting position of $x = 0$ and $y = 0$, move a virtual particle by a distance of 1 in a random direction.
2. Repeat $n = 1000$ times and plot the track of the particle as a line.

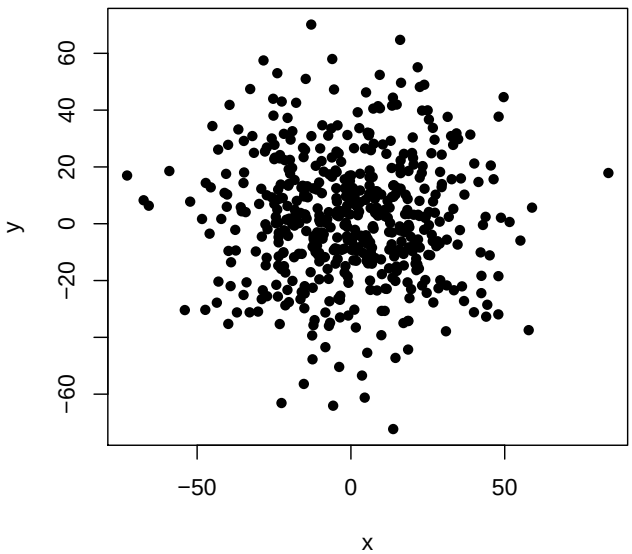
```
# brownian motion for one particle:
brown <- function(xy0=rep(0,2),n=1000){
  xy <- matrix(rep(xy0,n),ncol=2,byrow=TRUE)
  d <- runif(n,min=0,max=2*pi) # random directions
  for (i in 2:n){
    dx <- cos(d[i])
    dy <- sin(d[i])
    xy[i,1] <- xy[i-1,1] + dx
    xy[i,2] <- xy[i-1,2] + dy
  }
  return(xy)
}
xy <- brown()
plot(xy,type='l',xlab='x',ylab='y')
```



3. Repeat steps 1 and 2 for $N = 500$ virtual particles and visualise their final positions on a scatter plot.

```
n <- 1000
N <- 500
xyf <- matrix(NA,nrow=N,ncol=2)
for (i in 1:N){
  xyf[i,] <- brown(n=n)[n,]
```

```
}
plot(xyf,type='p',pch=16,xlab='x',ylab='y')
```

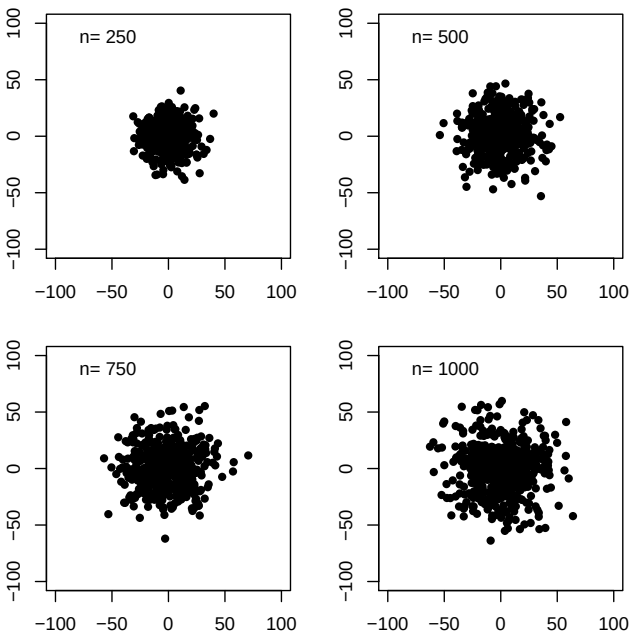


2 Diffusion

Using the code from Section 1:

1. Repeat step 3 for $n = 250$, 500, 750 and 1000 iterations and visualise the final positions on a 2×2 panel grid of scatter plots. Adjust the axis limits so that all four panels are plotted at the same scale.

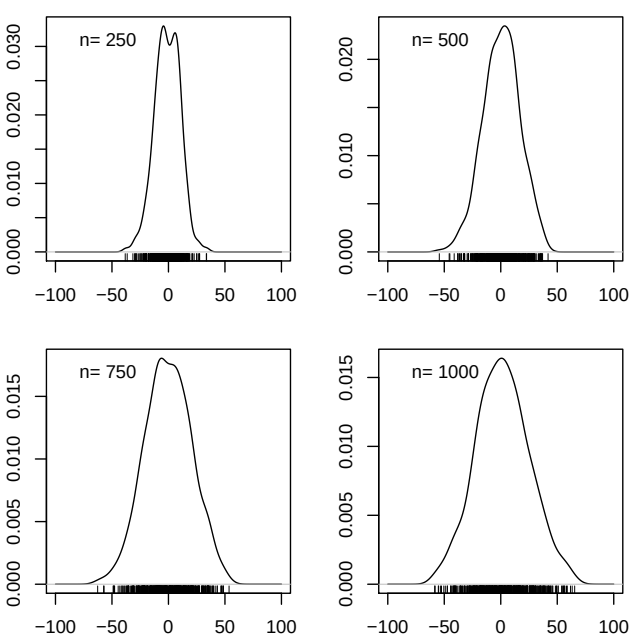
```
cloud <- function(n=1000,N=500,plot=TRUE,...){
  xyf <- matrix(NA,nrow=N,ncol=2)
  for (i in 1:N){
    xyf[i,] <- brown(n=n)[n,]
  }
  if (plot) plot(xyf,pch=16,xlab='x',ylab='y',...)
  invisible(xyf) # returns the values without printing them at the console
}

oldpar <- par(mfrow=c(2,2),mar=rep(2,4))
lims <- c(-100,100)
for (n in c(250,500,750,1000)){
  cloud(n=n,xlim=lims,ylim=lims)
  legend('topleft',legend=paste('n=',n),bty='n')
}
```



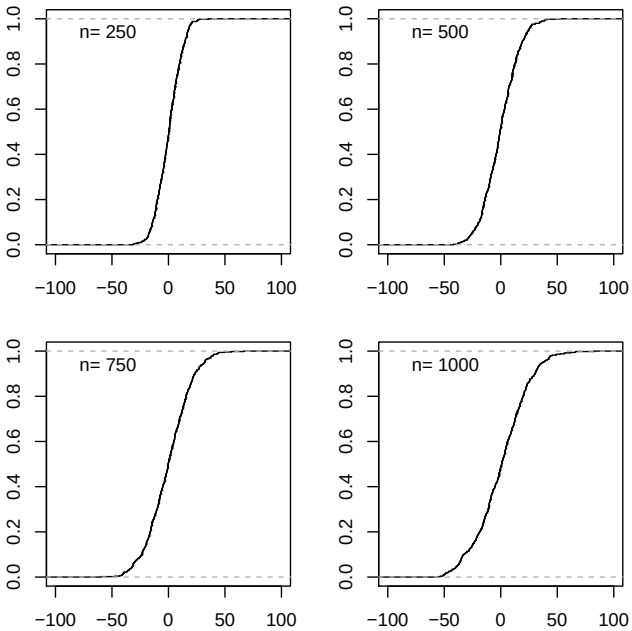

```r
par(oldpar)
```

2. Plot the marginal distributions of the x-values as kernel density estimates and empirical cumulative distribution functions.

```r
oldpar <- par(mfrow=c(2,2),mar=rep(2,4))
for (n in c(250,500,750,1000)){
  xy <- cloud(n=n,plot=FALSE)
  d <- density(xy[,1],from=-100,to=100)
  plot(d,main='')
  rug(xy[,1])
  legend('topleft',legend=paste('n=',n),bty='n')
}
```



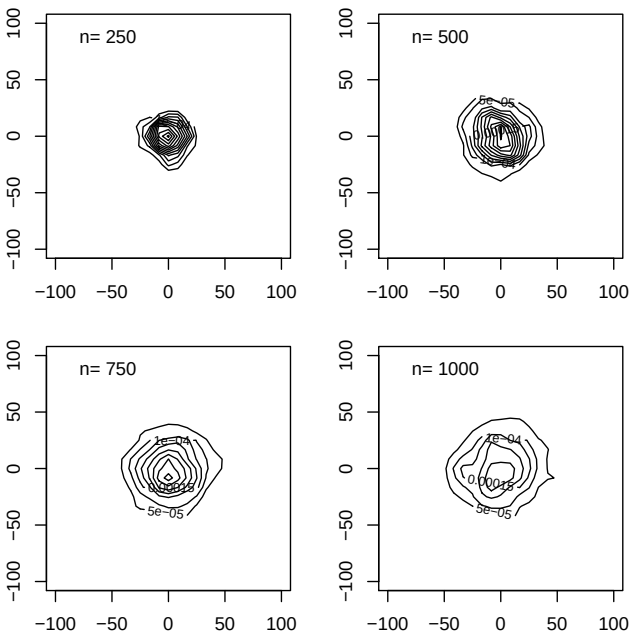
```r
for (n in c(250,500,750,1000)){
  xy <- cloud(n=n,plot=FALSE)
  d <- ecdf(xy[,1])
  plot(d,xlim=c(-100,100),main='',verticals=TRUE,pch=NA)
  legend('topleft',legend=paste('n=',n),bty='n')
}
```




```r
par(oldpar)
```

3. Visualise the bivariate datasets as 2-dimensional KDEs.

```r
oldpar <- par(mfrow=c(2,2),mar=rep(2,4))
lims <- c(-100,100)
for (n in c(250,500,750,1000)){
  xy <- cloud(n=n,plot=FALSE)
  d2d <- MASS::kde2d(x=xy[,1],y=xy[,2],lims=rep(lims,2))
  contour(d2d,xlim=lims,ylim=lims)
  legend('topleft',legend=paste('n=',n),bty='n')
}
```




```
par(oldpar)
```

3 Summary statistics

Using the code from the previous exercises:

1. Construct a table with the means, medians, standard deviations and interquartile ranges of the synthetic datasets (x-values) of exercise 2.

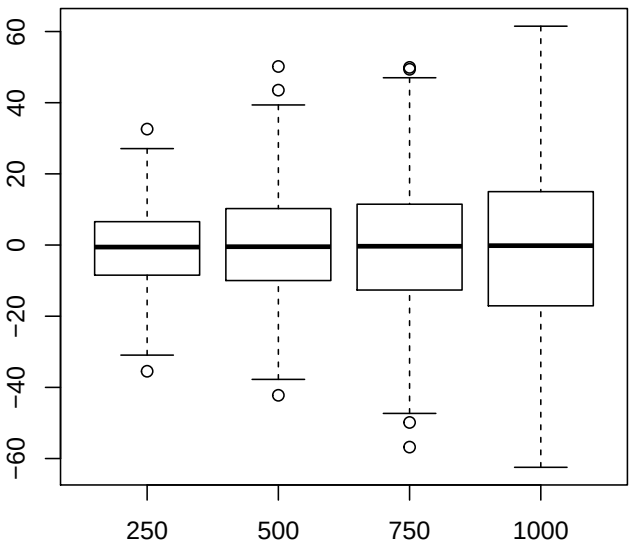
```
n <- c(250,500,750,1000)
nn <- length(n)
tab <- data.frame(means=rep(0,nn),medians=rep(0,nn),sds=rep(0,nn),iqrs=rep(0,nn))
for (i in 1:nn){
  x <- cloud(n=n[i],plot=FALSE)[,1]
  tab[i,'means'] <- mean(x)
  tab[i,'medians'] <- median(x)
  tab[i,'sds'] <- sd(x)
  tab[i,'iqrs'] <- IQR(x)
}
tab
```

```
##        means    medians      sds     iqrs
## 1 -0.6522625 -0.7328471 11.45335 14.99442
## 2 -1.4863479 -0.7663299 16.83688 21.74967
## 3  1.5098769  1.6914350 18.51301 24.23334
## 4 -0.9054386 -1.7237713 22.03064 29.02776
```

2. Plot the four datasets as box plots.

```
n <- c(250,500,750,1000)
dat <- list()
for (i in 1:length(n)){
  dat[[i]] <- cloud(n=n[i],plot=FALSE)[,1]
}
names(dat) <- n
```

```r
boxplot(dat)
```



4 Probability

Suppose that a password needs to contain exactly 8 characters and must include at least one lowercase letter, uppercase letter and digit. How many such passwords are possible?

The solution to this problem requires the *inclusion-exclusion principle*, which can be derived from the additive rule of probability (Equation 4.4 of the notes). Let u, l and d represent outcomes containing uppercase letters, lowercase letters and digits, respectively. Then we are looking for all possible combinations of those three outcomes:

$$
\begin{aligned}
&|u \cup l \cup d| \\
&= |u \cup l| + |d| - |u \cup l \cap d| \\
&= |u| + |l| + |d| - |u \cap l| - |(u \cup l) \cap d| \\
&= |u| + |l| + |d| - |u \cap l| - |u \cap d| - |l \cap d| + |u \cap l \cap d|
\end{aligned}
$$

In R:

```r
len <- 8 # length of the password
nupper <- 26 # A -> Z
nlower <- 26 # a -> z
ndigit <- 10 # 0 -> 9
# total number of 8-character strings:
N <- (nupper+nlower+ndigit)^len
# remove all the passwords with no uppercase letter, lowercase letter, or digit:
N <- N - (nlower+ndigit)^len - (nupper+ndigit)^len - (nlower+nupper)^len
# But then you removed some passwords twice. You must add back all passwords with
# no lowercase AND no uppercase, no lowercase AND no digit, or no uppercase AND no digit:
N <- N + ndigit^len + nupper^len + nlower^len
# hence the total number of passwords is
print(N)
```

```
## [1] 1.596559e+14
```

5 Bernoulli variables

1. Draw a random number from a uniform distribution between 0 and 1 and store this number in a variable (p, say).

```
#set.seed(2) # to get reproducible results (comment out for truly random values)
p <- runif(1)
print(p)
```

```
## [1] 0.3805307
```

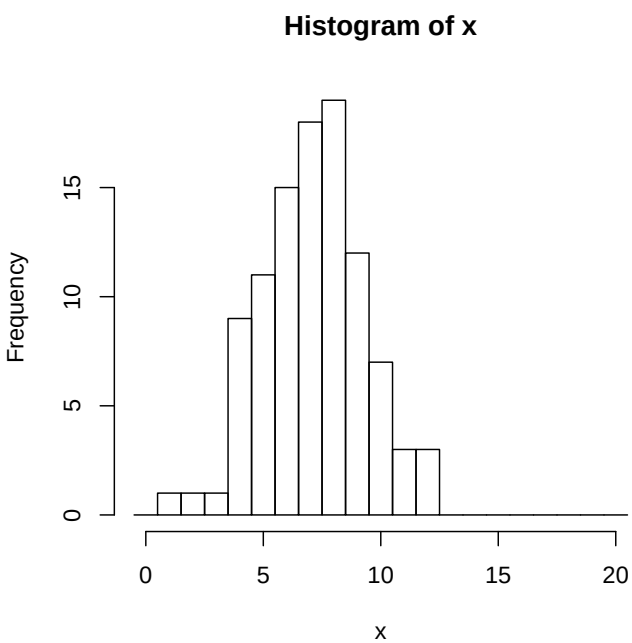
2. draw $n = 20$ additional numbers from the same uniform distribution and count how many of these values are less than p.

```
n <- 20
r <- runif(n)
print(sum(r<p))
```

```
## [1] 6
```

3. Repeat steps 1 and 2 $N = 100$ times, store the results in a vector (x, say), and plot the results as a histogram.

```
N <- 100
m <- matrix(runif(n*N),nrow=N,ncol=n)
x <- rowSums(m<p)
h <- hist(x,breaks=seq(from=-0.5,to=20.5,by=1))
```



4. Suppose that you did not know the value of p, then you could estimate it (as a new variable $\hat{p}$, say) from the data x. To this end, compute the binomial density (or 'likelihood') of all values in x for $\hat{p} = 0.5$ and sample size n.

```
phat <- 0.5 # example value
d <- dbinom(x=x, size=n, prob=phat)
unique(d)
```

```
##  [1] 1.201344e-01 3.696442e-02 7.392883e-02 1.907349e-05 1.478577e-02
##  [6] 1.601791e-01 4.620552e-03 1.761971e-01 1.811981e-04 1.601791e-01
```

```
## [11] 1.087189e-03
```

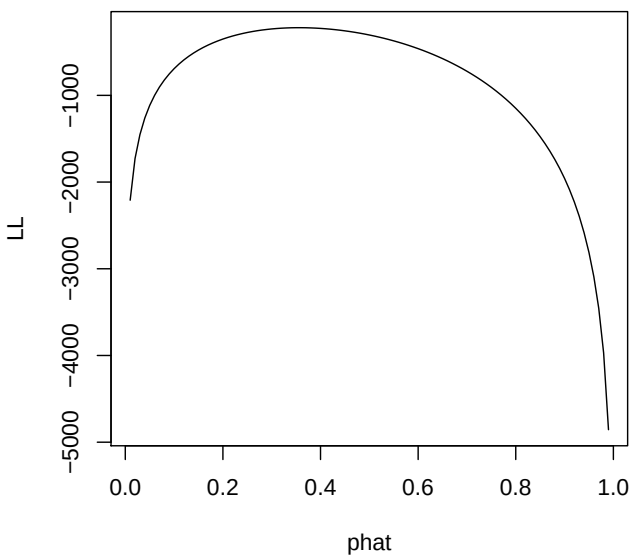
5. Now take the sum of the logarithms of the binomial likelihoods of x. Call this sum LL (for 'log-likelihood').

```
d <- dbinom(x=x, size=n, prob=phat, log=TRUE)
LL <- sum(d)
print(LL)
```

```
## [1] -303.6544
```

6. Repeat step 5 for a regularly spaced sequence of $\hat{p}$-values. Then plot the resulting LL-values against those $\hat{p}$-values.

```
phat <- seq(from=0.01,to=0.99,by=0.01) # values of 0 and 1 don't work
np <- length(phat)
LL <- rep(0,np)
for (i in 1:np){
  d <- dbinom(x=x, size=n, prob=phat[i], log=TRUE)
  LL[i] <- sum(d)
}
plot(phat,LL,type='l')
```



7. Approximately which value of $\hat{p}$ corresponds to the maximum value for LL? How does this compare to the true value of p?

```
i <- which.max(LL)
message('phat=',phat[i],', p=',p)
```

```
## phat=0.36, p=0.380530662834644
```

8. Calculate the weighted average of the histogram counts obtained under step 2:

$$\hat{p}' = \frac{1}{n}\sum_{i=0}^{n}\frac{i \times h_i}{N}$$

where h_i is the number of counts in bin i.

```
mean(h$mids*h$counts/N)
```

```
## [1] 0.3390476
```

9. How does $\hat{p}'$ compare to the result obtained under step 7?

By completing this exercise, you have numerically extended the result of Section 5.1 in the notes.