Additional exercises

Unlike the exercises in the notes, whose solutions are provided in Chapter 19, you must attempt these questions yourself before I will share my answer with you. Please don't hesitate to ask questions. Then upload your solution to `http://doubleblind.dynu.net/GEOL0061` to download the model answer. Submissions won't be formally assessed, so there is no need to worry if your answer is wrong. Try, struggle, ask questions, and learn!

1 Random walk

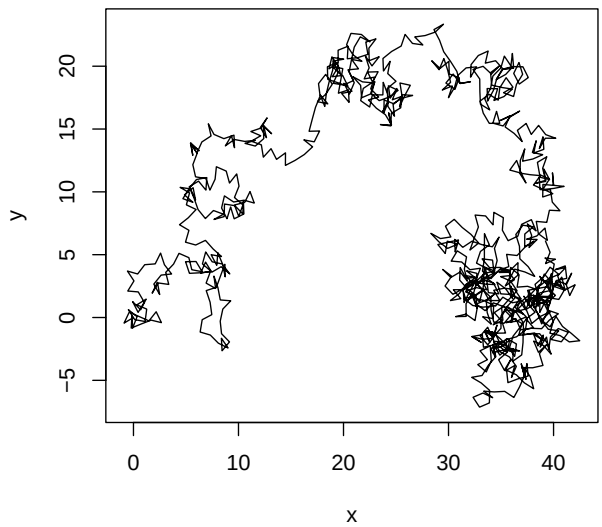
Write an `R` function that simulates a random walk:

1. From a starting position of $x = 0$ and $y = 0$, move a virtual particle by a distance of 1 in a random direction.

```
xy <- c(0,0) # initial position
d <- runif(1,min=0,max=2*pi) # choose a random direction
dx <- cos(d)
dy <- sin(d)
xy[1] <- xy[1] + dx # updated x-position
xy[2] <- xy[2] + dy # updated y-position
```

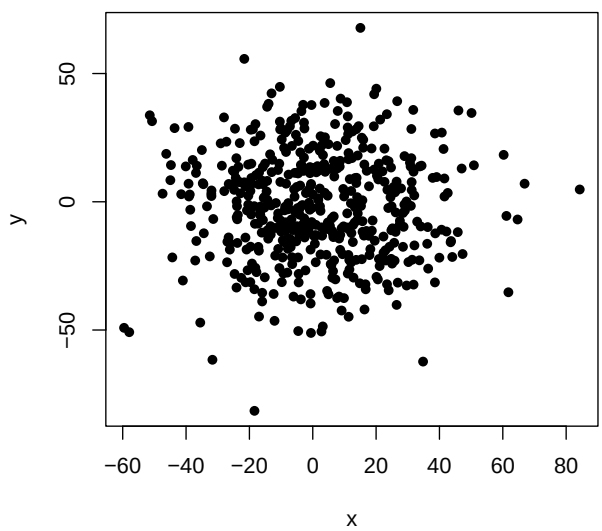
2. Repeat $n = 1000$ times and plot the track of the particle as a line.

```
# random walk for one particle:
walk <- function(xy0=rep(0,2),n=1000){
   xy <- matrix(rep(xy0,n),ncol=2,byrow=TRUE)
   d <- runif(n,min=0,max=2*pi)
   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 <- walk()
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,] <- walk(n=n)[n,]
}
plot(xyf,type='p',pch=16,xlab='x',ylab='y')
```



The distribution of the end positions is dense in the middle and thins out near the edges. We will see (in Chapter 7 of the notes) that it follows a so-called bivariate Gaussian distribution.

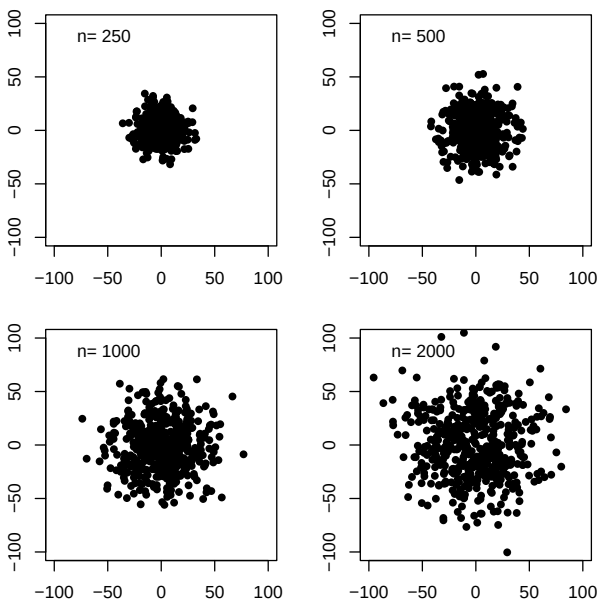
2 Diffusion

Using the code from exercise 1:

1. Repeat step 1.3 for $n = 250$, 500, 1000 and 2000 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.

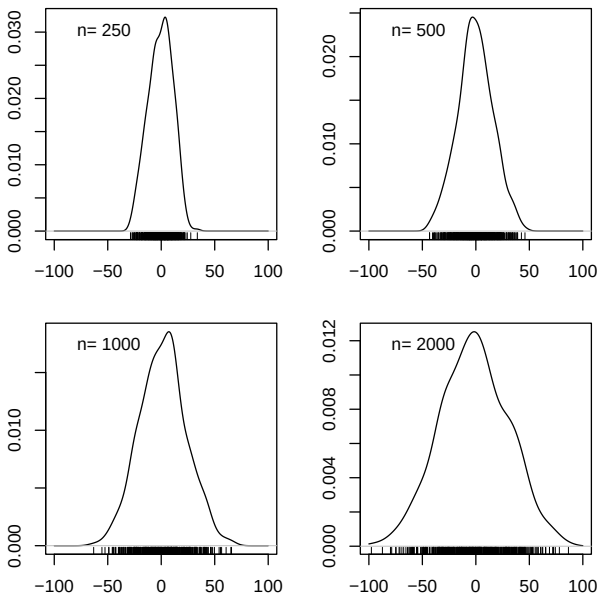
```r
cloud <- function(n=1000,N=500,plot=TRUE,...){
   xyf <- matrix(NA,nrow=N,ncol=2)
   for (i in 1:N){
      xyf[i,] <- walk(n=n)[n,]
   }
   if (plot) plot(xyf,pch=16,xlab='x',ylab='y',...)
   invisible(xyf) # returns the values without printing them at the console
}

par(mfrow=c(2,2),mar=rep(2,4))
lims <- c(-100,100)
ns <- c(250,500,1000,2000)
for (n in ns){
   cloud(n=n,xlim=lims,ylim=lims)
   legend('topleft',legend=paste('n=',n),bty='n')
}
```

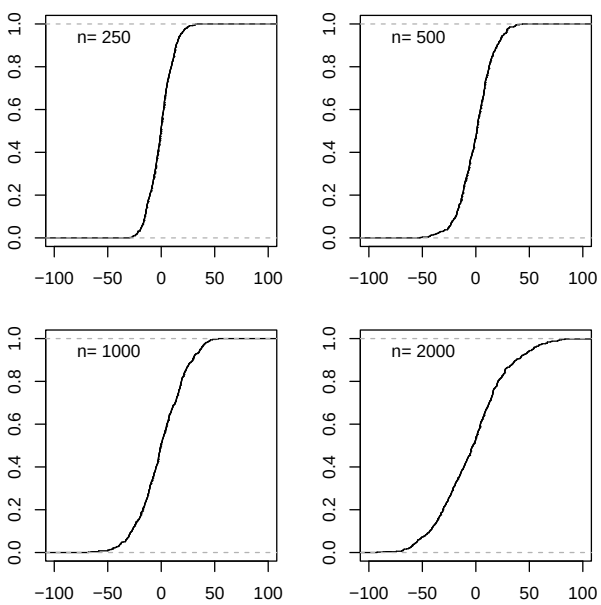


2. Plot the marginal distributions of the x-values as kernel density estimates and empirical cumulative distribution functions.

```r
par(mfrow=c(2,2),mar=rep(2,4))
for (n in ns){
   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')
}
```

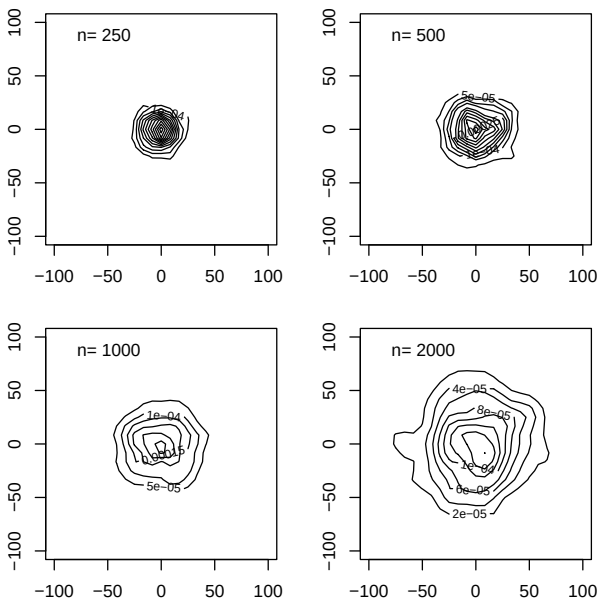



```r
for (n in ns){
   xy <- cloud(n=n,plot=FALSE)
   d <- ecdf(xy[,1])
   plot(d,xlim=c(-100,100),main='',verticals=TRUE,pch=NA)
   legend('topleft',legend=paste('n=',n),bty='n')
}
```



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

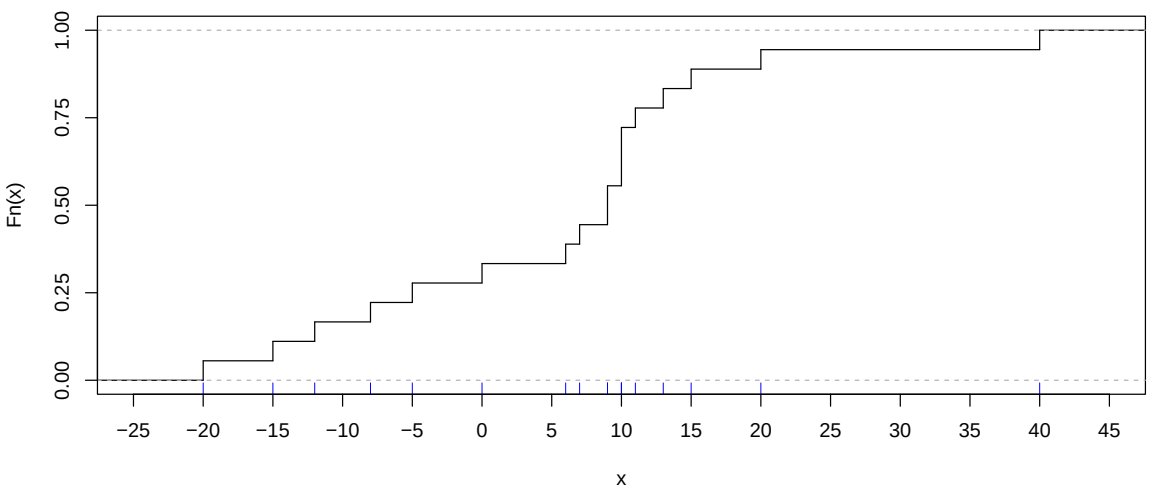
```r
par(mfrow=c(2,2),mar=rep(2,4))
lims <- c(-100,100)
for (n in ns){
   xy <- cloud(n=n,plot=FALSE)
   d2d <- MASS::kde2d(x=xy[,1],y=xy[,2],lims=rep(lims,2))
   contour(d2d,xlim=lims,ylim=lims)
   legend('topleft',legend=paste('n=',n),bty='n')
}
```



This exercise has shown how random walks of particles give rise to Gaussian diffusion, which produces characteristic 'bell shaped' probability distributions (in one and two dimensions) that spread out over time.

3 Summary statistics

Considering the dataset of counts whose ECDF is shown in the following Figure:

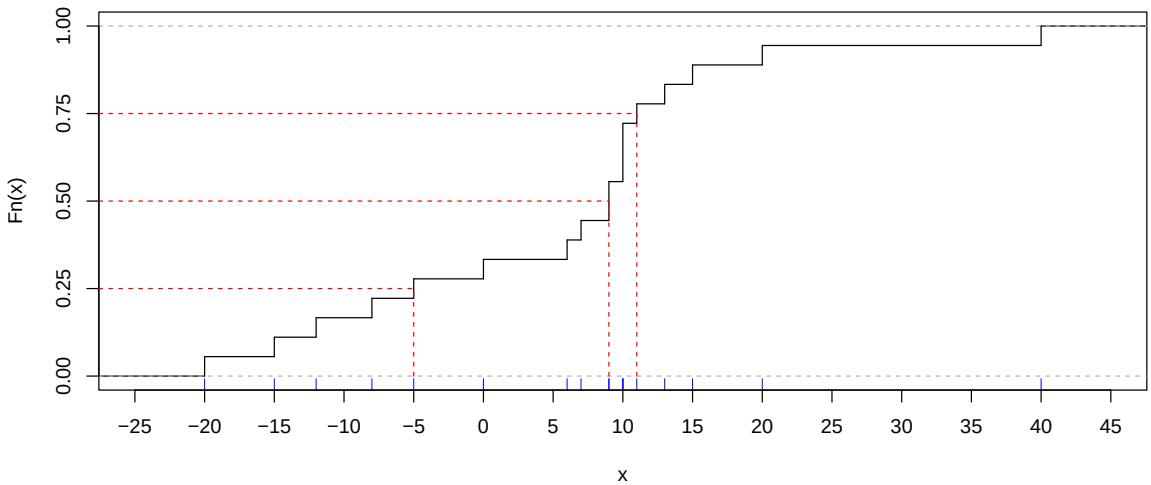


1. How many measurements are there in this dataset?

   There are 16 ticks in the rug plot. However, this includes some duplicate values. A better way to count the observation is to use the steps in the ECDF. There are 13 single steps, one double step (at 9), and one triple step (at 10). Thus, the dataset contains 18 values.

2. What are their median and IQR?

   Drawing the 25, 50 and 75 percentiles as dashed lines on the ECDF:



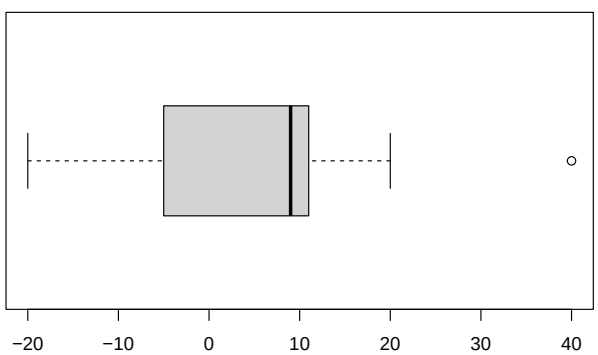
The 25, 50 and 75 percentiles are -5, 9 and 10, respectively. Therefore, the median is 9 and the IQR is $10 - (-5) = 15$.

3. Draw a box plot for this dataset (either using `R`, or by hand with pencil on paper). Does it identify any outliers?

   By hand, you would first draw a box from -5 *to 10, with a line at 9. Then, whiskers would extend from this box until* -20 *and* $+20$. *The value of 40 is an outlier, because* $40 - 9 = 31 > 1.5IQR = 22.5$.

   Using R:

```
dat <- c(-20,-15,-12,-8,-5,0,6,7,9,9,10,10,10,11,13,15,20,40)
boxplot(dat,horizontal=TRUE)
```



4 Probability

1. A deck of 52 cards is shuffled thoroughly. What is the probability that the four aces are all next to each other?

   There are 52! ways to shuffle a deck of cards. There are 4! ways to arrange a sequence of four aces, and 48! ways to arrange the remaining cards. Finally, there are 49 ways to insert the group of four aces in between the 48 remaining cards. Therefore, the probability of the four aces sitting next to each other is:

$$\frac{4!48!49}{52!} = \frac{4 \cdot 3 \cdot 2}{52 \cdot 51 \cdot 50} = \frac{1}{5525}$$

2. 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?

Let $|x|$ stand for 'the number of outcomes for x'. Then

$$|good\ passwords| = |all\ passwords| - |bad\ passwords|$$

where

$$|all\ passwords| = (26 + 26 + 10)^8$$

The number of bad passwords requires the inclusion-exclusion principle*, which is similar to the additive rule of probability (Equation 4.4 of the notes). Let u, l and d represent outcomes* ***lacking*** *uppercase letters, lowercase letters and digits, respectively. Then we are looking for all possible combinations of those three outcomes:*

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

where $|u \cap l \cap d| = 0$ so that

$$|good\ passwords| = (26 + 26 + 10)^8 - (26 + 26)^8 - (26 + 10)^8 - (26 + 10)^8 + 26^8 + 26^8 + 10^8$$

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. The following table shows the cumulative distribution function of a discrete random variable.

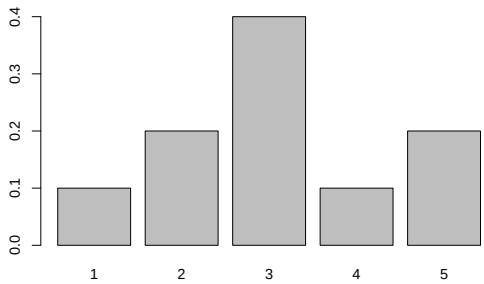
x	0	1	2	3	4	5
$F(x)$	0	.1	.3	.7	.8	1.0

Find the probability mass function. What is the mode?

Taking the differences of the cumulative probabilities:

x	*0*	*1*	*2*	*3*	*4*	*5*
$F(x)$	*0*	*.1*	*.3*	*.7*	*.8*	*1.0*
$F(x_i) - F(x_{i-1})$	–	*.1*	*.2*	*.4*	*.1*	*.2*

Plotting as a bar chart:



The mode is 3.

2. Which is more likely: 9 heads in 10 tosses of a fair coin, or 18 heads in 20 tosses?

$$\binom{10}{9} 0.5^{10} = 0.0098$$

$$\binom{20}{18} 0.5^{20} = 0.00018$$

Conclusion: 9 heads out of 10 tosses is more likely than 18 heads out of 20 tosses.

3. Suppose that in a sequence of independent Bernoulli trials each with probability of success p, the number of failures (k) up to the first success is counted. What is the probability mass function of this random variable? Suppose that $p = 0.25$. What is the probability that $k = 5$?

The probability of failure is $(1-p)$. The probability of k failures is $(1-p)^k$. Therefore, the probability of k failures followed by 1 success is

$$P(X = k) = p(1-p)^k$$

Hence

$$P(X = 5) = 0.25(1 - 0.25)^5 = 0.059$$

4. Continuing with the previous question, what is the probability mass function for the number of failures up to the rth success?

The probability of a specific outcome of k failures and r successes:

$$p^r(1-p)^k$$

The number of ways to shuffle these outcomes is $\binom{k+r}{r}$. However, the last outcome must be a success and cannot be shuffled. Therefore, the number of shuffled outcomes is $\binom{k+r-1}{r}$. Therefore, the probability of k failures up to the rth success is

$$\binom{k+r-1}{r} p^r(1-p)^k$$

This is the PMF of the so-called negative binomial distribution (see Equation 7.1 of the notes).

6 (binomial) hypothesis tests

A coin is thrown independently 10 times to test the hypothesis that the probability of heads is 0.5 versus the alternative that the probability is not 0.5. The test rejects if either 0 or 10 heads are observed.

1. What is the significance level of the test?

The fact that the rejection region consists of 0 and 10 implies that

$$\binom{10}{0} = \binom{10}{10} = 0.5^1 0 = \alpha/2$$

Therefore

$$\alpha = 2 \times 0.5^{10} = 0.002$$

2. If in fact the probability of heads is 0.1, what is the power of the test?

   The power is defined as $1-\beta$, where β is the probability of rejecting a true null hypothesis. In other words, the power of the test is the probability of rejecting a false null hypothesis. Since our rejection region consists of the outcomes 0 and 10, this probability is

$$power = \binom{10}{0} 0.9^{10} + \binom{10}{10} 0.1^{10} = 0.35$$

3. Suppose that the coin is thrown 20 times, with again a 0.1 probability of heads. Using the same significance level as before, what is the power? Use `R` to answer this question.

```
alpha <- 2*0.5^10
n <- 20
p0 <- 0.5
pa <- 0.1
k <- qbinom(alpha/2,size=n,prob=p0)
message('rejection region = {0,...,',k-1,',',n-k,',...,',n,'}')
```

```
## rejection region = {0,...,2,17,...,20}
```

```
power <- pbinom(k-1,size=n,prob=pa) + pbinom(n-k,size=n,prob=pa,lower.tail=FALSE)
message('power = ',signif(power,2))
```

```
## power = 0.68
```

7 Confidence intervals

1. Consider a Bernoulli experiment with $k = 16$ successes out of $n = 40$ trials. Use `R`'s `binom.test()` function to construct a 95% confidence interval for the parameter p. Compare your result with the approximate Equations 5.7 and 5.8 of the notes.

```
k <- 16
n <- 40
ci1 <- binom.test(x=k,n=n)$conf.int
message('exact C.I:[',paste(signif(ci1,3),collapse=','),']')
```

```
## exact C.I:[0.249,0.567]
```

```
phat <- k/n
sphat <- sqrt(phat*(1-phat)/n)
ci2 <- phat + c(-1,1)*1.96*sphat
message('approximate C.I:[',paste(signif(ci2,3),collapse=','),']')
```

```
## approximate C.I:[0.248,0.552]
```

2. Repeat the previous experiment for $k = 4$ and $n = 10$, and then again for $k = 40$ and $n = 100$, and for $k = 1$ and $n = 40$. How does the accuracy of the approximation change as a function of sample size and the value of the parameter p?

```
conftest <- function(n,k){
   ci1 <- binom.test(x=k,n=n)$conf.int
   message('exact C.I:[',paste(signif(ci1,3),collapse=','),']')
   phat <- k/n
   sphat <- sqrt(phat*(1-phat)/n)
```

```r
  ci2 <- phat + c(-1,1)*1.96*sphat
  message('approximate C.I:[',paste(signif(ci2,3),collapse=','),']')
}
conftest(n=10,k=4)
```

```
## exact C.I:[0.122,0.738]
## approximate C.I:[0.0964,0.704]
```

```r
conftest(n=100,k=40)
```

```
## exact C.I:[0.303,0.503]
## approximate C.I:[0.304,0.496]
```

```r
conftest(n=40,k=1)
```

```
## exact C.I:[0.000633,0.132]
## approximate C.I:[-0.0234,0.0734]
```

Conclusion: the accuracy of the approximation is excellent for $n = 100$, and poor for $n = 10$. It is best not to use the approximation for $n < 20$. The accuracy also suffers when $\hat{p} = n/k$ is close to 0 (or 1). In the case of $k = 1$ and $n = 40$, the approximate confidence interval spills over into nonsensical negative values.

3. How large a sample do you need to reduce the width of a 95% confidence interval for p to 0.01 when $k/n = 0.5$?

$$1.96s[\hat{p}] = 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} = 0.01$$

Therefore

$$n = \hat{p}(1-\hat{p})196^2 = 0.5^2 196^2 = 9604$$

8 The Poisson distribution

1. Telephone calls are received at a certain residence as a Poisson process with parameter $\lambda = 2$ per hour.

   (a) If Diane takes a 10-min. shower, what is the probability that the phone rings during that time? *Let λ' be the expected number of phone calls per 10-min. period. Then $\lambda' = \lambda \times 10/60 = 2/6 = 1/3$. Hence,the probability that the phone does not ring is*

$$P(k > 0|\lambda' = 1/3) = 1 - P(k = 0|\lambda' = 1/3) = 1 - \frac{\lambda'^k e^{-\lambda'}}{k!} = 1 - e^{-1/3} = 0.28$$

   Conclusion: there is a 28% chance that Diane's shower is interrupted.

   (b) How long can her shower be if she wishes the probability of no phone calls to be at most 0.5?

$$P(k > 0|\lambda' \equiv \lambda t/60) = 1 - P(k = 0|\lambda') = 0.5 = 1 - \frac{\lambda'^k e^{-\lambda'}}{k!} 1 - e^{-\lambda'}$$

$$\Rightarrow \lambda' = -\ln(0.5) = 0.051 = \lambda t/60 \Rightarrow t = \frac{0.051 \times 60}{2} = 20.8$$

   Answer: Diane's shower can last at most 20.8 minutes.

2. Recoil track geochronology is an novel dating technique that is based on the following equation:

$$t = \frac{1}{\lambda_{238}} \ln\left(\frac{N}{U} + 1\right)$$

where t is the age, λ_{238} is the decay constant of ^{238}U (=0.000155125 Myr^{-1}, which is not to be confused with the Poisson parameter λ!), U is the ^{238}U-concentration (in atoms/nm^{-3}), and N is the volume density of the recoil tracks (in nm^{-3}). N follows a Poisson distribution with unknown parameter. Suppose that $n = 100$ and $U = 10^5$. Could the sample be less than 8 million years old?

Rearranging the age equation for N:

$$N = U\left(\exp[\lambda_{238}t] - 1\right)$$

Plugging the measurements into this rearranged equation:

$$\hat{N} = 10^5\left(e^{0.000155125\times 7} - 1\right) = 124.177$$

where $\hat{N}$ stands for the 'expected value' of N. To test whether this is compatible with the observed outcome of $N = 100$, we formulate the following null hypothesis:

$$H_0 : \lambda = \hat{N} = 124.177$$

versus the alternative hypothesis:

$$H_a : \lambda < 124.177$$

Using R:

```
age <- 8
N <- 100
U <- 1e5
L38 <- 0.000155125
Nhat <- U*(exp(L38*age)-1)
pval <- ppois(N,lambda=Nhat)
message(ifelse(pval<0.05,'H0 is rejected','H0 is not rejected'))
```

```
## H0 is rejected
```

9 Continuous variables and normal distributions

The following two questions require some simple calculus, and are based on the relationship that the probability density f and the cumulative distribution function F are related as follows:

$$f(x) = \frac{\partial F(x)}{\partial x}$$

1. What is the probability density for the distance from an event to its nearest neighbour for a Poisson process in the plane?

   Let λ be the Poisson parameter that controls the average number of events per unit area:

   $$P(k|\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

   The Poisson process produces a random collection of points. When we take any location (the origin, say) and look for the closest point from our collection. Let the distance to this point be X. Then the probability that this distance falls inside a circle of radius r is given by

   $$P(X \leq r) = 1 - P(X > r) = 1 - \exp[-\lambda\pi r^2]$$

By definition, the probability density is given by

$$f(r) = \frac{\partial F(X \leq r)}{\partial r} = 2\lambda\pi r \exp[-\lambda\pi r^2]$$

2. If x follows a normal distribution with mean μ and standard deviation μ, and $y = a + bx$, prove that y follows a normal distribution with mean $a + b\mu$ and standard deviation $b\sigma$

   It is easy to show that

$$x[y] = \frac{y-a}{b}$$

   The cumulative distribution of x is

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} \exp\left[\frac{(x-\mu)^2}{2\sigma^2}\right]$$

   Therefore

$$f(y) = \frac{\partial F(x[y])}{\partial y} = \frac{\partial F(x)}{\partial x}\frac{\partial x}{\partial y} = \frac{f(x)}{b} = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[\frac{(x[y]-\mu)^2}{2\sigma^2}\right]\frac{1}{b}$$

$$= \frac{b}{b\sigma\sqrt{2\pi}} \exp\left[\frac{(y-a-b\mu)^2}{2(b\sigma)^2}\right]\frac{1}{b} = \frac{1}{b\sigma\sqrt{2\pi}} \exp\left[\frac{(y-[a-b\mu])^2}{2(b\sigma)^2}\right] = \frac{1}{\sigma'\sqrt{2\pi}} \exp\left[\frac{(y-\mu')^2}{2\sigma'^2}\right]$$

   where $\mu' = a + b\mu$ and $\sigma' = b\sigma$, which proves the proposition.

10 Error propagation

Consider a bivariate normal distribution with the following mean vector and covariance matrix:

$$\mu = \begin{bmatrix} x = -2 \\ y = 3 \end{bmatrix}, \Sigma = \begin{bmatrix} 1 & -2 \\ -2 & 5 \end{bmatrix}$$

1. Predict $z = x + 2y$ and estimate its standard error.

```
mu <- c(-2,3)
Sigma <- matrix(c(1,-2,-2,5),nrow=2,ncol=2)
x <- mu[1]
y <- mu[2]
sx <- sqrt(Sigma[1,1])
sy <- sqrt(Sigma[2,2])
sxy <- Sigma[1,2]
z <- x + 2*y
# using equation 8.10:
sz <- sqrt( sx^2 + (2*sy)^2 + 2*2*sxy )
message('z=',signif(z,2),', s[z]=',signif(sz,2))
```

```
## z=4, s[z]=3.6
```

2. Draw $n = 1000$ pairs of random numbers from the bivariate normal distribution. Compute z for each pair and calculate the mean and standard error of the resulting vector. How does it compare with your analytical solution?

```
# instead of loading the entire MASS package,
# you can also call one of its functions like this:
xy <- MASS::mvrnorm(n=1000,mu=mu,Sigma=Sigma)
Z <- xy[,1] + 2*xy[,2]
sZ <- sd(Z)
message('z=',signif(mean(Z),2),', s[z]=',signif(sZ,2))
```

```
## z=3.9, s[z]=3.6
```

3. Repeat steps 1 and 2 for $z = x^2y^3$.

```r
# step 1:
z <- (x^2)*(y^3)
dzdx <- 2*x*y^3
dzdy <- 3*(x^2)*(y^2)
# using equation 8.8:
sz <- sqrt( (dzdx*sx)^2 + (dzdy*sy)^2 + 2*dzdx*dzdy*sxy )
message('z=',signif(z,2),', s[z]=',signif(sz,2))
```

```
## z=110, s[z]=340
```

```r
# step 2:
xy <- MASS::mvrnorm(n=1000,mu=mu,Sigma=Sigma)
Z <- (xy[,1]^2)*(xy[,2]^3)
sZ <- sd(Z)
message('z=',signif(mean(Z),2),', s[z]=',signif(sZ,2))
```

```
## z=790, s[z]=2100
```

4. Repeat step 4 for

$$\Sigma = \begin{bmatrix} 0.01 & -0.02 \\ -0.02 & 0.05 \end{bmatrix}$$

```r
Sigma <- matrix(c(.01,-.02,-.02,.05),nrow=2,ncol=2)
sx <- sqrt(Sigma[1,1])
sy <- sqrt(Sigma[2,2])
sxy <- Sigma[1,2]
# step 1:
z <- (x^2)*(y^3)
dzdx <- 2*x*y^3
dzdy <- 3*(x^2)*(y^2)
# using equation 8.8:
sz <- sqrt( (dzdx*sx)^2 + (dzdy*sy)^2 + 2*dzdx*dzdy*sxy )
message('z=',signif(z,2),', s[z]=',signif(sz,2))
```

```
## z=110, s[z]=34
```

```r
# step 2:
xy <- MASS::mvrnorm(n=1000,mu=mu,Sigma=Sigma)
Z <- (xy[,1]^2)*(xy[,2]^3)
sZ <- sd(Z)
message('z=',signif(mean(Z),2),', s[z]=',signif(sZ,2))
```

```
## z=110, s[z]=34
```

Conclusion: the error propagation formulas work well for linear functions but not for strongly non-linear ones, unless the measurements are precise.

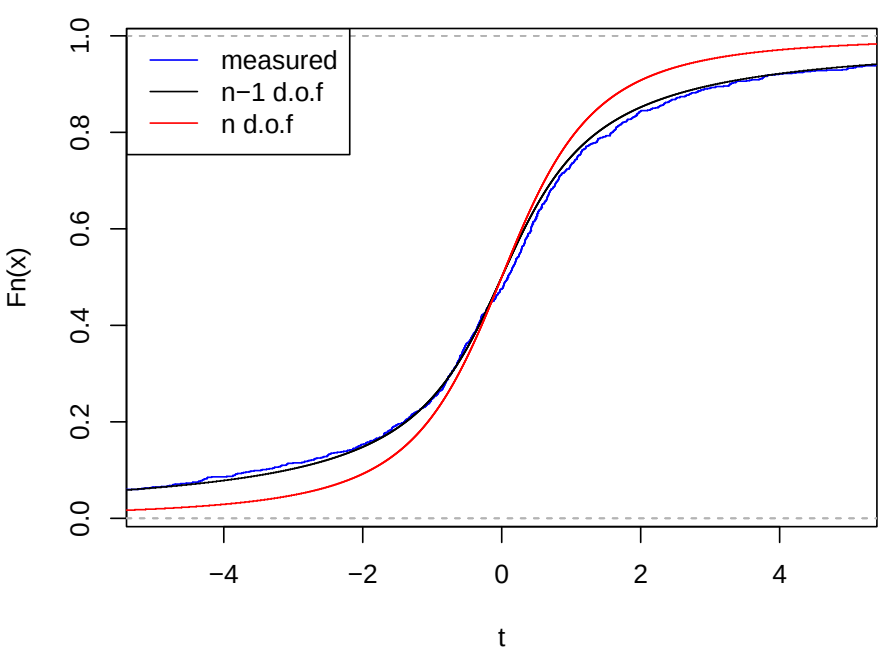
11 Degrees of freedom

The degrees of freedom of a problem represents the number of independent ways in which a system can vary. In most cases, the degrees of freedom represents the number of measurements, adjusted for the number of parameters. In the context of the t-test, we use n measurements to estimate the one-sample t-statistic t. But to do so we also have to calculate $\bar{x}$. So there is some redundancy in the system, which is reducing the apparent dispersion of the t-statistic. To account for this, we subtract one degree of freedom, in exactly

the same way as the Bessel correction, which is briefly discussed at the end of Chapter 7 of the notes. See Wikipedia for a derivation of the Bessel correction. It is a useful exercise to simulate the t-distribution on your computer:

1. Draw $N = 1000$ times $n = 2$ random numbers from a standard normal distribution and calculate their means.
2. Compute the t-statistics of the 1000 resulting values, using Equation 9.2 of the notes. Visualise as an ECDF.
3. Superimpose the CDF of the t-distribution with $n - 1$ degrees of freedom on the existing plot, and then again with n degrees of freedom. Which curve fits the simulated results best?

```
ns <- 1000       # number of samples
nv <- 2          # number of values per sample
# 1000 samples of 2 values drawn from a standard normal distribution:
obs <- matrix(rnorm(nv*ns),nrow=ns,ncol=nv)
tstat <- rep(NA,ns)  # initialise the t-statistic
for (i in 1:ns){     # loop through the samples
   tstat[i] <- sqrt(nv)*mean(obs[i,])/sd(obs[i,])  # equation 9.2 of the notes
}
# predicted quantiles of the t-distribution with nv-1 degrees of freedom:
pred1 <- qt(seq(from=0,to=1,length.out=ns),df=nv-1)
# predicted quantiles of the t-distribution with nv degrees of freedom:
pred2 <- qt(seq(from=0,to=1,length.out=ns),df=nv)
# plot the empirical cumulative distribution function of the 1000 t-statistics:
plot(ecdf(tstat),verticals=TRUE,pch=NA,col='blue',xlim=c(-5,5),xlab='t',main='')
# add the predicted distribution:
lines(ecdf(pred1),verticals=TRUE,pch=NA,col='black')
# add the second predicted distribution:
lines(ecdf(pred2),verticals=TRUE,pch=NA,col='red')
legend('topleft',legend=c('measured','n-1 d.o.f','n d.o.f'),
       lty=1,col=c('blue','black','red'))
```



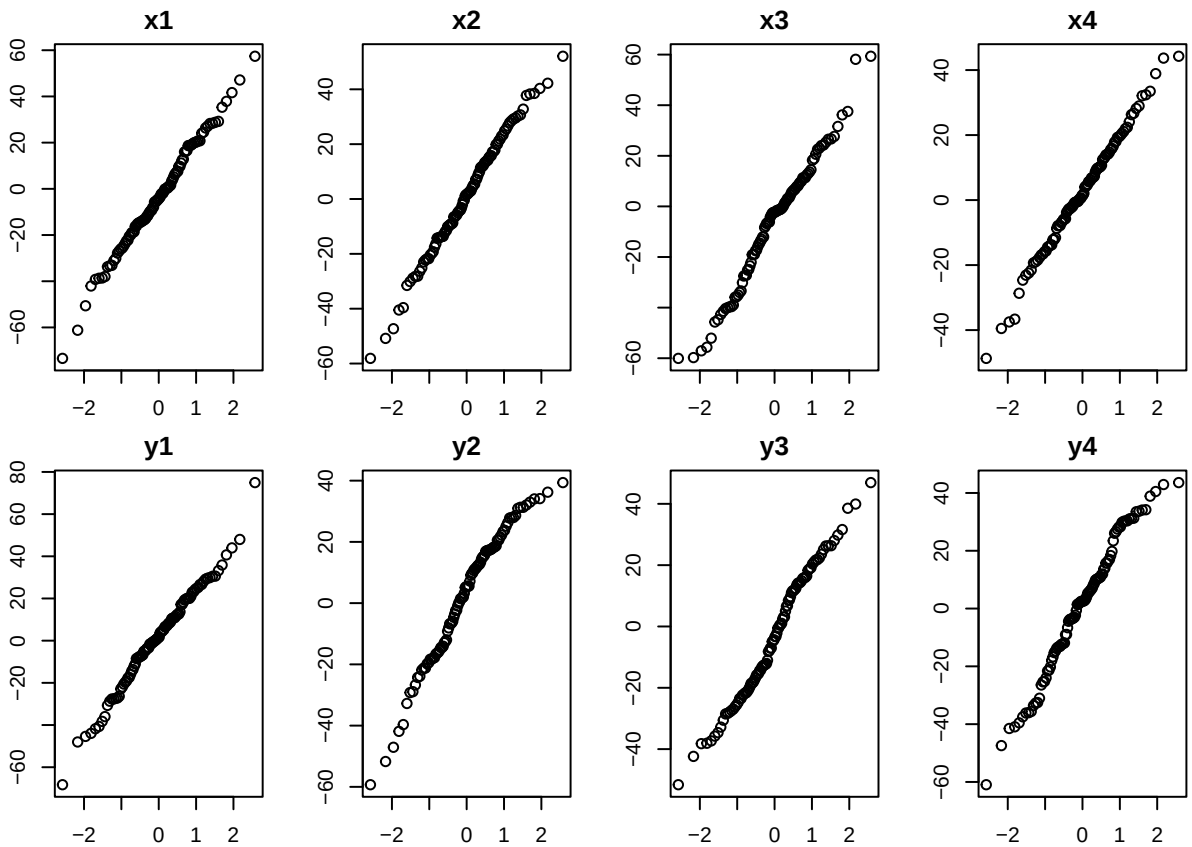
In the two-sample t-test (Equation 9.4), we have $n_1 + n_2$ measurements, and have to estimate two parameters, $\bar{x}_1$ and $\bar{x}_2$. Hence the number of degrees of freedom is $n_1 + n_2 - 2$. If you want you can modify this code for

the two-sample case to verify that this requires $n_1 + n_2 - 2$ degrees of freedom.

12 Comparing distributions

1. Compare the marginal distributions of exercise 9 with a normal distribution using Q-Q plots.

```
a <- c(1,2 ,2 ,2)
b <- c(1,0.5,0.5,0.5)
alpha <- c(0,pi/4,-pi/4,0)
n <- 1000
N <- 100
nc <- length(a)
par(mfcol=c(2,nc),mar=rep(2,4))
xy <- list() # storing the matrices in a list will save trouble later
for (i in 1:nc){
   xy[[i]] <- cloud(n=n,N=N,a=a[i],b=b[i],alpha=alpha[i],plot=FALSE)
   qqnorm(xy[[i]][,1],main=paste0('x',i))
   qqnorm(xy[[i]][,2],main=paste0('y',i))
}
```



2. Formalise the comparisons using a Kolmogorov-Smirnov test. See the documentation of the `ks.test()` function for help. Note: you should *normalise* the data by specifying the mean and standard deviation of the marginal distributions to the `ks.test()` function. See the documentation for the ellipsis (`...`) in said documentation.

```
pvals <- matrix(0,nrow=2,ncol=nc)
rownames(pvals) <- c('x','y')
for (i in 1:nc){
```

```r
   x <- xy[[i]][,1]
   y <- xy[[i]][,2]
   ksx <- ks.test(x=x,y='pnorm',mean=mean(x),sd=sd(x))
   ksy <- ks.test(x=y,y='pnorm',mean=mean(y),sd=sd(y))
   pvals['x',i] <- ksx$p.value
   pvals['y',i] <- ksy$p.value
}
signif(pvals,2)
```

```
##   [,1] [,2] [,3] [,4]
## x 0.97 0.99 0.42 1.00
## y 0.75 0.44 0.46 0.71
```

Conclusion: the marginal distributions are Gaussian.

13 Effect size

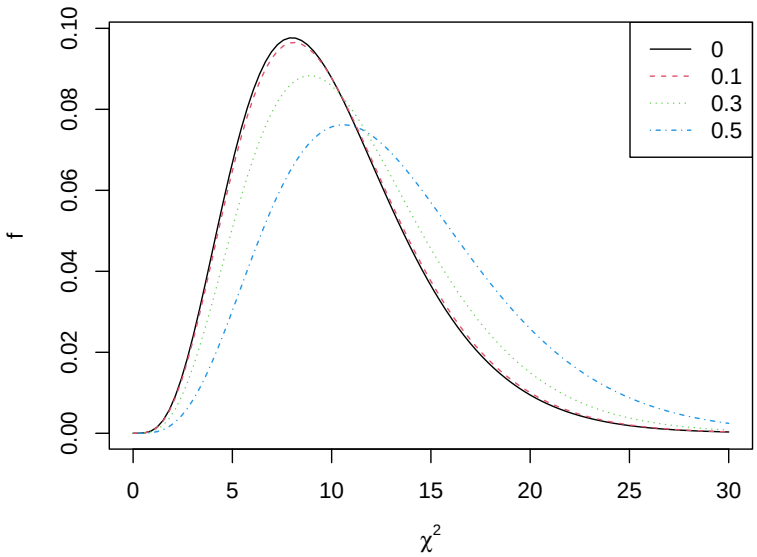
The optional arguments of `R`'s `rchisq`, `dchisq`, `pchisq` and `qchisq` functions include a 'noncentrality parameter' (`ncp`), which is related to the effect size (w) as follows:

$\texttt{ncp} = n \times w^2$

where n is the sample size. By default, the noncentrality parameter is zero, which is generally used as a null distribution. Changing `ncp` to a positive number allows the user to explore different alternative distributions.

1. Plot the PDFs of (non-central) chi-square distributions with 10 degrees of freedom (i.e. $n = 12$) and effect sizes of 0, 0.1, 0.3 and 0.5.

```r
# (a)
n <- 12
w <- c(0,0.1,0.3,0.5)
nw <- length(w)
nn <- 100
x <- seq(from=0,to=30,length.out=nn)
y <- matrix(0,nn,nw)
for (i in 1:nw){
y[,i] <- dchisq(x,df=n-2,ncp=n*w[i]^2)
}
matplot(x,y,type='l',xlab=expression(chi^2),ylab='f')
legend('topright',legend=w,lty=1:nw,col=1:nw)
```



1. What is the probability of committing a type-II error for $w \in \{0.1, 0.3, 0.5\}$ with $\alpha = 0.05$?
2. What is the probability of committing a type-II error for the same values of w when $n = 1000$?

```
# (b)
type2 <- function(n,w){
# Calculate the 95-precentile of the central chi-square distribution:
cutoff <- qchisq(0.95,df=n-2)
# Calculate the probability of exceeding this value for different values of w:
for (i in 2:length(w)){
prob <- pchisq(cutoff,df=n-1,ncp=n*w[i]^2)
message('p(type-2 error | w=',w[i],') = ',signif(prob,2))
}
}
type2(n,w)
```

```
## p(type-2 error | w=0.1) = 0.92
## p(type-2 error | w=0.3) = 0.88
## p(type-2 error | w=0.5) = 0.79
```

```
# (c)
type2(1000,w)
```

```
## p(type-2 error | w=0.1) = 0.92
## p(type-2 error | w=0.3) = 0.37
## p(type-2 error | w=0.5) = 0.00038
```

14 Regression

Consider the following data table:

x	$s[x]$	y	$s[y]$	$r[x,y]$
3	1	7	1	0.9
7	1	9	1	0.9
9	1	13	1	0.9

x	$s[x]$	y	$s[y]$	$r[x,y]$
12	1	14	1	0.9
14	1	19	1	-0.9

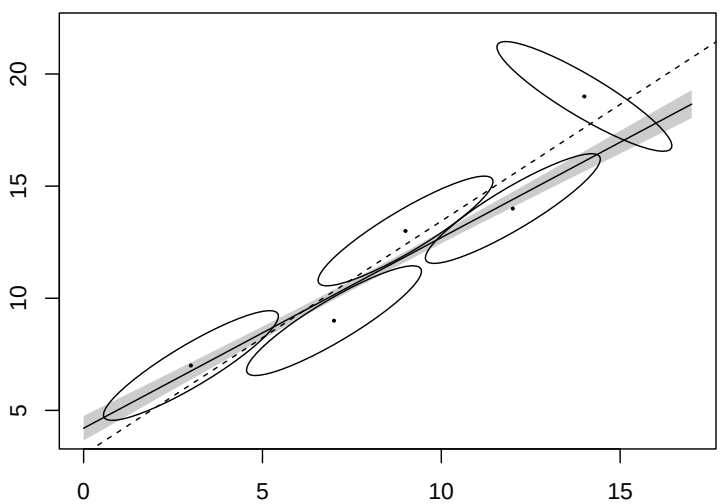
1. Fit a straight line through the x and y values, ignoring the uncertainties ($s[x]$, $s[y]$) and error correlations ($r[x,y]$). Predict a 95% confidence interval for y at $x = 20$.

```
x <- c(3,7,9,12,14)
y <- c(7,9,13,14,19)
lmfit <- lm(y ~ x)
predict(lmfit,newdata=data.frame(x=20),interval='confidence')
```

```
##        fit      lwr      upr
## 1 23.84595 17.22376 30.46813
```

2. Repeat the linear regression taking into account the analytical uncertainties, using the `geostats` package's `york()` function. Predict the y-value at $x = 20$ and estimate its standard error. Calculate the corresponding 95% confidence interval for y using a t-distribution with $n - 2 = 3$ degree of freedom. Note that `york()` does not use formula notation. See `?york` for details.

```
sx <- rep(1,5)
sy <- rep(1,5)
rxy <- c(rep(0.9,4),-0.9)
tab <- cbind(x,sx,y,sy,rxy)
yfit <- geostats::york(tab)
abline(lmfit$coefficients,lty=2)
```




```
xnew <- 20
ynew <- yfit$coef[1] + yfit$coef[2]*xnew
# error propagation formula 8.10 of the notes:
synew <- sqrt( yfit$cov[1,1] + yfit$cov[2,2]*xnew^2 + 2*xnew*yfit$cov[1,2])
df <- length(x)-2
ci <- ynew + qt(c(0.025,0.975),df=df)
out <- signif(c(ynew,ci),5)
names(out) <- c('fit','lwr','upr')
out
```

```
##    fit    lwr    upr
## 21.207 18.025 24.389
```

Conclusion: 'York regression' produces more accurate and more precise predictions.

15 Fractals

1. What is the fractal dimension of the following pattern?

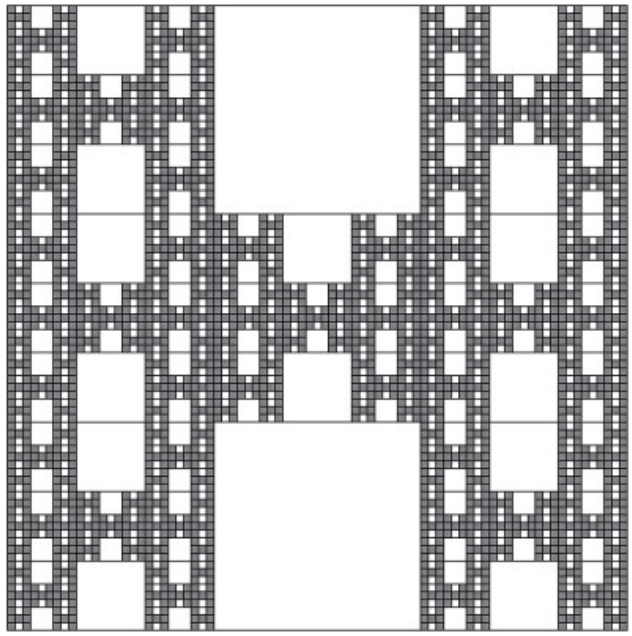


Figure 1: Rivera H-I fractal

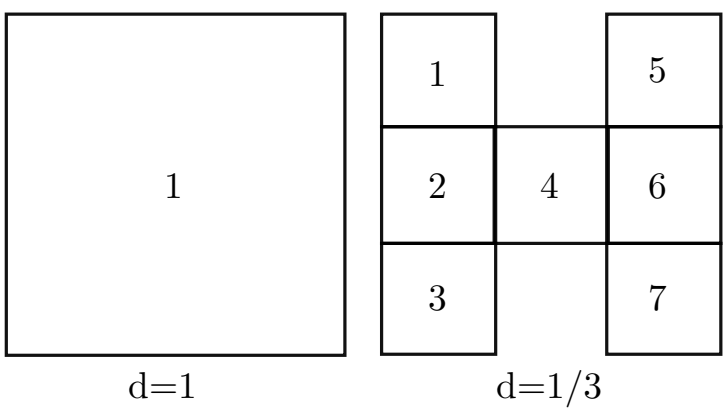
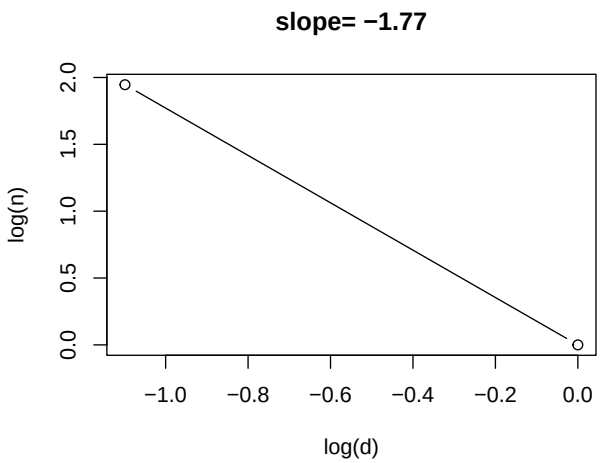


Figure 2: Box counting

1 square block with side $d = 1$ covers the entire pattern, and so do 7 square blocks of side $d = 1/3$. Hence the fractal dimension is

$$f = -\frac{\ln(7) - \ln(1)}{\ln(1/3) - \ln(1)} = \frac{\ln(7)}{\ln(3)} = 1.77$$

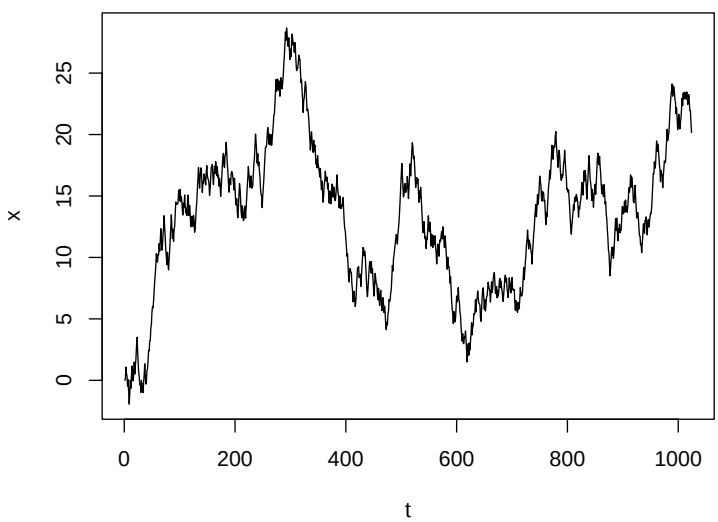
```
d <- c(1,1/3)
n <- c(1,7)
fdim <- (log(n[2])-log(n[1]))/(log(d[2])-log(d[1]))
plot(log(d),log(n),type='b',main=paste('slope=',signif(fdim,3)))
```



2. Using your code from exercise 1:

   (a) Create a random walk of 1024 steps and plot the x-position of a virtual particle against time (where time goes from 0 to 1024).

```
x <- walk(n=1024)[,1]
nx <- length(x)
t <- 1:nx
plot(t,x,type='l')
```

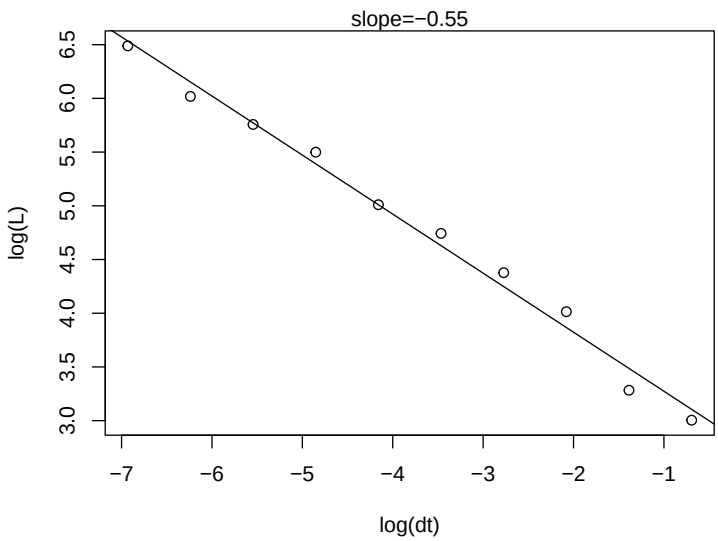


   (b) Subsample the vector of x-positions into N equally spaced segments (for $N = 2, 4, 8, 16, ..., 1024$) and add up their respective lengths.

   (c) Plot the lengths of the curve against the duration of the corresponding time steps, on a log-log plot. What is the slope?

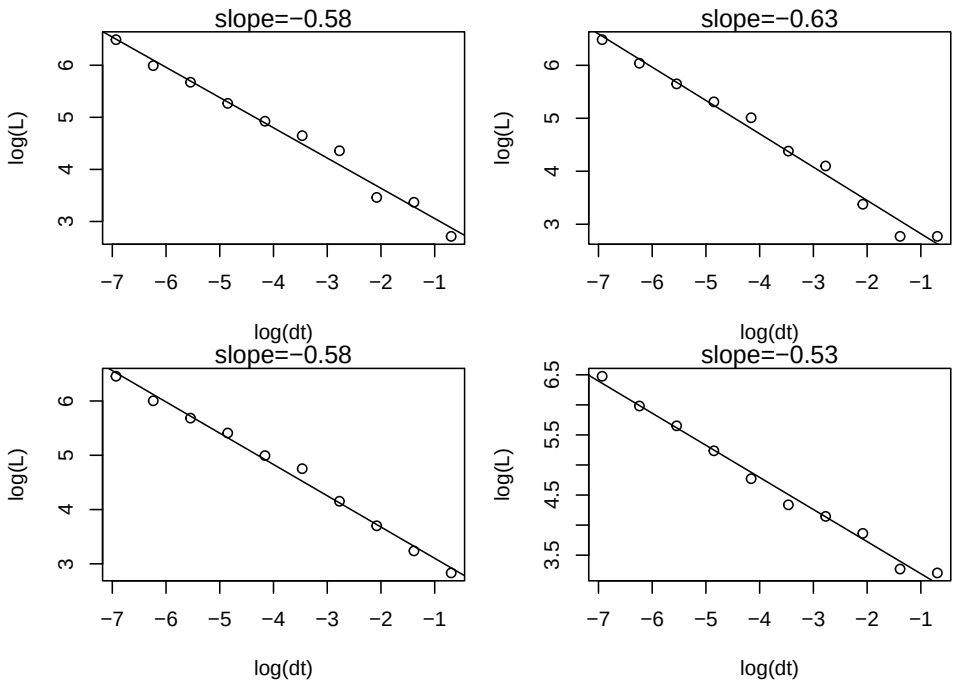
```
walklength <- function(x,N=1){
   nx <- length(x)
   i <- seq(from=1,to=nx,length.out=N+1)
   dx <- abs(diff(x[i]))
   sum(dx)
}
walkdim <- function(x){
   N <- 2^(1:10)
   nN <- length(N)
```

```
   L <- rep(0,nN)
   for (i in 1:nN){
      L[i] <- walklength(x,N=N[i])
   }
   dt <- 1/N
   plot(log(dt),log(L))
   fit <- lm(log(L) ~ log(dt))
   abline(fit)
   slope <- fit$coefficients[2]
   mtext(paste0('slope=',signif(slope,2)))
}
walkdim(x)
```



(d) Repeat steps (a)–(c) several times to assess the robustness of the result.

```
par(mfrow=c(2,2),mar=c(4,4,1,1))
for (i in 1:4){
   x <- walk(n=1024)[,1]
   walkdim(x)
}
```



16 Unsupervised learning

Consider three multivariate normal distributions with the following mean vectors and covariance matrices:

$$\mu_X = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \mu_Y = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 0 \\ 0 \end{bmatrix}, \mu_Z = \begin{bmatrix} 4 \\ 0 \\ 0 \\ -2 \\ -2 \end{bmatrix}, \Sigma_X = \Sigma_Y = \Sigma_Z = \begin{bmatrix} 5 & 1 & 1 & 1 & 1 \\ 1 & 5 & 2 & 2 & 2 \\ 1 & 2 & 5 & 3 & 3 \\ 1 & 2 & 3 & 5 & 4 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

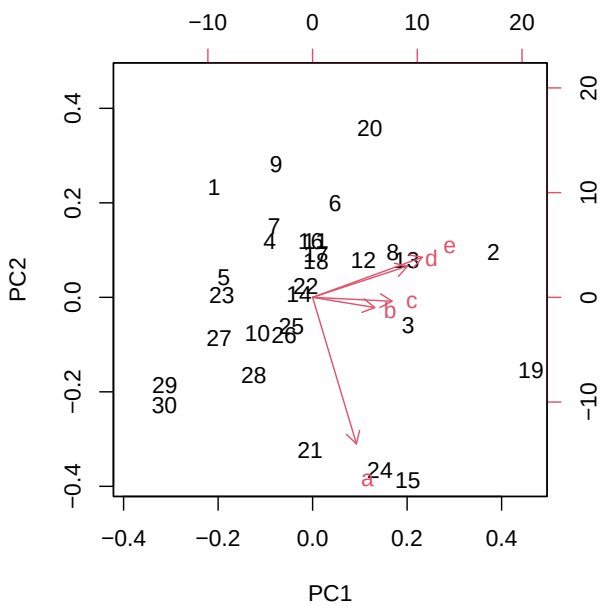
1. Draw 10 sets of random values from each of these 3 distributions and store them in a matrix. Name the variables a, b, c, d and e.

```
n <- 10
muX <- rep(0,5)
muY <- c(2,2,2,0,0)
muZ <- c(4,0,0,-2,-2)
S <- rbind(c(5,1,1,1,1),
           c(1,5,2,2,2),
           c(1,2,5,3,3),
           c(1,2,3,5,4),
           c(1,2,3,4,5))
library(MASS)
# defining a function because we will use this code again in a later exercise:
XYZ <- function(n,muX,muY,muZ,S){
   X <- mvrnorm(n=n,mu=muX,Sigma=S)
   Y <- mvrnorm(n=n,mu=muY,Sigma=S)
   Z <- mvrnorm(n=n,mu=muZ,Sigma=S)
   out <- rbind(X,Y,Z)
   colnames(out) <- c('a','b','c','d','e')
   return(out)
}
```

```
dat <- XYZ(n,muX,muY,muZ,S)
```

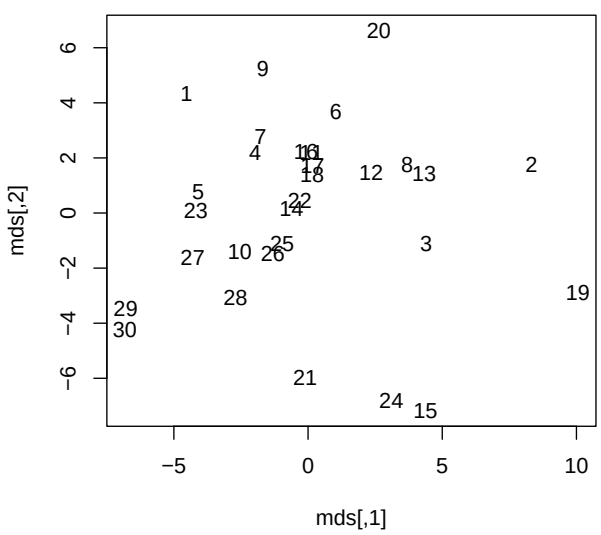
2. Analyse the dataset by principal component analysis and present the results as a biplot.

```
pc <- prcomp(dat)
biplot(pc)
```



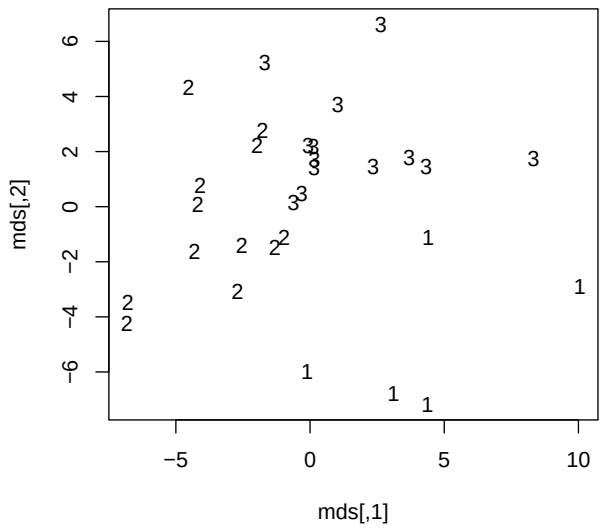
3. Analyse the dataset by classical multidimensional scaling. Plot the results as a scatter plot.

```
mds <- cmdscale(dist(dat))
plot(mds,type='n')
text(mds,labels=1:(3*n))
```



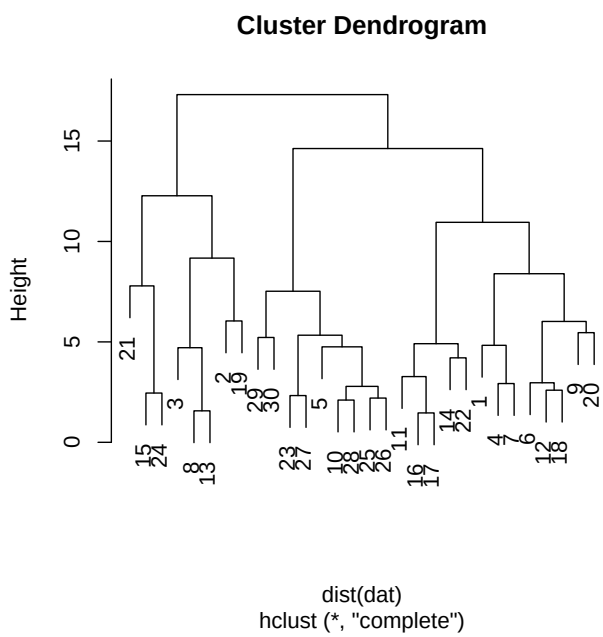
4. Classify the dataset by k-means clustering with $k = 3$. Use the results to add text labels to the MDS configuration. How successful was the clustering?

```
fit <- kmeans(dat,centers=3)
plot(mds,type='n')
text(mds,labels=fit$cluster)
```



5. Create a hierarchical dendrogram for the dataset. Would you have found the three groups?

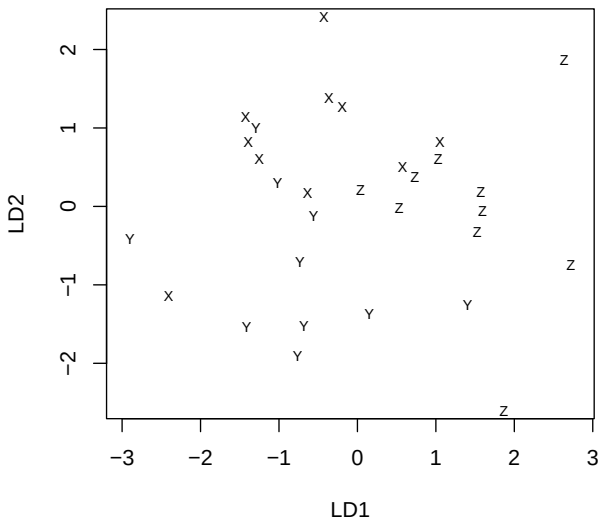
```
tree <- hclust(dist(dat))
plot(tree)
```



17 Supervised learning

1. Create a 30-element vector with the names of the known classes (X, Y, Z) of the 3×10 samples generated in the previous exercise. Build a linear discriminant model for the data given the group names. Plot the first two linear discriminants on a scatter plot.

```
library(MASS)
groups <- c(rep('X',n),rep('Y',n),rep('Z',n))
ldat <- data.frame(groups=groups,dat)
ld <- lda(groups ~ ., data=ldat)
plot(ld)
```



2. What is the misclassification rate of the LDA using the training data? Create a new dataset of 3×10 samples from the multivariate distributions of Exercise 16. Classify the new data using the linear discriminants generated in the previous step. What is the misclassification rate of the test data?

```
table(predict(ld)$class,groups)
```

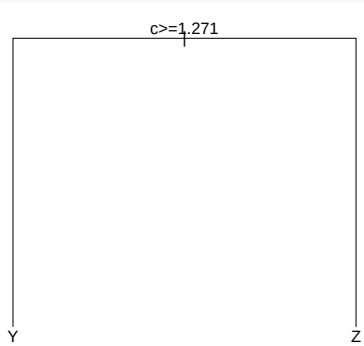
```
##    groups
##     X Y Z
##   X 7 2 1
##   Y 1 7 0
##   Z 2 1 9
```

```
ndat <- data.frame(groups=groups,XYZ(n,muX,muY,muZ,S))
pred <- predict(ld,newdata=ndat)
table(pred$class,groups)
```

```
##    groups
##     X Y Z
##   X 4 0 1
##   Y 1 8 2
##   Z 5 2 7
```

3. Build a decision tree using the same training data as before. How many branches does the default tree have?

```
library(rpart)
tree <- rpart(groups ~ ., data=ldat, method='class')
plot(tree)
text(tree,xpd=NA)
```

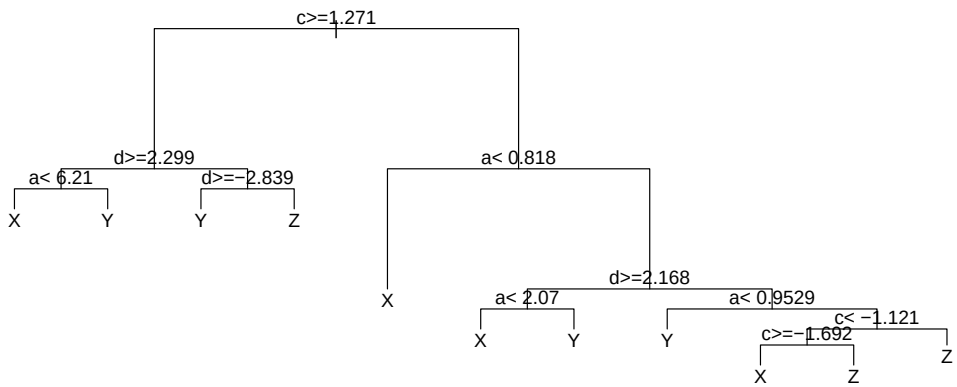


4. If your tree is too small then this is probably because, by default, `rpart` aims to have at least 20 items per branch. You can override this default setting with the optional `control` argument to the `rpart` function. Using the `iris` dataset to illustrate this:

```
my.control <- rpart.control(minsplit=1)
iristree <- rpart(Species ~ ., data=iris, method="class", control=my.control)
```

Apply the same procedure to your mixture of multivariate normal distributions.

```
my.control <- rpart.control(minsplit=1)
tree <- rpart(groups ~ ., data=ldat, method="class", control=my.control)
plot(tree)
text(tree,xpd=NA)
```



5. The `rpart.control` function can also be used to change other settings as well. For example, you can use it to build a maximum sized tree with 'pure' nodes:

```
my.control <- rpart.control(cp=0,minsplit=1)
iristree.unpruned <- rpart(Species ~ ., data=iris,
                           method="class", control=my.control)
```

You can then retrieve a nested set of subtrees by inspecting the cost-complexity table as follows (try it!):

```
printcp(iristree.unpruned)
plotcp(iristree.unpruned)
```

Finally, you can prune the tree using the, guess what, `prune` function. For example:

```
iristree.pruned <- prune(iristree, cp=.08)
```

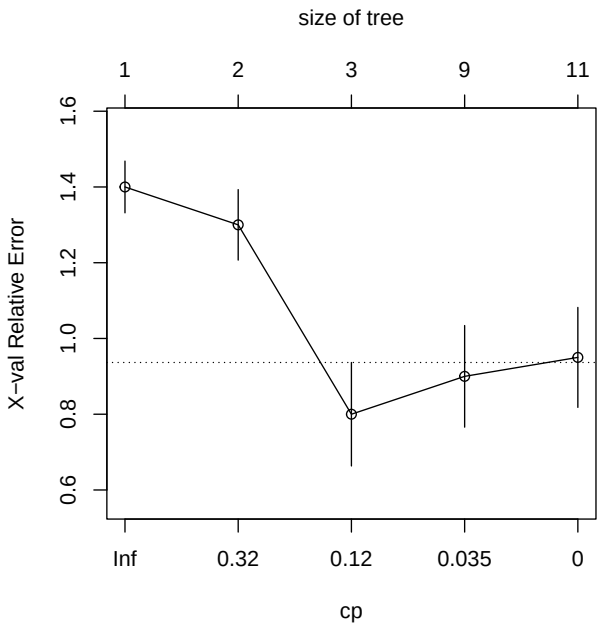
Apply the above functions to your multivariate normal mixture.

```
my.control <- rpart.control(cp=0,minsplit=1)
tree.unpruned <- rpart(groups ~ ., data=ldat, method="class", control=my.control)
printcp(tree.unpruned)
```

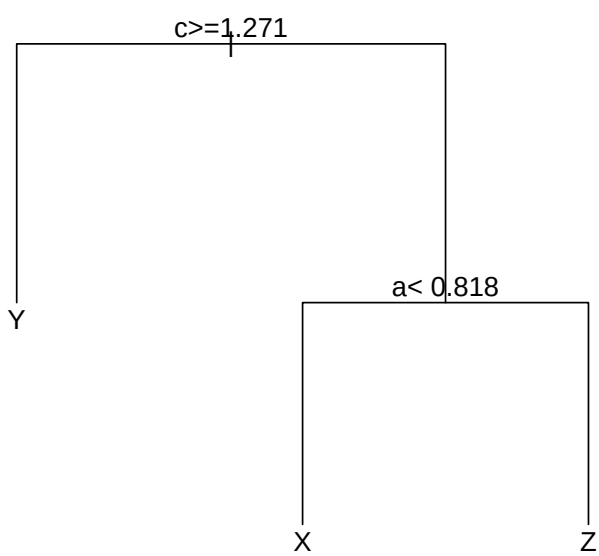
```
##
## Classification tree:
## rpart(formula = groups ~ ., data = ldat, method = "class", control = my.control)
##
## Variables actually used in tree construction:
## [1] a c d
##
## Root node error: 20/30 = 0.66667
##
```

```
## n= 30
##
##      CP nsplit rel error xerror     xstd
## 1 0.350      0      1.00   1.40 0.068313
## 2 0.300      1      0.65   1.30 0.093095
## 3 0.050      2      0.35   0.80 0.136626
## 4 0.025      8      0.05   0.90 0.134164
## 5 0.000     10      0.00   0.95 0.131972
```

```
plotcp(tree.unpruned)
```




```
tree.pruned <- prune(tree.unpruned, cp=.13)
plot(tree.pruned)
text(tree.pruned,xpd=NA)
```

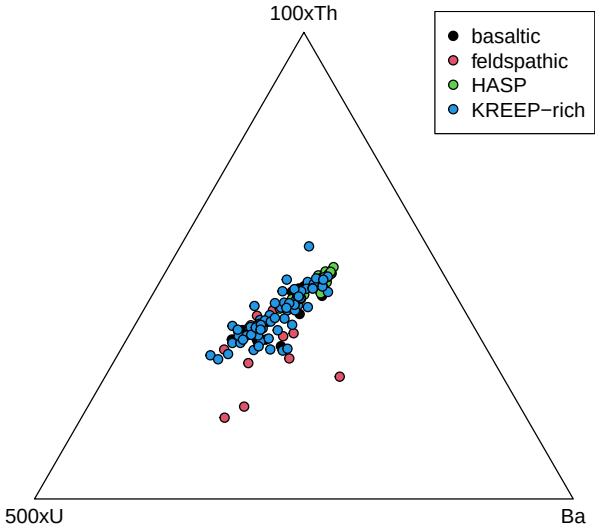


18 Compositional data

Table 2 of Nemchin et al. (2022, **https://doi.org/10.1016/j.gca.2021.12.013**) presents the major and trace element composition of 139 lunar spherules from the Apollo 14 mission. I have extracted the trace element columns from this table for you as a csv file, which you can download **here**.

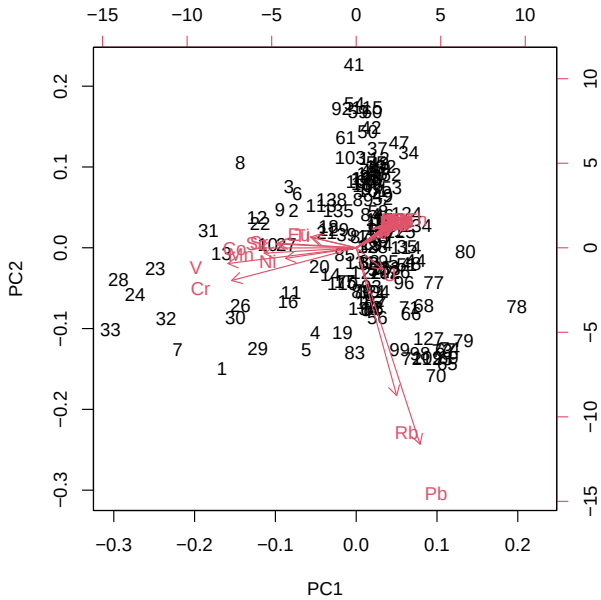
1. Plot the data on a ternary diagram containing Th, U and Ba. Colour code the samples by group ('basaltic', 'feldspathic', ...).

```
library(geostats)
dat <- read.csv('Nemchin2022trace.csv',stringsAsFactors=TRUE)
ternary(dat[,c('Th','U','Ba')],f=c(100,500,1),pch=21,bg=dat$group)
groups <- unique(dat$group)
legend('topright',legend=groups,pch=21,pt.bg=groups)
```



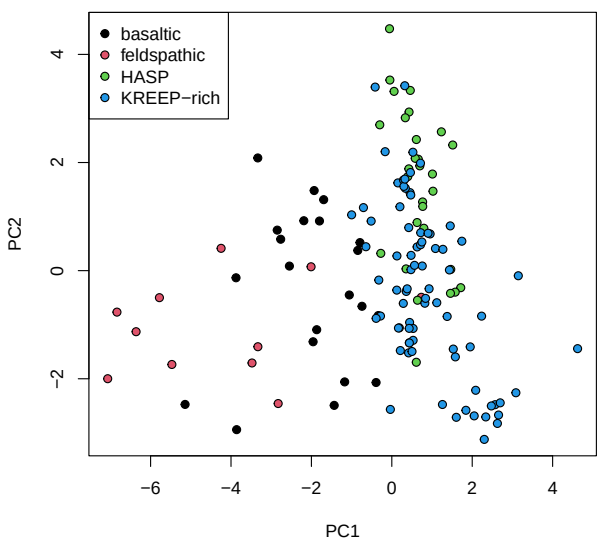
2. Analyse the dataset by compositional PCA.

```
trace_elements <- dat[,-c(1,2)]
trace_comp <- clr(trace_elements)
pc <- prcomp(trace_comp)
# plot the results as a biplot
biplot(pc)
```



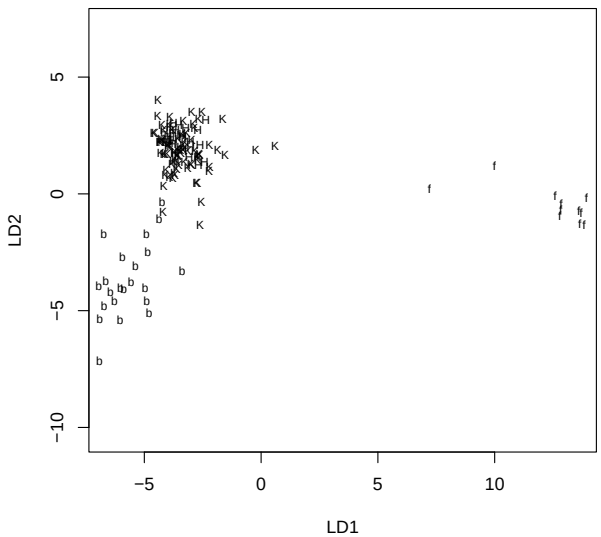
Alternatively, you can also show the first two principal components on a colour-coded scatter plot:

```
plot(pc$x,pch=21,bg=dat$group)
legend('topleft',legend=groups,pch=21,pt.bg=groups)
```



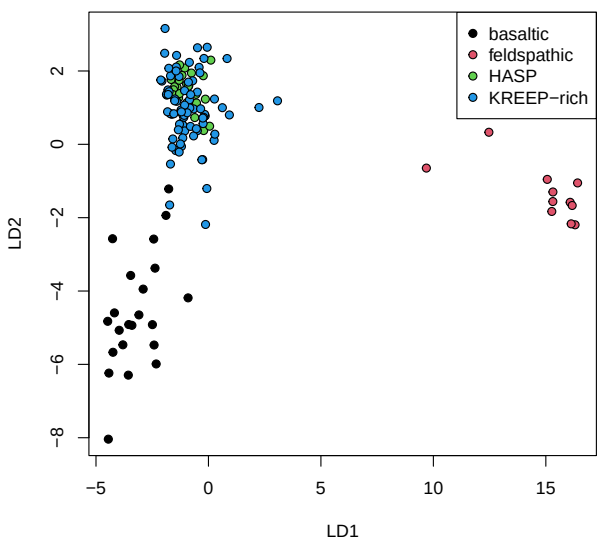
3. Analyse the lunar spherule data by LDA.

```
train_trace_comp <- data.frame(group=dat$group,alr(trace_elements))
ld <- MASS::lda(group ~ ., data=train_trace_comp)
plot(ld,dimen=2,abbrev=TRUE)
```



You can also plot the results on a scatter plot using the output of the `predict()` function applied to the output of the `lda()` function:

```
plot(predict(ld)$x[,1:2],pch=21,bg=dat$group)
legend('topright',legend=groups,pch=21,pt.bg=groups)
```



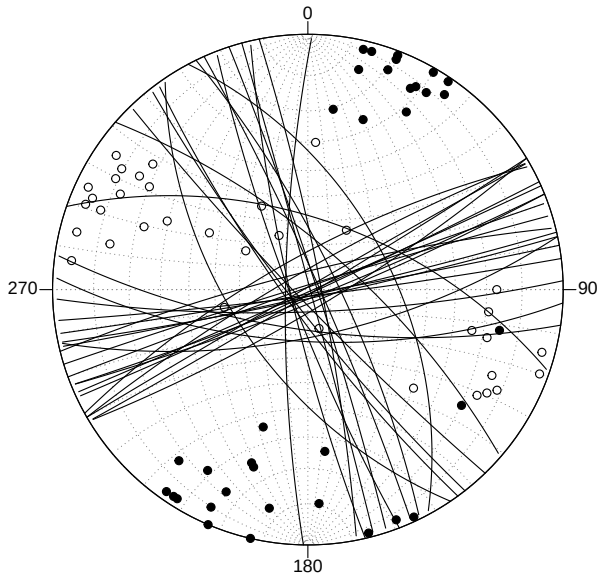
19 Directional data

Table 7 of Ali and Badrelin (2019, **https://doi.org/10.1007/s00024-019-02262-9**) presents earthquake focal mechanism parameters for Aswan, Egypt. You can also download the data **here**.

1. Plot the data on a Wulff equal angle diagram, marking the P- and T-axis orientations as white and black circles, respectively, and the strike and dip measurements as great circles.

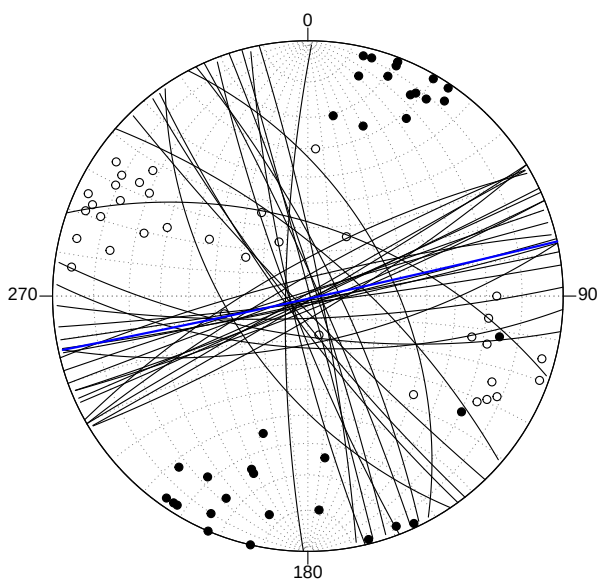
```
library(geostats)
ddat <- read.csv('Ali2019.csv',check.names=FALSE)
stereonet(trd=ddat[,'P-trd'],plg=ddat[,'P-plg'],option=1,
          degrees=TRUE,pch=21,bg='white')
stereonet(trd=ddat[,'T-trd'],plg=ddat[,'T-plg'],option=1,
          degrees=TRUE,pch=21,bg='black',add=TRUE)
```

```
stereonet(trd=ddat[,'str'],plg=ddat[,'dip'],option=2,
          degrees=TRUE,pch=NA,add=TRUE)
```



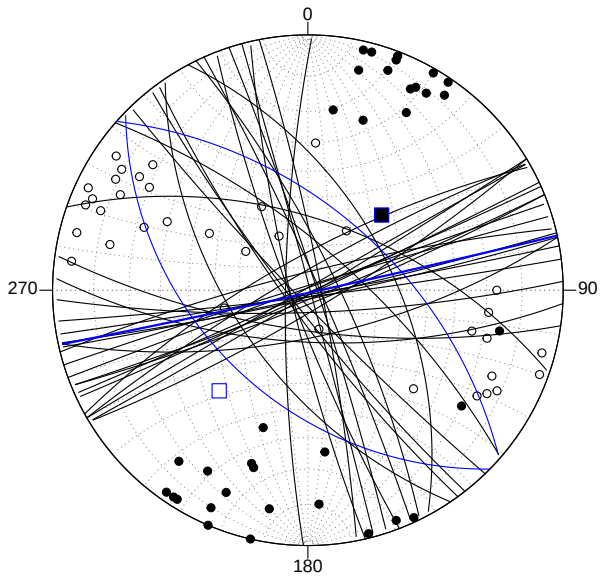
2. Calculate the average strike and dip of the first nodal plane. Add this line to the stereoplot.

```
mangle <- meanangle(trd=ddat[,'str'],plg=ddat[,'dip'],option=2,degrees=TRUE)
stereonet(trd=mangle[1],plg=mangle[2],option=2,lwd=2,
          degrees=TRUE,pch=NA,col='blue',add=TRUE)
```



3. Add the average orientation of the P- and T-axes to the existing plot. Do you notice any problems?

```
meanP <- meanangle(trd=ddat[,'P-trd'],plg=ddat[,'P-plg'],option=1,degrees=TRUE)
meanT <- meanangle(trd=ddat[,'T-trd'],plg=ddat[,'T-plg'],option=1,degrees=TRUE)
stereonet(trd=meanP[1],plg=meanP[2],option=2,degrees=TRUE,
          cex=2,pch=22,col='blue',bg='white',add=TRUE)
stereonet(trd=meanT[1],plg=meanT[2],option=2,degrees=TRUE,
          cex=2,pch=22,col='blue',bg='black',add=TRUE)
```



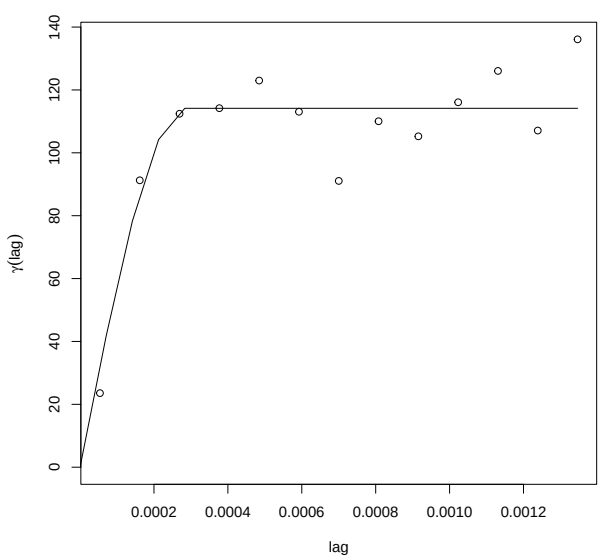
The mean azimuth and dip of the P- and T-axes do not plot in their respective data clouds. These data are regarded as nondirectional axes rather than vectors. For information about this situation, and a solution, see pages 334-337 of Davis, 2002 ('Statistics and Data Analysis in Geology', as mentioned in the Introduction chapter of the notes).

20 Spatial data

The Supplementary Information of Baird et al. (2021,**https://doi.org/10.3389/feart.2021.669440**)] contains a data table with soil depth measurements (in cm) in the apron of a blowout dune on the Gong He Plateau in China. You can download this dataset **here**.

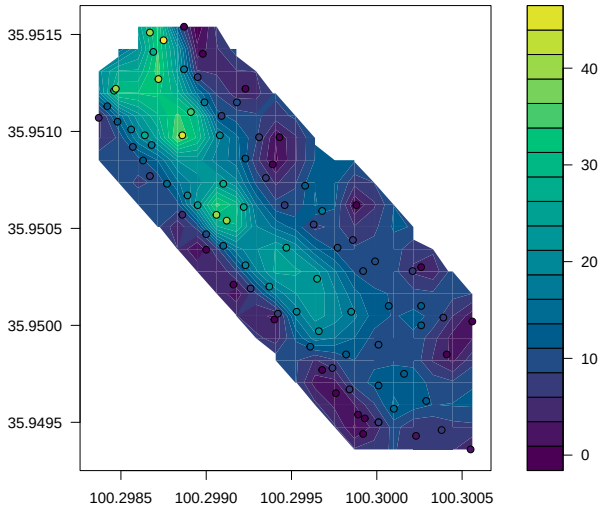
1. Reproduce Figure 8 of the paper by kriging interpolation of the soil depth measurements.

```
pits <- read.csv('Baird2021.csv')
resolution <- 20
svm <- semivariogram(x=pits$lon,y=pits$lat,z=pits$depth)
```




```
xi <- seq(from=min(pits$lon),to=max(pits$lon),length.out=resolution)
```

```r
yi <- seq(from=min(pits$lat),to=max(pits$lat),length.out=resolution)
zi <- kriging(x=pits$lon,y=pits$lat,z=pits$depth,svm=svm,xi=xi,yi=yi,grid=TRUE)
colourplot(x=xi,y=yi,z=zi,X=pits$lon,Y=pits$lat,Z=pits$depth)
```



2. Estimate the approximate volume of the apron by evaluating the soil thickness along a regular grid of points, and using the relationship that 1 degree latitude equals 111 km, and 1 degree of longitude equals $(111 \times \cos[lat])$ km.

```r
dx <- (xi[2]-xi[1])*pi/180 # pixel width in radians
dy <- (yi[2]-yi[1])*pi/180 # pixel height in radians
mperrad <- 111e3*180/pi    # metres per radian of latitude
mx <- dx*cos(meanangle(yi))*mperrad # pixel width in metres
my <- dy*mperrad                    # pixel width in metres
vol <- sum(zi,na.rm=TRUE)*mx*my/100 # volume in m^3
message('the volume of the apron is', signif(vol,3),' m^3')
```

```
## the volume of the apron is784 m^3
```