# Data reduction and (geo) statistics

Pieter Vermeesch
University College London
Gower Street, London WC1E 6BT
United Kingdom
p.vermeesch@ucl.ac.uk

## Contents

## 1 Introduction

According to the Oxford dictionary of English, the definition of 'statistics' is:

*The practice or science of collecting and analysing numerical data in large quantities, especially for the purpose of* ***inferring*** *proportions in a whole from those in a representative* ***sample****.*

The words 'inferring' and 'sample' are written in bold face because they are really central to the practice and purpose of Science in general, and Geology as a whole. For example:

1. The true proportion of quartz grains in a sand deposit is *unknown* but can be *estimated* by counting a number of grains from a respresentative sample.
2. The true crystallisation age of a rock is unknown but can be estimated by measuring the $^{206}$Pb/$^{238}$U-ratio in a representative number of U-bearing minerals from that rock.
3. The true $^{206}$Pb/$^{238}$U-ratio of a U-bearing mineral is unknown but can be estimated by repeatedly measuring the ratio of $^{206}$Pb- and $^{238}$U- ions extracted from that mineral in a mass spectrometer.
4. The spatial distribution of arsenic in groundwater is unknown but can be estimated by measuring the arsenic content of a finite number of water wells.

Thus, pretty much everything that we do as Earth Scientists involves statistics in one way or another. This module will introduce you to some basic principles of statistics that you may already have learned about in another module. But it will highlight some peculiarities about 'geological' data that represent ever so many potential pitfalls for 'conventional' statistics. The module takes a hands-on approach, using a popular statistical programming language called `R` (Section 2). Before we can quantitatively infer population properties from a sample, it is helpful to first have a look at the raw data. This is known as *exploratory data analysis* (Section 3).

## 2 An introduction to `R`

The `R` programming environment is an increasingly popular programming language (similar in scope and purpose to `Matlab`) that is available free of charge on any operating system at `http://r-project.org`. A number of different graphical user interfaces (GUIs) are available for `R`, the most popular of which are `RGui`, `RStudio`, `RCommander` and `Tinn-R`. For this tutorial, however, the simple command line console suffices.

1. First, do some arithmetic:

```
> 1 + 1
> sqrt(2)
> exp(log(10))
> 13%%5
```

2. You can use the arrow to assign a value to a variable. Note that the arrow can point both ways:

```
> foo <- 2
> 4 -> bar
> foo <- foo*bar
```

3. Create a sequence of values:

```
> myvec <- c(2,4,6,8)
> myvec*2
```

Query the third value of the vector:

```
> myvec[3]
```

Change the third value of the vector:

```
> myvec[3] <- 100
```

Change the second and the third value of the vector:

```
> myvec[c(2,3)] <- c(100,101)
```

Create a vector of 1, 2, 3, ..., 10:

```
> seq(from=1,to=10,by=1)
```

Equivalently:

```
> seq(1,10,1)
> seq(1,10)
> seq(to=10,by=1,from=1)
> seq(to=10)
> 1:10
```

Create a 10-element vector of twos:

```
> rep(2,10)
```

4. Create a $2 \times 4$ matrix of ones:

```
> mymat <- matrix(1,nrow=2,ncol=4)
```

Change the third value in the first column of `mymat` to 3:

```
> mymat[1,3] <- 3
```

Change the entire second column of `mymat` to 2:

```
> mymat[,2] <- 2
```

The transpose of `mymat`:

```
> t(mymat)
```

Element-wise multiplication (`*`) vs. matrix multiplication (`%*%`):

```
> mymat * mymat
> mymat %*% t(mymat)
```

5. Lists are used to store more complex data objects:

```
> mylist <- list(v=myvec, m=mymat, nine=9)
> mylist$v
```

6. Plot the first against the second row of `mymat`:

```
> plot(mymat[1,],mymat[2,],type='p')
```

Draw lines between the points shown on the existing plot:

```
> lines(mymat[1,],mymat[2,])
```

Create a new plot with red lines but no points:

```
> plot(mymat[1,],mymat[2,],type='l',col='red')
```

Use a 1:1 aspect ratio for the X- and Y-axis:

```
> plot(mymat[1,],mymat[2,],type='l',col='red',asp=1)
```

7. Exercise:

Plot the sine and cosine functions from 0 to $2\pi$.

Save the currently active plot as a vector-editable `.pdf` file:

```
> dev.copy2pdf(file="trigonometry.pdf")
```

8. If you want to learn more about a function, type '`help`' or '`?`':

```
> help(c)
> ?plot
```

9. You can also define your own functions:

```
> cube <- function(n){
>     return(n^3)
> }
```

Using the newly created function:

```
> cube(2)
> result <- cube(3)
```

10. Exercise:

Write a function to plot an ellipse:

$$\begin{cases} x = a\cos(t) \\ y = b\sin(t) \end{cases} \quad \text{for } 0 < t < 2\pi$$

11. Create some random (uniform) numbers:

```
> rand.num <- runif(100)
> hist(rand.num)
```

12. List all the variables in the current workspace:

```
> ls()
```

Remove all the variables in the current workspace:

```
> rm(list=ls())
```

To get and set the working directory:

```
> getwd()
> setwd("/path/to/a/valid/directory")
```

13. Collect the following commands in a file called '`myscript.R`'. Note that this text does not contain any '`>`'-symbols because it is not entered at the command prompt but in a separate text editor:

```
# the 'print' function is needed to show intermediate
# results when running commands from an .R file
print(pi)
```

You can run this code by going back to the command prompt (hence the '`>`' in the next box) and typing:

```
> source("myscript.R")
```

This should result in the number $\pi$ being printed to the console. Note that everything that follows the '`#`'-symbol was ignored by `R`.

14. Conditional statements. Add the following function to `myscript.R`:

```
toss <- function(){
    if (runif(1)>0.5){
        print("head")
    } else {
        print("tail")
    }
}
```

Save and run at the command prompt:

```
> source('myscript.R')
> toss()
```

15. Exercise:

    Write a function that takes two numbers as input and tells the user whether these are multiples of each other or not. Add this function to `myscript.R`

16. Loops. Add the following function to `myscript.R`:

```
fibonnaci <- function(n){
    if (n < 3) { stop('n must be at least 3') }
    # seed the output vector with 0 and 1:
    s <- c(0,1)
    # loop through all numbers from 3 to n:
    for (i in 3:n){
        s[i] <- s[i-1] + s[i-2]
    }
    return(s)
}
```

Save and run at the command prompt to calculate the first 20 numbers in the Fibonnaci series:

```
> source('myscript.R')
> fibonnaci(20)
```

17. Exercise:

    Write a function to print the following number triangle:
    1
    2 2
    3 3 3
    4 4 4 4
    5 5 5 5 5
    ⋮
    down to any value $n$

18. Arguably the greatest power of `R` is the availability of 10,000 *packages* that provide additional functionality. For example the `provenance` package (written by yours truly) implements a number of statistical tools for sedimentary provenance analysis. To install this package:

```
> install.packages('provenance')
```

Once installed, the package can be loaded into memory by entering:

```
> library(provenance)
```

Let's use provenance to create a ternary diagram. First, generate a 10 × 3 matrix of random numbers:

```
> comp <- matrix(runif(30),10,3)
> colnames(comp) <- c('Q','F','L')
```

Convert the numbers to an object of class ternary so that the fractions add up to unity for each row:

```
> tern <- ternary(comp)
```

Finally, plot the data on a descriptive QFL diagram, as blue circles that are magnified by a factor of 2 and do not carry any sample labels. Type ?plot.ternary for further details:

```
> plot(tern,type='QFL.descriptive',
       pch=21,bg='yellow',cex=2,labels=NA)
```

## 3 Plotting data

1. The distribution of one-dimensional data can be shown as histograms. As a first example, let's load a synthetic dataset of ostracod data into memory using the read.csv function:

```
ostracods <- read.csv(file='SEM.csv',header=TRUE)
```

ostracods is a table with three columns named 'rounded', 'elongate' and 'irregular', containing the number of rounded, elongated and irregularly shaped ostracods in 10 small samples of 10 ostracods each. The proportion of elongated valves can be used as a palaeosalinity indicator. We can access the elongate column as ostracods[,'elongate']. Let's plot these data as a histogram:

```
hist(ostracods[,'elongate'],main='elongated ostracods')
```

where main is an optional argument that specifies the (main) title of the plot.

2. Load the USGS earthquake catalog from 1970-2013 into memory and plot the earthquake magnitudes as a histogram:

```
EQ <- read.csv(file='earthquakes.csv',header=TRUE)
hist(EQ[,'Magnitude'],main='Earthquake magnitudes (1970-2013)')
```

You can see that the frequency of earthquakes tails off very quickly with increasing magnitude. This will be explored in more detail in a later tutorial.

3. I have prepared some auxiliary functions for you that can be loaded into memory by entering:

```
source('helper.R')
```

helper.R includes a function called countQuakes that extracts the number of earthquakes exceeding a given magnitude for each year represented in the earthquake catalog. For example:

```
m75 <- countQuakes(EQ,mag=7.5)
hist(m75,main='Magnitude > 7.5 earthquakes per year (1970-2013)')
```

This histogram looks similar to the ostracod distribution, but does not have a theoretical upper bound. Let's do the same analysis for the smaller earthquakes:

```
m6 <- countQuakes(EQ,mag=6)
hist(m6,main='Magnitude > 6 earthquakes per year (1970-2013)')
```

The default number of bins in the histogram ($k$) is calculated using *Sturges' Rule*:

$$k = \log_2(n) + 1 \tag{1}$$

where $n$ is the number of observations. We can change this value using the optional `breaks` argument:

```
hist(m6,breaks=seq(from=0,to=200,by=10),
     main='Magnitude > 6 earthquakes per year (1970-2013)')
```

Note how the distribution of $M > 6$ events is shifted towards the right relative to that of the $M > 7.5$ events, and is also more symmetrical.

4. A number of interesting datasets are built into `R`. One of these is the 'Old Faithful' dataset:

```
data(faithful)
```

It contains a two-column table with the waiting time between eruptions and the duration of the eruption for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA. Let's have a look at this dataset in a multi-panel figure. To create a $2 \times 2$ grid of plots:

```
par(mfrow=c(2,2))
```

Let's add a scatter plot of the waiting times and durations in the upper left corner, and two one-dimensional histograms in the upper right and lower left corner of the plot window:

```
plot(faithful)
hist(faithful[,'waiting'])
hist(faithful[,'eruptions'])
```

Notice that this distribution has two 'peaks'. In statistical jargon, it is said to be 'bimodal'. We can close the existing plot by entering:

```
> graphics.off()
```

at the command prompt.

5. Histograms require the data to be grouped into discrete *bins*. This is fine for *categorical* data such as ostracod counts that are naturally divided into discrete groups. But it does not work so well for *continuous* data such as the Old Faithful waiting times. The appearance of a histogram can be greatly altered by changing the number of bins, or by shifting them slightly to higher or lower values. A Kernel Density Estimate (KDE) is a continuous alternative to the histogram that reduces or eliminates these problems. A KDE is constructed as follows (Figure 1):

(a) Let $x_i$ be the measurements, where $1 \leq i \leq n$. Rank these values in order of increasing value along the horizontal axis of the plot.

(b) Add a symmetric function on top of each measurement. This could be a triangle, rectangle, or bell shape. Let the area under each curve be $1/n$, and its width (the 'bandwidth') $h$.

(c) Stack these curves on top of each other to produce one continuous curve $f$.

The mathematical formula for this procedure is:

$$f(x) = \sum_{i=1}^{n} K(x|x_i, h) \quad (2)$$

where $x_i$ is the $i^{\text{th}}$ measurement in the dataset, $n$ is the total number of measurements in the dataset, $h$ is the *bandwidth*, and $K(x|x_i, h)$ is the *kernel* function evaluated at $x$ given the measurement $x_i$ and the bandwidth $h$. To produce the KDE of the Old Faithful data:

```
kde <- density(faithful[,'waiting'])
plot(kde)
```

which can also be written as a 'one liner' as follows:

```
plot(density(faithful[,'eruptions']))
```

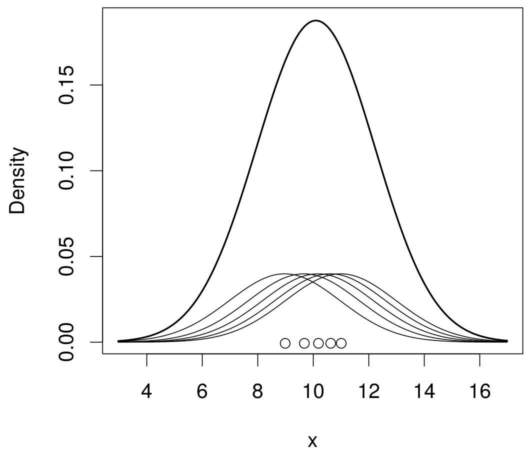

Figure 1: Construction of a Kernel Density Estimate (KDE) by stacking a (Gaussian) bell curve on top of each of five measurements, shown as circles along the horizontal axis. The area under the KDE (bold curve) is one.

6. Kernel density estimation can easily be generalised from one to two (or more) dimensions. However, to do this in `R`, we first need to load the 'Modern Applied Statistics with S[1]' package (`MASS`):

```
library(MASS)
kde2 <- kde2d(x=faithful[,'waiting'],y=faithful[,'eruptions'])
contour(kde2,xlab='waiting',ylab='eruptions')
```

See `?kde2d` and `?contour` for further details.

7. Although KDEs avoid the need to divide the data into bins, they still require the user to pick a bandwidth. Changing this value will change the appearance of the plot by *smoothing* the data to different degrees. For example:

[1] `S` is the name of the programming language that is implemented in `R`. Another implementation is the proprietary `S-PLUS` software.

```
plot(density(faithful[,'eruptions'],bw=1))
```

or:

```
plot(density(faithful[,'eruptions'],bw=0.1))
```

8. The smoothing required by KDEs (and histograms, for that matter) can be avoided by plotting the data as an empirical cumulative distribution fuction (ecdf). This is generated by:

   (a) Ranking the measurements in order of increasing value along the horizontal axis of the plot (i.e. $x_1 < \ldots < x_i < \ldots < x_n$).

   (b) Plotting the rank order (or *quantiles*) of each measurement as the vertical coordinate, in steps of $1/n$.

   The mathematical formula for this procedure can be written as:

$$F(x) = \sum_{i=1}^{n} 1(x_i < x)/n \tag{3}$$

   where $1(*) = 1$ if $*$ is 'True' and $1(*) = 0$ if $*$ is 'False'. In R:

```
plot(ecdf(faithful[,'waiting']))
```

9. The quantiles can also be obtained independently from the ecdf function. For example, to calculate the 0.9 quantile (a.k.a. the 90% *percentile*):

```
quantile(faithful[,'waiting'],prob=0.9)
```

   which returns a value of 86 minutes, meaning that 90% of the waiting times between Old Faithful eruptions are less than 86 minutes long. Go ahead and verify this graphically on the ecdf. The 50-percentile is also known as the *median* and the difference between the 75- and 25-percentiles as the *interquartile range* (IQR):

```
IQR(faithful[,'waiting'])
```

   The median and IQR are two summary statistics, which are discussed further in the next section of this proposal.

## 4 Summary statistics

Now that we have had a look at the data in a purely qualitative way, we may want to describe them more quantitatively. There are two prime aspects that need to be considered (Figure 2):

1. Location (accuracy): the proximity of some 'average' value of the measurements to the true value of whatever we are trying to estimate.

2. Dispersion (precision): the reproducibility of our estimate.

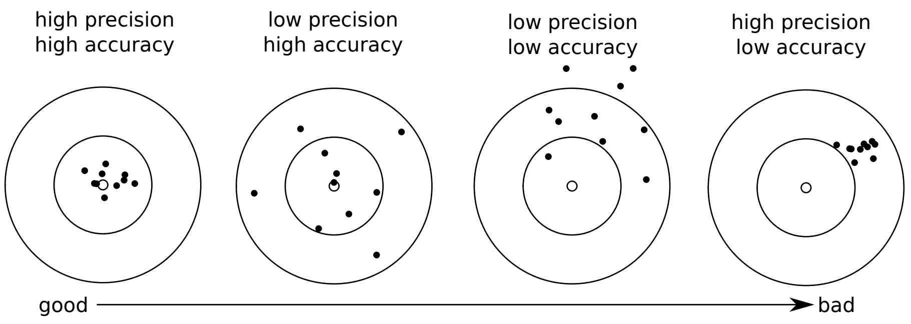

Figure 2: Accuracy and precision.

## 4.1 Accuracy and precision

There are several ways to calculate the location and dispersion of a dataset:

Location:

$$\text{arithmetic mean: } (x_1 + x_2 + \ldots + x_n)/n \tag{4}$$

$$\text{geometric mean: } \sqrt[n]{x_1 \times x_2 \times \ldots x_n} = \exp\left(\Sigma_{i=1}^n \ln[x_i]/n\right) \tag{5}$$

$$\text{harmonic mean: } n/\Sigma_{i=1}^n (1/x_1 + 1/x_2 + \ldots + 1/x_n) \tag{6}$$

$$\text{median: } x_1 < \ldots < \mathbf{x_{n/2}} < \ldots < x_n \tag{7}$$

$$\tag{8}$$

Dispersion:

$$\text{arithmetic standard deviation: } \sqrt{\sum_{i=1}^n (x_i - \bar{x})^2/n} \tag{9}$$

$$\text{geometric standard deviation: } \exp\left\{\sqrt{\sum_{i=1}^n (\ln[x_i] - \ln[\bar{x}])^2/n}\right\} \tag{10}$$

$$\text{mean absolute deviation: } \sum_{i=1}^n |x_i - \bar{x}|/n \tag{11}$$

$$\text{median absolute deviation: } |x_1 - \bar{x}| < \ldots < |\mathbf{x_{n/2}} - \bar{\mathbf{x}}| < \ldots < |x_n - \bar{x}| \tag{12}$$

$$\tag{13}$$

where $\bar{x}$ is a pre-specified location parameter. In R, using the earthquake example:

```
> mean(m6)   # mean
> sd(m6)     # standard deviation
> median(m6) # median
> mad(m6)    # median absolute deviation
```

Exercise: Write R functions to calculate the harmonic mean, the mean absolute deviation, the geometric mean and geometric standard deviation.

Another location measure is the *mode* of the distribution. For discrete data, this marks the value that appears most often. For continuous data, the mode corresponds to the highest point on the KDE, or the steepest point on the ecdf:

```
kde <- density(faithful[,'waiting'])
imax <- which.max(kde$y)
themode <- kde$x[imax]
```

where `which.max(...)` returns the index of the maximum KDE value. To see the full data structure of the `kde` object, type:

```
> str(kde)
```

at the command prompt. You will see that it is a list of vectors including `x` and `y`. See also `?density` for additional details. Finally, the *box-and-whisker* plot is a final summary diagram that jointly visualises the minimum (0-percentile), 25-percentile, median (50-percentile), 75-percentile and maximum (100-percentile):

```
boxplot(faithful)
```

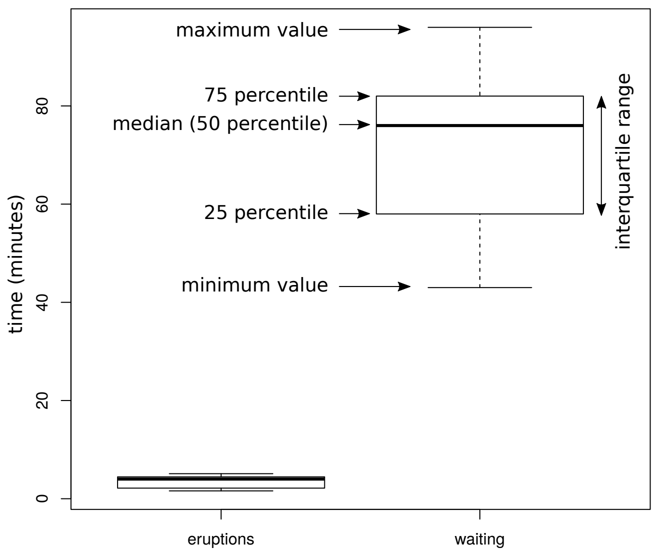

Figure 3: Box-and-whisker plot of the Old Faithful data. The whiskers mark the range of the data (minimum and maximum), while the box outlines the interquartile range (middle 50 percent of the data). The median (50-percentile) is drawn as a horizontal line inside the box. Diagrams like this are particularly useful when comparing multiple sample distributions.

## 4.2 Which average to use?

With so many options to quantify the location and dispersion, all of which generally yield different values, the question arises which of these is correct. The answer to this is: “it depends”. People (including Earth Scientists) are most familiar with the arithmetic mean and standard deviation. However, this is not always the best option. To illustrate the dangers of blindly applying the arithmetic mean to any data, consider another built-in dataset:

```
data(mtcars)
```

which returns a table with the fuel consumption and 10 aspects of automobile design and performance for 32 automobiles features in a 1974 edition of ‘Motor Trend’ magazine. Let us calculate the average fuel consumption of these 32 vehicles –which are listed in miles per gallon– using the arithmetic mean:

```
avg.mpg <- mean(mtcars[,'mpg'])
```

which returns a value of ∼20.09 mpg. Now let’s do the same calculation in European units, namely litres per 100km. To convert from mpg to l/100km:

```
litres <- 235.21/mpg
```

The arithmetic mean of these values is given by

```
avg.litres <- mean(litres)
```

returning a value of 11.71 l/100km. Now let's convert this value back to mpg:

```
avg.mpg.from.litres <- 235.21/avg.litres
```

which is ... $18.44 \neq 20.09$! In other words: the average fuel consumption depends on the measurement units used. This is deeply worrying.

Exercise: Use the function that you wrote in the previous exercise to show that the geometric mean fuel consumption is the same regardless of the measurements units used.

Exercise: A car drives 50 km at 50 km/h followed by 50 km at 100 km/h. What is the average speed over the total distance of 100 km driven? Which of our average functions is appropriate here? Can you use this function to calculate the average speed of the 50 cars in `R`'s built-in `cars` dataset?

If you wonder about the relevance of this example for the Earth Sciences, then consider some other examples. How would you average the sedimentation rate in different sedimentary successions? Or what is the density of a mixture of 5 g magnetite (5.2 g/cm$^3$) and 5 g quartz (2.6 g/cm$^3$)?

One of the principal objectives of this module is to caution against the inappropriate use of elementary statistical operations such as averaging.

## 5 Probability Distributions

A probability distribution is a mathematical function that describes the likelihood of observing a particular value of some property in a randomly selected sample from a larger population. The function space of probability distributions is infinite. In fact, the next paragraph will show that it is easy to invent one's own probability distribution. Yet there are a few distributions that are so common in science that they have been given their own name. The following sections will introduce four of these distributions: the normal, binomial, poisson and fractal distributions.

### 5.1 The Normal distribution

There is one distribution that is so commonly used in statistics that it is also known as the 'normal' distribution. The purpose of this exercise is to show why the normal distribution is so popular.

1. Let us begin this section by (re)loading the auxiliary code into memory:

   ```
   source('helper.R')
   ```

   `helper.R` contains a new function called `randy()` that generates random draws from four two-dimensional distributions. `randy()` takes two arguments: `pop` is an integer number between 1 and 4 marking the population of interest, and `n` controls the number of random pairs of values to draw from that population. Given these two arguments, `randy()` returns a [2×`n`] matrix of plot coordinates. Let us use this function to plot 250 samples from the first population:

   ```
   xy <- randy(pop=1,n=250)
   plot(xy,type='p',xlab='X',ylab='Y')
   ```

2. Exercise:

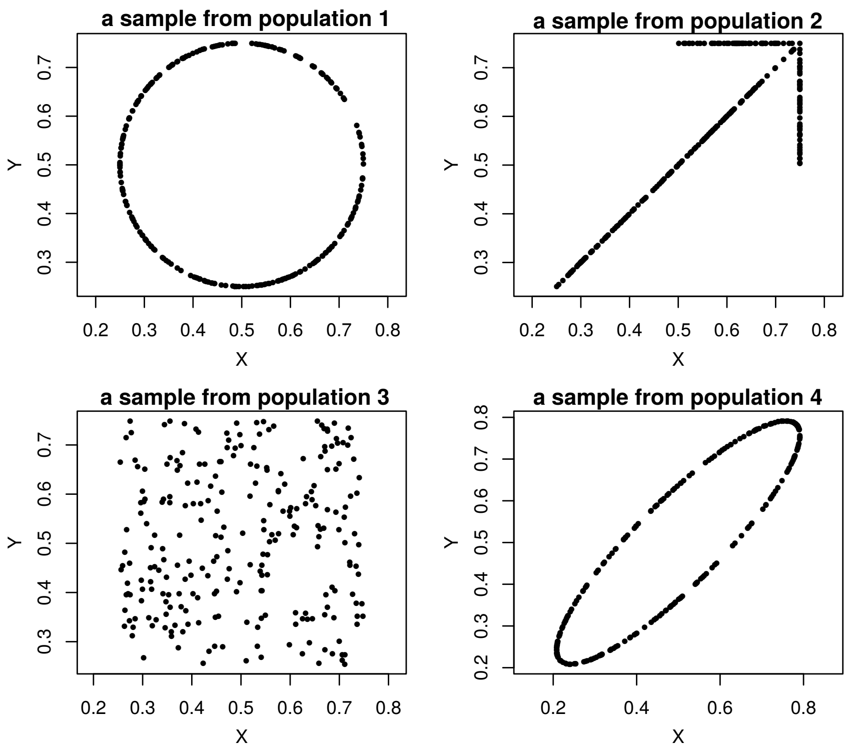

Figure 4: 250 random samples from four two-dimensional populations. Although these four distributions look completely different from each other, their sums are all adequately described by a simple bivariate normal distribution (Figure 5).

Repeat this exercise for the other three populations, and write a function to show all four populations on a single plot. Hint: use a for loop and the `mfrow` argument of the `par()` function to produce a multi-panel plot. See page 8 for details.

The output of this exercise is shown in Figure 4. It is fair to say that the four populations look very different from each other.

3. Now let us repeat this experiment multiple times, take the *sum* of the X- and the Y-values for each sample of our four populations, and plot the resulting values on a new diagram:

```
# returns ns sums of n samples from population pop:
plotSums <- function(n=100,ns=100,pop=1){
    XY <- matrix(0,nrow=ns,ncol=2) # initialise the output matrix
    for (i in 1:ns){               # compute ns sums of n values
      xy <- randy(pop=pop,n=n)     # collect n random samples
      XY[i,] <- colSums(xy)        # take the sum of the X- and Y-values
    }
    plot(XY,type='p',xlab='sum(X)',ylab='sum(Y)')
}
```

```
> plotSums(n=250,ns=200,pop=1)
```

produces the upper left panel of Figure 5.

Exercise: modify `plotSums()` to plot all four panels.

4. Despite the completely different appearance of the four parent populations (Figure 4), the distributions of their sums (Figure 5) all look very similar. They consist of an elliptical point cloud that is dense in the middle and thins out towards the edges. The density of the points per unit area is accurately described by a bivariate Gaussian distribution:

$$f(x,y) = \frac{\exp\left(-\begin{bmatrix}(x-\mu_x) & (y-\mu_y)\end{bmatrix}\begin{bmatrix}\sigma_x^2 & \sigma_{x,y}\\ \sigma_{x,y} & \sigma_y^2\end{bmatrix}^{-1}\begin{bmatrix}x-\mu_x\\ y-\mu_y\end{bmatrix}\Big/2\right)}{2\pi\sqrt{\begin{vmatrix}\sigma_x^2 & \sigma_{x,y}\\ \sigma_{x,y} & \sigma_y^2\end{vmatrix}}} \tag{14}$$

which is completely described by five parameters: the means $\mu_x$ and $\mu_y$, the standard deviations $\sigma_x$ and $\sigma_y$, and the covariance $\sigma_{x,y}$.

A one-dimensional projection of the data yields a univariate Gaussian distribution. For example, the distribution of the x-values:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma_x}\exp\left[-\frac{(x-\mu_x)^2}{2\sigma_x^2}\right] \tag{15}$$

According to the Central Limit Theorem, Equations 15 and 14 emerge whenever one takes the sum of a sufficiently large random sample, regardless of the initial parent distribution. The Gaussian distribution is very common because additive processes are common in nature. For example, the Brownian motion that underpins diffusive processes is a result of cumulative collisions that give rise to bell-shaped diffusion profiles. Because the Gaussian distribution is so common, it is also known as the *normal* distribution.

It can be shown that 95% of the x-values fall in the interval between $\mu_x \pm 1.96\sigma_x$. And similarly, 95% of the y-values fall in the interval between $\mu_y \pm 1.96\sigma_y$. Rounding the factor 1.96 up to 2 gives rise to a convenient '2-sigma' prediction interval for the data.

5. The parameters $\mu_x$, $\mu_y$, $\sigma_x^2$, $\sigma_y^2$ and $\sigma_{x,y}$ are unknown but can be *estimated* from the data as follows:

$$\begin{aligned}
\mu_x \approx \bar{x} \equiv{}& \textstyle\sum_{i=1}^n x_i/n\\
\mu_y \approx \bar{y} \equiv{}& \textstyle\sum_{i=1}^n y_i/n\\
\sigma_x^2 \approx s_x^2 \equiv{}& \textstyle\sum_{i=1}^n (x_i-\bar{x})^2/(n-1)\\
\sigma_y^2 \approx s_y^2 \equiv{}& \textstyle\sum_{i=1}^n (y_i-\bar{y})^2/(n-1)\\
\sigma_{x,y} \approx s_{x,y} \equiv{}& \textstyle\sum_{i=1}^n (x_i-\bar{x})(y_i-\bar{y})/(n-1)
\end{aligned} \tag{16}$$

where $\bar{x}$ and $\bar{y}$ are the (arithmetic) means of $x$ and $y$, respectively; and $s_x$, $s_y$ and $s_{x,y}$ are the sample (co)variances. The latter three parameters are grouped in a covariance matrix. The factor of $(n-1)$ in the definition of the (co-)variances is the number of *degrees of freedom.* One d.o.f has been removed from the (co-)variance because the means $\bar{x}$ and $\bar{y}$ are *estimated* from the data rather than *given* as in Equation 9. This inflation of the dispersion parameter is called the 'Bessel correction'.

6. The following functions add the approximate 95% boundaries for $x$ and $y$ to an existing plot:

```
# plot a 2-sigma prediction interval
plotBounds <- function(XY){
  mu <- colMeans(XY)                 # compute the means
  covmat <- cov(XY)                  # compute the covariance matrix
  sig <- sqrt(diag(covmat))          # extract the standard deviations
  lims <- rbind(mu-2*sig,mu+2*sig)  # 2-sigma bounds
  lines(lims[,1],rep(mu[1],2))      # draw the 2-sigma line for X
  lines(rep(mu[2],2),lims[,2])      # draw the 2-sigma line for Y
}
```

7. After adding the above functions to `gauss.R`, we need to modify the `plotSums()` code shown under point 3:

```
plotSums <- function(n=100,ns=100,pop=1){
  ...
  plotBounds(XY)
}
```

Re-running `plotSums(n=200,ns=250,pop=1)` at the command prompt now adds the desired prediction intervals to (the first panel of) Figure 5. The covariance between $X$ and $Y$ is given by `covmat[1,2]` in the body of the `plotBounds()` function. These values are shown in the subtitles of Figure 5. They quantify the positive correlation between the two dimensions in populations 2 and 4, and the absence of such correlation in populations 1 and 3.

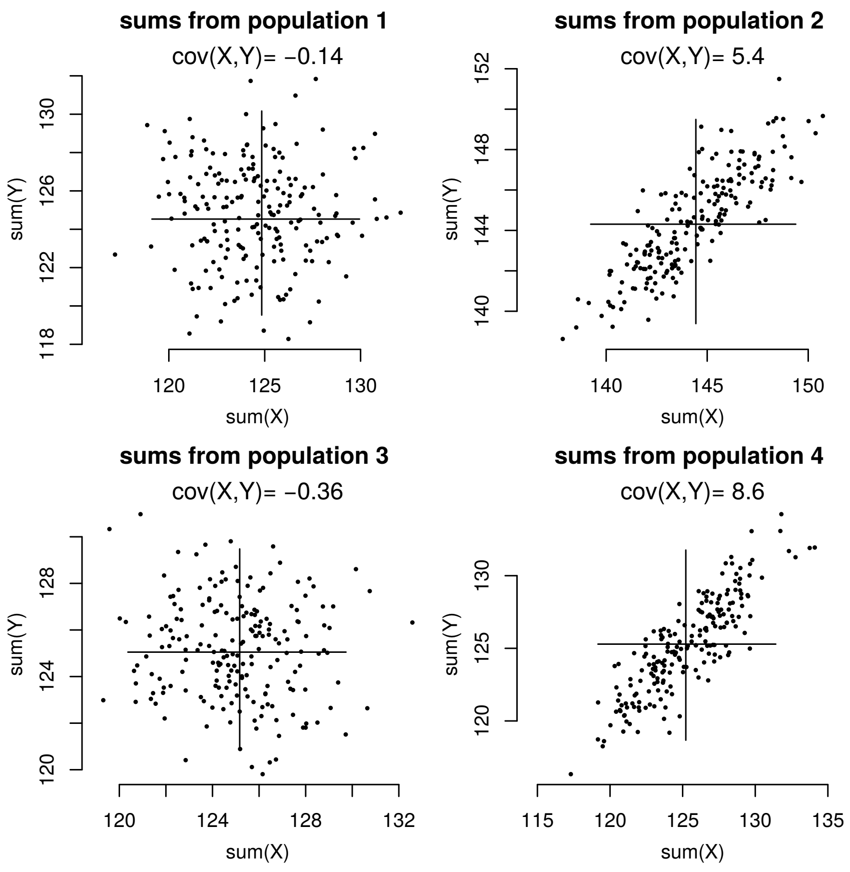

Figure 5: The sums of 200 sets of random samples like those displayed in Figure 4. Although the four distributions are completely different, their sums follow simple bivariate normal distributions. These distributions are completely described by five parameters: the means of the X- and Y-variables, their respective standard deviations, and the covariance between them. Approximately 95% of the X- and Y-variables are contained within an interval of 2 standard deviations around their respective means. These intervals are shown as crosses on each of the panels.

## 5.2 The Binomial distribution

Section 3 and 4.2 introduced some graphical and numerical methods to characterise the *sampling distributions* of fossil counts, geyser eruptions, and any other measurable quantity. An infinite number of these distributions are possible. In fact, later sections of these notes will show that you can easily make up your own distributions. However there are a number of distributions that are so common that they have been given a formal name. This section will introduce the binomial distribution, whereas the next sections will discuss the Poisson, fractal and normal distributions.

The first dataset of Section 3 contains three classes of data: rounded, elongated and irregular. Suppose that the true fraction of rounded ostracods is 20% of the total, i.e. $p_r = 0.2$. Therefore, the true fraction of 'other' shells is 80%, i.e. $p_o = 0.8$. The likelihood that a random sample of $n = 10$ ostracods does not contain a single rounded shell is:

$$P(oooooooooo) = 0.8^{10} = 0.107$$

The probability that only the first of 20 randomly selected ostracods is rounded:

$$P(roooooooooo) = 0.2^1 \times 0.8^9 = 0.0268$$

This is the same as the probability that only the second of 20 randomly selected ostracods is rounded:

$$P(oroooooooo) = 0.8 \times 0.2^1 \times 0.8^8 = 0.2^1 \times 0.8^9 = 0.0268$$

so that the probability of observing one rounded and nine other shells is

$$P(1 \times r, 9 \times o) = 10 \times 0.2^1 \times 0.8^9 = 0.268$$

The general formula for observing $k$ rounded and $n-k$ other shells in a sample of $n$ randomly selected ostracods is:

$$P(k \times r, (n-k) \times o) = \binom{n}{k} p_r^k (1-p_r)^{n-k} \quad (17)$$

where $\binom{n}{k}$ is the binomial coefficient:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad (18)$$

It returns the number of ways to arrange $k$ items among a collection of $n$. Equation 17 describes the *probability mass function* of the Binomial distribution.

1. In R, the probability that a sample of 10 ostracods contains 2 rounded shells is:

```
dbinom(x=2,size=10,prob=0.2)
```

   To plot the entire probability mass function:

```
f <- dbinom(x=0:10,size=10,prob=0.2)
barplot(f,xlab='n[r]')
```

2. Using the cumulative distribution function to calculate the probability of observing two *or fewer* rounded shells among a sample of ten randomly selected ostracods:

```
pbinom(q=2,size=10,prob=0.2)
```

   Verify that this does indeed return the probability of observing 0, 1 or 2 rounded ostracods:

```
sum(dbinom(x=0:2,size=10,prob=0.2))
```

   To plot the entire cumulative distribution function (cdf):

```
x <- seq(from=0,to=10,length.out=100)
plot(x=x,y=pbinom(q=x,size=10,prob=0.2),type='l')
```

3. The probability of observing $k$ or more rounded ostracods among a sample of $n$ under the hypothesis that the true proportion is $p_r$ is called the *p-value*. For example, consider the following 'null hypothesis':

   $H_\circ$: $p_r = 0.2$ (i.e. "the true fraction of rounded ostracods is 20%")

   and the 'alternative hypothesis':

   $H_a$: $p_r > 0.2$ (i.e. "the true fraction of rounded ostracods is greater than 20%")

   and suppose that we count $k = 5$ rounded ostracods among a sample of $n = 10$. Then the probability of observing 5 *or more* ostracods under $H_\circ$ is given by:

   ```
   p <- 1-pbinom(q=5,size=10,prob=0.2)
   ```

   The lower the p-value, the more confident we can be that the null hypothesis is false. A cutoff of $p < 0.05$ is often use to reject the null hypothesis on a '95% confidence level' and accept the alternative hypothesis instead. Please note that observing a p-value *greater* than 0.05 does *not* mean that the null hypothesis is *true*. It merely means that we do not have enough evidence to reject the null hypothesis. Increasing the sample size from $n = 10$ to $n = 100$, say, will generally lower the p-value and will eventually lead to the rejection of the null hypothesis.

   Exercise: We believe that 50% of all dinosaur fossils are female (and 50% are male). A bone bed contains 50 dinosaur fossils among which 31 are female (and 19 are male). Do we have enough evidence to prove that females are more common than males?

4. Let us now consider a second pair of hypotheses:

   $H_\circ$: $p_r = 0.2$

   and the 'alternative hypothesis':

   $H_b$: $p_r \neq 0.2$

   So we have the same null hypothesis as last time, but a different alternative hypothesis ($\neq$ instead of $>$). Suppose that we count $k = 0$ rounded ostracods among a sample of $n = 10$. The probability of observing 1 *or more* ostracods under $H_\circ$ is given by:

   ```
   > 1-pbinom(q=0,size=20,prob=0.2)
   ```

   which is $0.988 > 0.05$. So using the rule applied in the previous exercise, we would fail to reject the null hypothesis. However, this would be wrong as a very high p-value is just as unlikely to occur under the null hypothesis as a very low one.

5. In order to account for the *symmetry* of $H_b$ (as opposed to $H_a$ in point 3), we need to split the $p < 0.05$ cutoff into two 'tails'. Under such a 'two-sided hypothesis test', $H_\circ$ is rejected if $p > 0.975$ OR $p > 0.025$. Note that $1 - 0.975 = 0.025$. In our case $0.988 > 0.975$ so we reject $H_\circ$ in favour of $H_b$.

   Exercise: We believe that the proportion of male and female dinosaurs are the same. A bone bed contains 19 male and 31 female specimens. Do our data support the notion that the proportions are *different*?

   If you have done the previous two exercises, you will see that the one-sided and two-sided hypothesis tests lead to different conclusions for the same dataset. It is extremely important that you choose your test before carrying out the experiment. So you must decide on your scientific question before designing your study. In the dinosaur example, it would be bad practice to decide on a one-sided hypothesis test *after* seeing that female fossils are more abundant in the bone bed than male ones.

6. So far we have assumed that we know the true proportion of rounded ostracods in our sample. In the real world this is not the case. Instead, we would like to use our fossil counts to *estimate* the proportion of rounded ostracods. Suppose that we have observed 1 rounded ostracod shell among a sample of 10 specimens. So our best guess for the proportion of rounded shells would be 0.1 (i.e. 10%). However if we repeated this experiment a second time, then it is likely that we would obtain a different estimate. So the true proportion of rounded ostracod shells might very well be different from our current best guess of 0.1. It might be higher or it might be lower. In order to estimate the *precision* of our estimate, it is useful to explore all possible proportions that are 'likely' to result in the observed count of 1 rounded ostracod. For example:

```
> pbinom(q=1,size=10,prob=0.1)
[1] 0.7360989
> pbinom(q=1,size=10,prob=0.2)
[1] 0.3758096
> pbinom(q=1,size=10,prob=0.3)
[1] 0.1493083
> pbinom(q=1,size=10,prob=0.4)
[1] 0.0463574
```

suggesting that $0.3 < p_r < 0.4$. Rather than manually looking for the highest value of $p_r$ that would be consistent with observing 1/10 rounded ostracods, we can automate this process:

```
pr <- seq(from=0.1,to=0.9,length.out=100)
pval <- pbinom(q=1,size=10,prob=pr)
i <- which.min(abs(pval-0.05))
pr[i]
```

which returns `pr[i] = 0.3909091` as the population proportion that is closest to the cutoff value of p=0.05. Even faster and more precise results are obtained using `R's` built-in `binom.test` function:

```
ul <- binom.test(x=1,n=10,p=0.2,alternative="less",conf.level=0.95)
```

`ul` is a list of items (see `?binom.test` for details) that includes an item called `conf.int` containing the confidence interval. Typing `ul$conf.int` at the command prompt yields a 95% confidence interval of $0 < p_r < 0.3941633$. This means that the observation of 1 rounded and 9 non-rounded ostracods is most likely due to the presence of 10% rounded shells in the underlying population, but it could also have resulted from a population proportion of up to 39.4%.

7. To calculate a 'two-sided' confidence interval:

```
ul <- binom.test(x=1,n=10,p=0.2,alternative="two.sided",conf.level=0.95)
```

which returns an interval of [0.0025 – 0.4450]. This means that the probability of observing 2 or more rounded ostracods in a sample of 10 is equally unlikely (namely 2.5%) if the true proportion of rounded ostracods were 0.0025 than the likelihood of observing 2 or fewer ostracod shells if the true proportion were 0.4450.

## 5.3 The Poisson distribution

1. The previous section has shown how the number of counts from a population with two classes of data follows a Binomial distribution. Let us continue on this theme and consider the likelihood of observing two or fewer detrital diamonds in a sample of 4,000,000 randomly selected sand grains from a population that contains 1/1,000,000$^{\text{th}}$ of diamonds and 999,999/1,000,000$^{\text{ths}}$ of other minerals:

```
> pbinom(q=2,size=4e6,prob=1/1e6)
[1] 0.2381032
```

2. Recall that the Binomial distribution is described by two parameters (`size` and `prob`). When the probability of success (`prob`) is very low and the number of samples is very large (`size`), then the Binomial distribution converges to a *Poisson distribution*, which is described by a single parameter, $\lambda$:

$$P(k|\lambda) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{19}$$

```
> ppois(q=2,lambda=4)
[1] 0.2381033
```

where `lambda` is the expected number of counts in the 4,000,000 grain sample. The Poisson distribution applies when the actual probability distribution is given by a binomial distribution and the number of trials is sufficiently bigger than the number of successes one is asking about.

3. Let us now generate 1,000 random numbers from a Poisson distribution with distribution parameter `lambda=4`:

```
rnum <- rpois(n=1e3,lambda=4)
average <- mean(rnum)
variance <- var(rnum)
```

You will see that the mean and variance (i.e., the squared standard deviation) are approximately equal to `lambda`. This is a diagnostic signature of a Poisson distributed variable.

4. Doing the same for the earthquake dataset:

```
m75 <- countQuakes(EQ,mag=7.5)
m75avg <- mean(m75)
m75var <- var(m75)
```

Verify that `m75avg` $\approx$ `m75var`.

5. Exercise:

Take `m75avg` as our best estimate for the population parameter. What is the likelihood of observing at least one earthquake of magnitude 7.5 or greater next week?

Hint: the equivalent of the `dbinom`, `pbinom` and `qbinom` functions for the Poisson distributions are called `dpois`, `ppois` and `qpois`. Use the built-in documentation for further details.

6. Note: the Poisson approximation to earthquake occurrences is a reasonable assumption for large earthquakes, which occur *independently* from each other. But it does not work so well for smaller magnitudes. This is because large earthquakes increase the probability of similar to weaker aftershocks. You can verify this by repeating exercise 4 above for smaller magnitudes.

## 5.4 Fractal distributions

1. Let us count the number of earthquakes exceeding a range of magnitudes between 6 and 7.5, and plot them against those magnitudes on a log-log scale:

```
n <- 20
mags <- seq(from=6,to=7.5,length.out=n)
counts <- rep(0,n)
for (i in 1:n){
    counts[i] <- sum(countQuakes(EQ,mag=mags[i]))
}
plot(x=mags,y=counts,log='xy')
```

   The linear fit of this plot and resulting power law relationship between the magnitude and frequency of earthquakes is called the Gutenberg-Richter Law. It is caused by the fact that faulting is a *fractal process*: small faults (which cause small earthquakes) are more abundant than large ones (which produce large earthquakes).

2. Let's create a synthetic fractal using the `sierpinski()` function of `helper.R`:

```
source('helper.R')
g <- sierpinski()
plot(g,pch=20,axes=FALSE,xlab='',ylab='')
```

   This pattern is called a *Sierpinski carpet*. It is similar to the spatial distribution of faults and fractures in that the frequency of empty patches is strongly dependent on their size. Small empty patches are more abundant than large ones, just like small faults are more abundant than large ones. If we plot the number of empty patches exceeding a given size against that size, then this would obey a power law. A more practical way of obtaining the power law relationship is by using the *box counting* method. This is done by counting the number of square boxes that are needed to cover all the black dots in the Sierpinski carpet against their size.

3. To do this, first install the `fractaldim` package:

```
> install.packages('fractaldim')
```

   The log-log plot is then generated as follows:

```
library(fractaldim)
fd.estim.boxcount(g,plot.loglog=TRUE,nlags=10)
```

4. Exercise:

The average number of magnitude $> 6$ earthquakes in our dataset is 94/year. The average number of magnitude $> 7$ earthquakes is 9.4/year. What is the expected number of earthquakes of magnitude $> 8$? Assuming that the number of earthquakes of magnitude $> 8$ follows a Poisson distribution, what is the likelihood of observing at least two such events next year? What is the likelihood of a magnitude 9 or greater event happening next year?

## 6 Error propagation

Error propagation is an essential part of any analytical measurement. For example, suppose that the extinction of the dinosaurs has been dated at 65 Ma in one field location, and a meteorite impact has been dated at 64 Ma elsewhere. These two numbers are effectively meaningless in the absence of an estimate of precision. Taken at face value, the dates imply that the meteorite impact took place 1 million years after the mass extinction, which rules out a causal relationship between the two events. If, however, the analytical uncertainty is significantly greater than 1 Myr (e.g. 64 ± 2 Ma and 65 ± 2 Ma), then such of a causal relationship remains very plausible. Suppose that the age or any other a physical quantity ($z$) is calculated as a function ($f$) of some measurements ($x$):

$$z = f(x) \tag{20}$$

then $\mu_x$, $\sigma_x^2$ can be estimated (as $\bar{x}$ and $s_x^2$) from the input data ($x_i$, for $i = 1...n$) using Equations 4 and 9. These values can then be used to infer $\sigma_z^2$, the variance of the calculated value $z$, a process that is known as 'error propagation'.

### 6.1 Linear approximation

Recall the definition of the sample standard deviation and variance (Equation 9):

$$s_z^2 \equiv \frac{1}{n-1}\sum_{i=1}^{n}(z_i - \overline{z})^2 \tag{21}$$

If the function $f$ is (approximately) linear in the vicinity of $\bar{x}$, then $(z_i - \overline{z})$ is (approximately) proportional to $(x_i - \overline{x})$ (Figure 6.1):

$$z_i - \overline{z} \approx (x_i - \overline{x})\frac{\partial z}{\partial x} \tag{22}$$

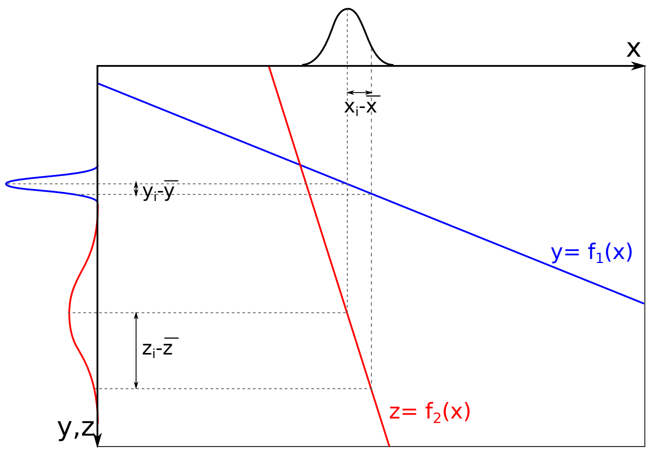

Figure 6: Error propagation of two linear functions. The uncertainties of $y$ and $z$ are proportional to the derivative (slope) of these functions w.r.t. the input data $x$. Because $f_2$ has a steeper slope than $f_1$, the spread of $z$ is greater than that of $y$.

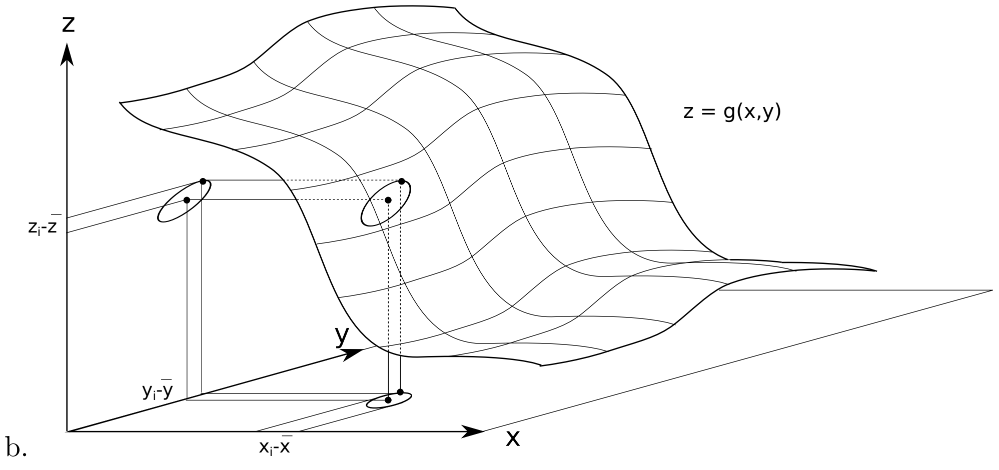

Figure 7: b. Error propagation of a bivariate function. The uncertainty in $z$ is *approximately* proportional to the slope of the surface $g$ w.r.t. the measurements $x$ and $y$.

where $\partial z/\partial x$ is the local slope, i.e. the first derivative of the function $f$. Plugging Equation 22 into 21, we obtain:

$$s_z^2 \approx \frac{1}{n-1}\sum_{i=1}^{n}\left[(x_i-\overline{x})\frac{\partial z}{\partial x}\right]^2 \tag{23}$$

$$= \left[\frac{\partial z}{\partial x}\right]^2\left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\overline{x})^2\right] = \left[\frac{\partial z}{\partial x}\right]^2 s_x^2 \tag{24}$$

This is the general equation for the propagation of uncertainty with one variables. Next, let us move on to two variables. Suppose that the age or any other physical quantity ($z$) is calculated as a function ($g$) of some measurements ($x$ and $y$):

$$z = g(x,y) \tag{25}$$

and further suppose that $x$ and $y$ follow a bivariate Normal distribution, then $\mu_x$, $\mu_y$, $\sigma_x^2$, $\sigma_y^2$ and $\sigma_{x,y}$ can all be estimated (as $\bar{x}, \bar{y}, s_x^2, s_y^2$ and $s_{x,y}$) from the input data ($x_i$,$y_i$, for $i = 1...n$). These values can then be used to infer $\sigma_z^2$, the variance of the calculated value $z$ in exactly the same way as the 1D case. Differentiating $g$ with respect to $x$ and $y$:

$$z_i - \overline{z} \approx (x_i-\overline{x})\frac{\partial z}{\partial x} + (y_i-\overline{y})\frac{\partial z}{\partial y} \tag{26}$$

Plugging Equation 26 into Equation 21:

$$s_z^2 \approx \frac{1}{n-1}\sum_{i=1}^{n}\left[(x_i-\overline{x})\left(\frac{\partial z}{\partial x}\right) + (y_i-\overline{y})\left(\frac{\partial z}{\partial y}\right)\right]^2 \tag{27}$$

$$= ... = s_x^2\left(\frac{\partial z}{\partial x}\right)^2 + s_y^2\left(\frac{\partial z}{\partial y}\right)^2 + 2\ s_{x,y}\frac{\partial z}{\partial x}\frac{\partial z}{\partial y} \tag{28}$$

This is the general equation for the propagation of uncertainty with two variables, which is most easily extended to more than two variables by reformulating Equation 28 into a matrix form:

$$s_z^2 = \left[\begin{array}{cc} \frac{\partial z}{\partial x} & \frac{\partial z}{\partial y} \end{array}\right] \left[\begin{array}{cc} s_x^2 & s_{x,y} \\ s_{x,y} & s_y^2 \end{array}\right] \left[\begin{array}{c} \frac{\partial z}{\partial x} \\ \frac{\partial z}{\partial y} \end{array}\right] \tag{29}$$

where the innermost matrix is known as the *variance-covariance* matrix and the outermost matrix (and its transpose) as the *Jacobian matrix*. Let us now apply this equation to some simple functions:

1. addition:

$$z = ax + by \Rightarrow \frac{\partial z}{\partial x} = a, \frac{\partial z}{\partial y} = b \tag{30}$$

$$\Rightarrow s_z^2 = a^2 s_x^2 + b^2 s_y^2 + 2ab\ s_{x,y} \tag{31}$$

2. subtraction:

$$z = ax - by \Rightarrow \cdots \Rightarrow s_z^2 = a^2 s_x^2 + b^2 s_y^2 - 2ab\ s_{x,y} \tag{32}$$

3. multiplication:

$$z = axy \Rightarrow \frac{\partial z}{\partial x} = ay, \frac{\partial z}{\partial y} = ax \tag{33}$$

$$\Rightarrow s_z^2 = (ay)^2 s_x^2 + (ax)^2 s_y^2 + 2a^2 xy\ s_{x,y} \tag{34}$$

$$\Rightarrow \left(\frac{s_z}{z}\right)^2 = \left(\frac{s_x}{x}\right)^2 + \left(\frac{s_y}{y}\right)^2 + 2\frac{s_{x,y}}{xy} \tag{35}$$

4. division:

$$z = a\frac{x}{y} \Rightarrow \cdots \Rightarrow \left(\frac{s_z}{z}\right)^2 = \left(\frac{s_x}{x}\right)^2 + \left(\frac{s_y}{y}\right)^2 - 2\frac{s_{x,y}}{xy} \tag{36}$$

5. exponentiation:

$$z = a\ \exp(bx) \Rightarrow \frac{\partial z}{\partial x} = ab\ \exp(bx) \Rightarrow \left(\frac{s_z}{z}\right)^2 = b^2 s_x^2 \tag{37}$$

6. logarithms:

$$z = a\ \ln(bx) \Rightarrow \frac{\partial z}{\partial x} = \frac{a}{x} \Rightarrow s_z^2 = a^2\left(\frac{s_x}{x}\right)^2 \tag{38}$$

7. power:

$$t = ax^b \Rightarrow \frac{\partial f}{\partial x} = b\frac{ax^b}{x} \Rightarrow \left(\frac{s_t}{t}\right)^2 = b^2\left(\frac{s_x}{x}\right)^2 \tag{39}$$

Ignoring the covariance terms greatly simplifies the error propagation if more than two terms are involved. This is obviously only valid if the covariance terms are small compared to the variances.

Exercises:

1. The height difference between Ordnance Survey benchmark A and a second point B is 25±0.25 m, and the height difference between B and a third point C is 50 ± 0.5 m. What is the height difference between A and C?

2. Consider the following two rock specimens:

| specimen | mass (g) | $\sigma$(mass) | volume (cm$^3$) | $\sigma$(volume) |
|---|---|---|---|---|
| A | 105 | 2 | 36 | 0.15 |
| B | 30 | 2.5 | 10 | 0.4 |

   (a) Compute the densities of rock A and B.
   (b) Propagate their respective uncertainties.
   (c) Construct 95% confidence intervals for the two densities.
   (d) Is there a significant difference in density between samples A and B?

3. Sand may consist of a variety of minerals with different densities. Zircon has a density of $4.85 \pm 0.10$ g/cm$^3$ ($1\sigma$), whereas quartz has a density of $2.65 \pm 0.05$ g/cm$^3$ ($1\sigma$). Because they are denser, zircon grains settle more quickly in water than quartz grains of the same size. Equivalently, small zircons settle at the same rate as large zircon grains. This results in a *size shift* between zircon and quartz in beach and river sands:

$$SS = \frac{1}{0.69}\ln\left(\frac{\rho_z - 1}{\rho_q - 1}\right)$$

where $\rho_z$ is the density of zircon, $\rho_q$ is the density of quartz, and $SS$ is the size shift ($SS \equiv \log_2(D_z/D_q)$ where $D_z$ = diameter of zircon grains and $D_q$ = diameter of quartz grains). Calculate $SS$ and propagate its uncertainty.

## 6.2 Accuracy vs. precision

Recall the definition of the arithmetic mean (Section 4):

$$\overline{x} \equiv \sum_{i=1}^{n} x_i/n$$

Applying the equation for the error propagation of a sum (Equation 31):

$$s_{\overline{x}}^2 = \sum_{i=1}^{n} s_{x_i}^2/n^2 = n\frac{s_x^2}{n^2} = \frac{s_x^2}{n} \tag{40}$$

where we assume that all n measurements were done *independently*, so that $s_{x_i,y_j} = 0$ for all $i, j$. The standard deviation of the mean is known as the <u>standard error</u>:

$$s_{\overline{x}} = \frac{s_x}{\sqrt{n}} \tag{41}$$

This means that the standard error of the mean monotonically decreases with the square root of sample size. In other words, we can arbitrarily increase the *precision* of our analytical data by acquiring more data. However, it is important to note that the same is generaly not the case for the *accuracy* of those data. The difference between precision and accuracy is best explained by the darts board analogy of Figure 2.

Whereas the analytical precision can be computed from the data using the error propagation formulas introduced above, the only way to get a grip on the accuracy is by analysing another sample of independently determined value. Such test samples are also known as 'secondary standards'.

<u>Exercise:</u>

We measured the following $^{40}$K and $^{40}$Ar concentrations:

| | | | | | | |
|---|---|---|---|---|---|---|
| $^{40}$K ($\times 10^{-10}$ mol/g) | 2,093 | 2,105 | 2,055 | 2,099 | 2,030 | |
| $^{40}$Ar ($\times 10^{-10}$ mol/g) | 6.015 | 6.010 | 6.030 | 6.005 | 6.020 | 6.018 |

1. Calculate the mean $^{40}$K and $^{40}$Ar concentrations.
2. Calculate the standard error of these means.
3. Estimate the $^{40}$Ar/$^{40}$K-ratio and its standard error.
4. The age equation for the $^{40}$K-$^{40}$Ar method is as follows:

$$t = \frac{1}{\lambda}\ln\left[1 + \frac{\lambda}{\lambda_e}\left(\frac{^{40}Ar}{^{40}K}\right)\right]$$

with $\lambda = 5.543 \times 10^{-4}$ Myr$^{-1}$ and $\lambda/\lambda_e = 9.3284$. Based on the results from the previous steps, calculate the age and its analytical uncertainty.

# 7 Linear regression

Let $x$ and $y$ be two correlated variables with standard deviations $\sigma_x$ and $\sigma_y$ and covariance $\sigma_{x,y}$. Pearson's correlation coefficient is defined as:

$$\rho_{x,y} = \frac{\sigma_{x,y}}{\sigma_x \sigma_y} \tag{42}$$

$0 < \rho_{x,y} \leq 1$ indicates a positive correlation, $\rho_{x,y} = 0$ no correlation and $-1 \leq \rho_{x,y} > 0$ a negative correlation. $\rho_{x,y}$ is generally unknown but can be *estimated* from data. Consider two paired sets of measurements $x = \{x_1, ..., x_i, ..., x_n\}$ and $y = \{y_1, ..., y_i, ..., y_n\}$. Using the definitions of Equation 16, the sample correlation $r_{x,y}$ is given by:

$$r_{x,y} = \frac{s_{x,y}}{s_x s_y} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{43}$$

## 7.1 Least squares estimation

For reasonably large correlation coefficients ($|r_{x,y}| > 0.8$, say), we can try and fit a straight line through the data (Figure 8):

$$y_i = a + bx_i + \epsilon_i \tag{44}$$

where $a$ is the intercept, $b$ is the slope and $\epsilon_i$ are the *residuals* (Figure 8). The best linear fit to the data is given by those values of a and b that minimise the *misfit*. The most common definition of 'misfit' is the sum of the squared residuals:

$$ss \equiv \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n}(a + bx_i - y_i)^2 \tag{45}$$

This formulation gives optimal results when the residuals are normally distributed *with zero mean*:

$$\epsilon_i \sim N(0, \sigma^2)$$

In that case, minimising Equation 45 is equivalent to maximising Equation 15:

$$\begin{aligned} \max_{a,b}\left[\prod_{i=1}^{n} f(\epsilon_i)\right] &= \max_{a,b}\left[\sum_{i=1}^{n}\ln\{f(\epsilon_i)\}\right] = \max_{a,b}\left[\ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \sum_{i=1}^{n}\left(\frac{\epsilon_i^2}{2\sigma^2}\right)\right] \\ &= \max_{a,b}\left[-\sum_{i=1}^{n}\epsilon_i^2\right] = \min_{a,b}\left[\sum_{i=1}^{n}\epsilon_i^2\right] = \min_{a,b}(ss) \end{aligned} \tag{46}$$

Equation 45 can be solved analytically, by taking the first derivatives of $ss$ w.r.t a and b and setting them to zero.

$$\begin{cases} \frac{\partial ss}{\partial a} = 2\sum(a + bx_i - y_i) = 0 \\ \frac{\partial ss}{\partial b} = 2\sum(a + bx_i - y_i)x_i = 0 \end{cases} \tag{47}$$

Solving this system of equations, it can be shown that:

$$\begin{cases} a = \left(\sum x_i \sum y_i - \sum x_i \sum x_i y_i\right) / \left(n\sum x_i^2 - \left(\sum x_i\right)^2\right) \\ b = \left(n\sum x_i y_i - \sum x_i \sum y_i\right) / \left(n\sum x_i^2 - \left(\sum x_i\right)^2\right) \end{cases} \tag{48}$$

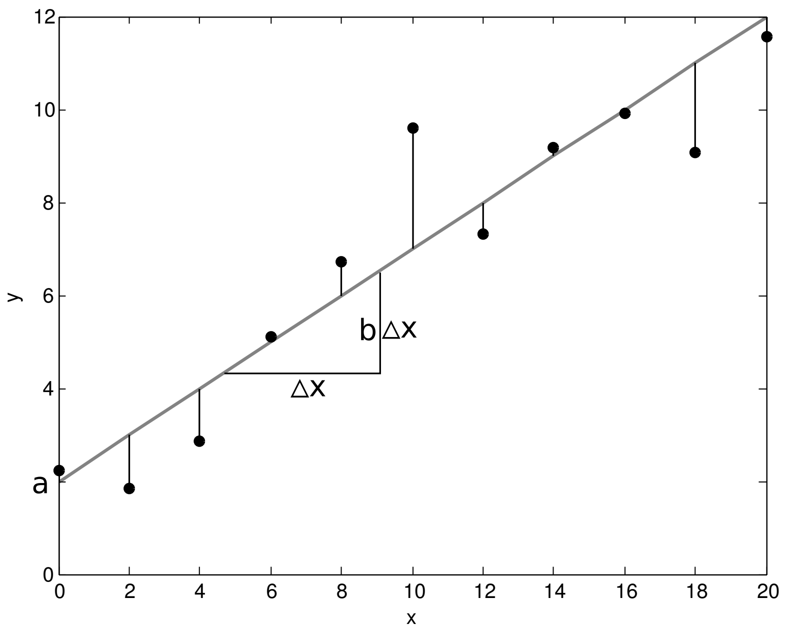

Figure 8: Linear regression with intercept a and slope b. Filled circles mark the data poins ($x_i$,$y_i$), the best fit line is shown in grey and the vertical lines mark the residuals $\epsilon_i$, for i=1→11

1. Load the `RbSr.csv` file into memory:

```
RbSr <- read.csv(file='RbSr.csv',header=TRUE)
```

   and investigate the column names:

```
> colnames(RbSr)
```

   `RbSr` contains two columns called `Rb87Sr86` and `Sr87Sr86`, respectively. Plot these two columns against each other:

```
plot(RbSr,type='p')
```

2. Calculate the covariance matrix and correlation matrix of the data:

```
covmat <- cov(RbSr)
cormat <- cor(RbSr)
```

   The off-diagonal term of `covmat` contains the covariance $s_{x,y}$, whereas the off-diagonal term of `cormat` contains the correlation coeffient $r_{x,y}$. The correlation coefficient is a sort of 'normalised' covariance. As shown by Equations 43, it is easy to calculate the correlation matrix from the covariance matrix. Verify this:

```
> cov2cor(covmat)
```

   produces the same output as `cormat`.

3. Exercise:

   Write an `R`-function to calculate the sum of squares ($ss$, Equation 45). Use this function to manually explore different values of the intercept ($a$) and slope ($b$) for the `RbSr` dataset. Which values produce the lowest values for $ss$?

4. Fit a linear model to the Rb-Sr data using `R`'s `lm()` function:

```
fit <- lm(Sr87Sr86 ~ Rb87Sr86, data=RbSr)
```

To view the coefficients of the fit:

```
> coef(fit)
(Intercept)    Rb87Sr86
 0.69915186  0.06481581
```

This means that $a = 0.69915186$ and $b = 0.06481581$ in Equation 44.

Exercise: Write an `R`-function that implements Equation 48 and show that it produces the same results as the built-in `lm()` function.

5. The uncertainties (standard errors) of the fit parameters can be obtained by:

```
covmat <- vcov(fit)
```

This returns the variance-covariance matrix of the intercept and slope. Do you notice the strong correlation between the uncertainties of these two parameters? Use the `cov2cor()` function to obtain the corresponding correlation coefficient. This value quantifies the correlation between the uncertainties of $a$ and $b$. This is something completely different from the correlation coefficient of the data that we obtained in step 2!

6. Let us add the best fit line to our plot:

```
x <- seq(from=0,to=max(RbSr[,'Rb87Sr86']),length.out=20)
y <- predict(fit,newdata=data.frame(Rb87Sr86=x))
lines(x,y)
```

To add a 95% confidence interval for the fit:

```
yci <- predict(fit,newdata=data.frame(Rb87Sr86=x),
               interval='confidence',level=0.95)
matlines(x,yci,lty=1,col='red')
```

where `yci` is a 3-column matrix and `matlines` plots each of these against the vector `x`. The resulting diagram shows the uncertainty of the fit *parameters*. To display the 95% prediction interval for the *data*:

```
ypi <- predict(fit,newdata=data.frame(Rb87Sr86=x),
               interval='prediction',level=0.95)
matlines(x,ypi,lty=1,col='blue')
```

Note how the prediction interval is wider than the confidence interval. This is akin to the standard deviation of the data being wider than the standard error of the mean (Equation 41).

7. Exercise:

The linear trend on the $^{87}$Sr/$^{86}$Sr vs. $^{87}$Rb/$^{86}$Sr plot is called an *isochron*. It describes the radioactive decay of $^{87}$Rb to $^{87}$Sr since time $t$:

$$\frac{^{87}Sr}{^{86}Sr} = \left(\frac{^{87}Sr}{^{86}Sr}\right)_{\circ} + \frac{^{87}Rb}{^{86}Sr}\left(e^{\lambda_{87}t} - 1\right) \tag{49}$$

where $\left({}^{87}Sr/{}^{86}Sr\right)_{\circ}$ is the initial (non-radiogenic) $^{87}$Sr/$^{86}$Sr-ratio and $\lambda_{87}$ is the $^{87}$Rb decay constant (= 0.014 Gyr$^{-1}$). Equation 49 defines a straight line with slope $\left(e^{\lambda_{87}t} - 1\right)$ and intercept $\left({}^{87}Sr/{}^{86}Sr\right)_{\circ}$. Can you use the isochron fit for the `RbSr` dataset to calculate the Rb-Sr age and uncertainty?

8. The `lm()` function can also be used for multivariate regression:

```
ThU <- read.csv(file='ThU.csv',header=TRUE)
ThUfit <- lm(Th230U238 ~ Th232U238 + U234U238, data=ThU)
```

Querying the coefficients:

```
> coef(ThUfit)
(Intercept)   Th232U238    U234U238
 0.72961274  0.19156147  0.01440391
```

indicates that

$$\frac{^{230}Th}{^{238}U} = 0.7296 + 0.1916\frac{^{232}Th}{^{238}U} + 0.0144\frac{^{234}U}{^{238}U}$$

which describes a 3-dimensional isochron for U-series disequilibrium geochronology.

## 7.2 Spurious correlation

Karl Pearson is the father of modern statistics, founder of the world's first university statistics department (at UCL!) and a person with highly questionable political views. Equations 43 and 42 are named after him. In 1897, Pearson pointed out that applying linear regression analysis to ratios may lead to *spurious correlation* (Figure 9). Verify this for yourself:

```
X <- runif(n=100,min=10,max=11)
Y <- runif(n=100,min=10,max=11)
Z <- runif(n=100,min=10,max=18)
par(mfrow=c(2,2)) # set up 2x2 panel plot
plot(X,Y)         # upper left panel
plot(X,Z)         # upper right panel
plot(Y,Z)         # lower left panel
plot(X/Z,Y/Z)     # lower right panel
```

So even though `X`, `Y`, `Z` are uncorrelated, their ratios are not because `Z` appears in the denominator of both plot axes (Figure 9). Pearson showed that it is possible to predict the expected correlation coefficient (or 'null correlation') associated with the spurious effect. For example:

$$r\left(\frac{X}{Z},\frac{Y}{Z}\right) = \frac{r(X,Y)s(Y)^2s(Z)^2 - r(X,Y)s(X)^2s(Y)^2 - s(Z)^4 + r(X,Z)s(X)^2s(Z)^2}{\sqrt{s(Y)^4 + s(Z)^4 - r(Y,Z)s(Y)^2s(Z)^2}\sqrt{s(X)^4 + s(Z)^4 - r(X,Z)s(X)^2s(Z)^2}} \tag{50}$$

It is important to be aware of this phenomenon because ratio scatter plots are commonplace in geology. Examples are the $^{40}$K/$^{36}$Ar-$^{40}$Ar/$^{36}$Ar isochron of geochronology and the Zr-Zr/Y tectonic discrimination diagram.

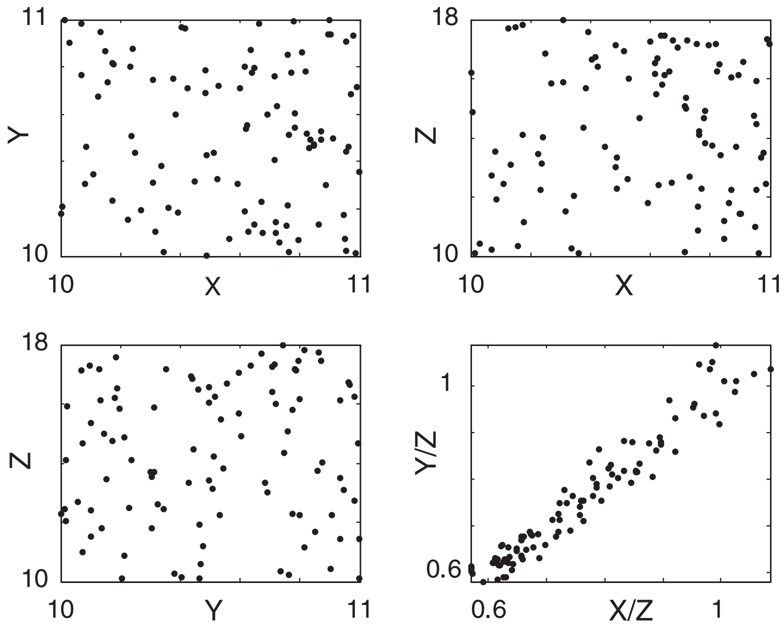

Figure 9: X, Y and Z are uncorrelated, uniform random numbers. The strong spurious correlation of the ratios Y/Z and X/Z is an artifact of the relatively large variance of Z relative to X, Y and Z.

## 8 Unsupervised learning

The simple plots of Section 3 are useful for visualising simple datasets of one or two dimensions. However many Earth Science datasets span multiple dimensions. For example, a geochemist may have analysed the concentration of 20 elements in 50 samples; or a palaeoecologist may have counted the relative abundances of 15 species at 50 sites. This Section will introduce some tools that can help us see some structure in such 'big' datasets without any prior knowledge of clusters, groupings or trends in the dataset. This is called unsupervised learning, as opposed to the supervised learning algorithms of Section 9.

### 8.1 Principal Component Analysis

Principal Component Analysis is an exploratory data analysis method that takes a high dimensional dataset as input and produces a lower (typically two-) dimensional 'projection' as output. PCA is closely related to Multidimensional Scaling (MDS), which is introduced in Section 8.2. This tutorial introduces PCA using the simplest working example of three two-dimensional points.

1. Consider the following bivariate ($a$ and $b$) dataset of three (1, 2 and 3) samples:

$$X = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cc} a & b \end{array} \\ \begin{bmatrix} -1 & 7 \\ 3 & 2 \\ 4 & 3 \end{bmatrix} \end{array} \tag{51}$$

   Generating and plotting $X$ in `R`:

```
X <- matrix(c(-1,3,4,7,2,3),nrow=3,ncol=2)
colnames(X) <- c('a','b')
plot(X)
```

   yields a diagram in which two of the three data points plot close together while the third one plots further away.

2. Imagine that you live in a one-dimensional world and cannot see the spatial distribution of the three points represented by $X$. Principal Component Analysis (PCA) is a statistical technique (invented by Karl Pearson) to represent multi- (e.g., two-) dimensional data in a lower- (e.g., one-) dimensional space whilst preserving the maximum amount of information (i.e., variance). This can be achieved by decomposing $X$ into four matrices ($C$, $S$, $V$ and $D$):

$$\begin{aligned} X &= 1_{3,1}\ C + S\ V\ D \\ &= \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 2 & 4 \end{bmatrix} + \begin{bmatrix} -1.15 & 0 \\ 0.58 & -1 \\ 0.58 & 1 \end{bmatrix} \begin{bmatrix} 3.67 & 0 \\ 0 & 0.71 \end{bmatrix} \begin{bmatrix} 0.71 & -0.71 \\ 0.71 & 0.71 \end{bmatrix} \end{aligned} \tag{52}$$

where $C$ is the centre (arithmetic mean) of the two data columns; $S$ are the *normalised scores*; the diagonals of $V$ correspond to the standard deviations of the two principal components; and $D$ is a rotation matrix (the *principal directions*). $S$, $V$ and $D$ can be recombined to define two more matrices:

$$P = S\ V = \begin{bmatrix} -4.24 & 0 \\ 2.12 & -0.71 \\ 2.12 & 0.71 \end{bmatrix}, \tag{53}$$

$$\text{and } L = V\ D = \begin{bmatrix} 2.6 & -2.6 \\ 0.5 & 0.5 \end{bmatrix} \tag{54}$$

where $P$ is a matrix of transformed coordinates (the *principal components* or *scores*) and $L$ are the scaled eigenvectors or *loadings*. Figure 10.i shows $X$ as numbers on a scatterplot, $C$ as a yellow square, and $1_{2,1}C \pm L$ as a cross. Thus, the first principal direction (running from the upper left to the lower right) has been stretched by a factor of $(3.67/0.71) = 5.2$ w.r.t the second principal component, which runs perpendicular to it. Figure 10 can be reproduced with the following `R`-code:

```
source('helper.R')
PCA2D(X)
```

3. Although the two-dimensional example is useful for illustrative purposes, the true value of PCA obviously lies in higher dimensional situations. As a second example, let us consider one of `R`'s built-in datasets. `USArrests` contains statistics (in arrests per 100,000 residents) for assault, murder, and rape in each of the 50 US states in 1973. Also given is the percentage of the population living in urban areas. Thus, `USArrests` is a four-column table that cannot readily be visualised on a two-dimensional surface. Applying PCA yields four principal components, the first two of which represent 62% and 25% of the total variance, respectively. Because the four columns of the input data are expressed in different units (arrests per 100,000 or percentage), it is necessary to scale the data to have unit variance before the analysis takes place:

```
pc <- prcomp(USArrests, scale=TRUE)
biplot(pc)
```

You will see that the loading vectors for `Murder`, `Assault` and `Rape` are all pointing in approximately the same direction (dominating the first principal component), perpendicular to `UrbanPop` (which dominates the second principal component). This tells us that crime and degree of urbanisation are not correlated in the United States.

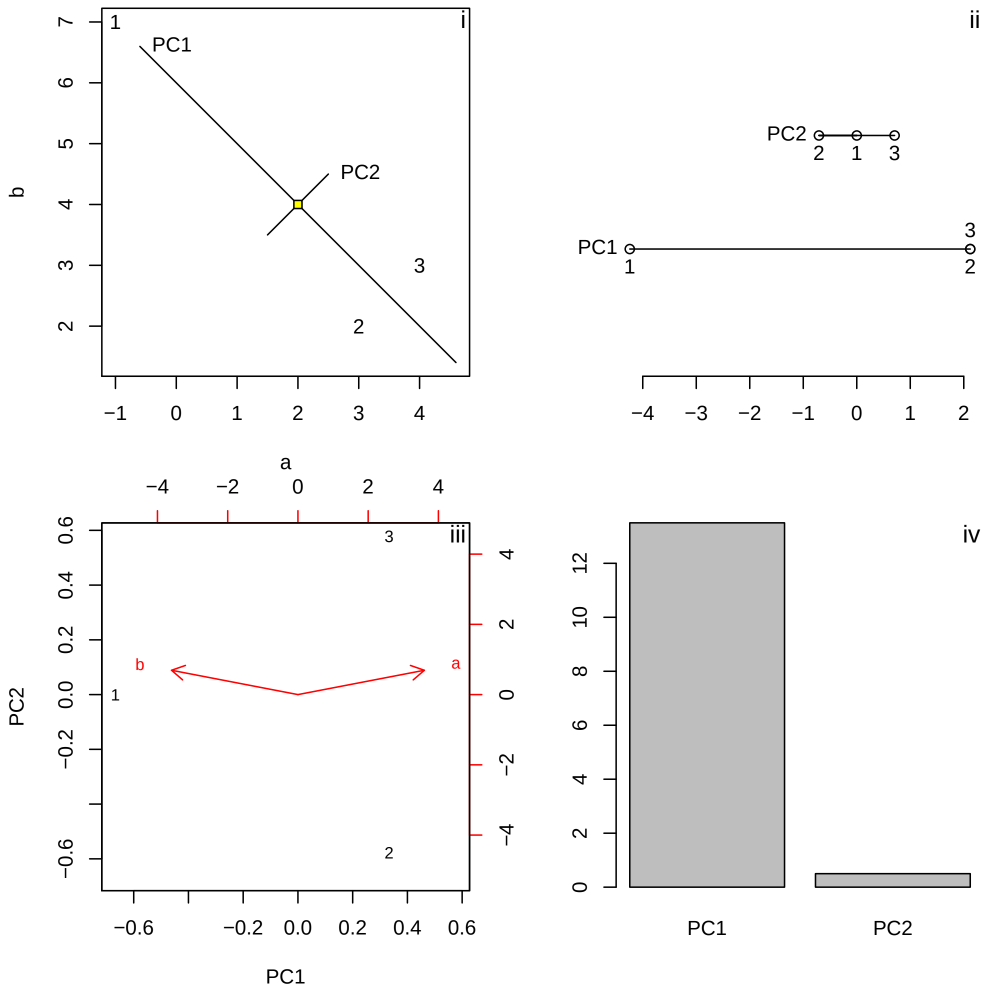

Figure 10: i – Three samples (1, 2 and 3) of bivariate ($a$ and $b$) data ($X$ in Equation 51). The yellow square marks the arithmetic mean ($C$ in Equation 52), the cross marks the two principal directions ($D$ in Equation 52) stretched by the diagonal elements (i.e. the standard deviations) of $V$ (Equation 52); ii – The projection of the data points on these two directions yields two principal components ($P$ in Equation 53), representing a one dimensional representation of the two-dimensional data; iii – A biplot of both principal components along with the loadings of the two variables shown as arrows; iv – The squared diagonal values of $V$ (Equation 52) indicate the relative amounts of variance encoded by the two principal components.

## 8.2 Multidimensional Scaling

Multidimensional Scaling (MDS) is a less restrictive superset of PCA. This tutorial uses a geographical example to demonstrate how MDS re-creates a map of Europe from a table of pairwise distances between European cities. Applying the same algorithm to the synthetic toy-example of Section 8.1 yields exactly the same output as PCA.

1. MDS is a dimension-reducing technique that aims to extract two (or higher) dimensional ‘maps’ from tables of pairwise distances between objects. This method is most easily illustrated with a geographical example. Consider, for example, the `eurodist` dataset that is built into `R`, and which gives the road distances (in km) between 21 cities in Europe (see `?eurodist` for further details):

```
> eurodist
```

2. To MDS configuration can be obtained by R's built-in cmdscale function

   ```
   conf <- cmdscale(eurodist)
   ```

   Set up an empty plot with a 1:1 aspect ratio, and then label the MDS configuration with the city names:

   ```
   plot(conf,type='n',asp=1)
   text(conf,labels=labels(eurodist))
   ```

   Note that the map may be turned 'upside down'. This reflects the rotation invariance of MDS configurations.

3. R's cmdscale function implements so-called 'classical' MDS, which aims to fit the actual distances. If these distances are Euclidean, then it can be shown that MDS is equivalent to PCA. To demonstrate this equivalence, let us apply MDS to the data in Equation 51. First run the first two lines of code from Section 8.1 if you haven't done so already. Calculating the Euclidean distances between the three samples produces a dissimilarity matrix $d$. For example, the distance between points 1 and 2 is $\sqrt{(-1-3)^2+(7-2)^2}=6.4$. This value is stored in d[1,1]. In R:

   ```
   d <- dist(X)
   ```

   which produces:

$$d = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 6.4 & 6.4 \\ 2 & 6.4 & 0 & 1.4 \\ 3 & 6.4 & 1.4 & 0 \end{array} \tag{55}$$

4. Next, calculate the MDS configuration:

   ```
   conf2 <- cmdscale(d)
   ```

   Finally, plot the MDS configuration as a scatterplot of text labels:

   ```
   plot(conf2,type='n')
   text(conf2,labels=1:3)
   ```

   Which is identical to the PCA configuration of Figure 10.iii apart from an arbitrary rotation or reflection.

5. An alternative implementation of MDS loosens the Euclidean distance assumption by fitting the *relative* distances between objects. Let us apply this to the dataset of European city distances using the isoMDS function of the 'Modern Applied Statistics with S' (MASS) package:

   ```
   library(MASS)
   ```

   To compute and plot the non-metric MDS configuration:

   ```
   conf3 <- isoMDS(eurodist)$points
   plot(conf3,type='n',asp=1)
   text(conf3,labels=labels(eurodist))
   ```

where conf3 is a list with two items: stress, which expresses the goodness-of-fit of the MDS configuration; and points, which contains the configuration. The '$' operator is used to access any of these items. Non-metric MDS is a less-restrictive superset of classical MDS and, hence, PCA, which opens this methodology up to non-Euclidean dissimilarity measures.

## 8.3 K-means clustering

K-means cluster is an unsupervised learning algorithm that tries to group data based on their similarity. As the name suggests, the method requires that we pre-specify the number of clusters ($k$) to be found in the dataset. The algorithm randomly assigns each observation to a cluster, and finds the centroid of each of these. Then, it iterates through two steps until the within cluster variation cannot be reduced any further:

i. Reassign data points to the cluster whose centroid is closest.

ii. Calculate new centroid of each cluster.

The within cluster variation is calculated as the sum of the Euclidean distances between the data points and their respective cluster centroids.

1. Let us illustrate the k-means method using R.F. Fisher's classic dataset of iris flower measurements, which is built into R:

```
> iris
```

   returns a $50 \times 5$ data table with the sepal and petal length and width measurements (in cm) for 50 specimens of 3 iris species. The fifth column contains the names of the species but these are not used by the k-means algorithm (because it is unsupervised).

2. Let us have a look at a two-dimensional projection of the data (Petal.Length vs. Petal.Width):

```
species <- as.numeric(iris[,'Species']) # turn the names into numbers
plotsym <- c(16,17,18)                  # define plot symbols
plotcol <- c("red","green","blue")      # define plot colours
plot(x=iris[,'Petal.Length'],y=iris[,'Petal.Width'],
     xlab='Petal length (cm)',ylab='Petal Width (cm)',
     pch=plotsym[species],col=plotcol[species])
legend('topleft',legend=levels(iris[,'Species']),
       pch=plotsym,col=plotcol)
```

   You can see that the three species of flowers clearly define three distinct groups.

3. Let us see if the k-means clustering algorithm can correctly classify them without looking at the labels:

```
means <- kmeans(iris[,-5],centers=3,nstart=20)
```

   where iris[,-5] removes the fifth column of the iris dataset. This is equivalent to typing iris[,1:4] (try it!). centers specifies that we are looking for three clusters, and nstart tells kmeans() to pick 20 random starting positions. This is necessary to avoid getting trapped in a 'local minimum'. Querying means at the command prompt returns a 9-element list of outputs, including an item called cluster. This returns the group assigned to each of the 150 flowers in the dataset. These can be tabulated along with the species:

```
> table(means$cluster,iris[,'Species'])
    setosa versicolor virginica
  1      0         48         4
  2     50          0         0
  3      0          2        46
```

This table shows that only 6/150 flowers were incorrectly classified (i.e., they have fallen into a different cluster than the rest of their species).

## 8.4 Hierarchical clustering

The k-means clustering algorithm of Section 8.3 requires that we pre-specify the number of groups. This is not always possible. Hierarchical clustering is an alternative approach that builds a hierarchy from the bottom-up, and doesn't require us to specify the number of groups beforehand. The algorithm works as follows:

i. Put each data point in its own cluster.

ii. Identify the closest two clusters and join them together.

iii. Repeat the above step till all the data points are in a single cluster.

1. First we need to define a way to measure the distance between the objects in our dataset. R's `dist()` function

   ```
   d <- dist(iris[,-5])
   ```

   uses the Euclidean distance by default (see Section 8.2.3). Other distances are possible as well (see `?dist` for some suggestions).

2. Apply the hierarchical clustering algorithm:

   ```
   tree <- hclust(d)
   plot(tree)
   ```

   This produces a tree with 150 'leaves', each corresponding to a single flower. The 'height' of the tree corresponds to the maximum possible distance between points belonging to two different clusters. Different definitions are possible as well, as specified by the optional `method` argument of the `hclust()` function. The height changes rapidly between one and three clusters, indicating that these correspond to the most significant bifurcations. So let us cut down the tree at this level:

   ```
   treecut <- cutree(tree,k=3)
   ```

3. View the performance of the simplified tree:

   ```
   > table(treecut, iris[,'Species'])
   treecut setosa versicolor virginica
         1     50          0         0
         2      0         23        49
         3      0         27         1
   ```

   The algorithm has done a good job at classifying `setosa` and `virginica` but is struggling with `versicolor`.

4. Exercise:

Experiment with the optional arguments of the `hclust()` function to see if you can improve the classification. See `?hclust` for details.

## 9 Supervised learning

PCA, MDS and cluster analysis are exploratory data techniques that do not require prior knowledge about the data. In contrast, linear discriminant analysis (LDA, Section 9.1) and classification and regression trees (CART, Section 9.2) use *training data* to classify samples into pre-defined categories.

### 9.1 Discriminant Analysis

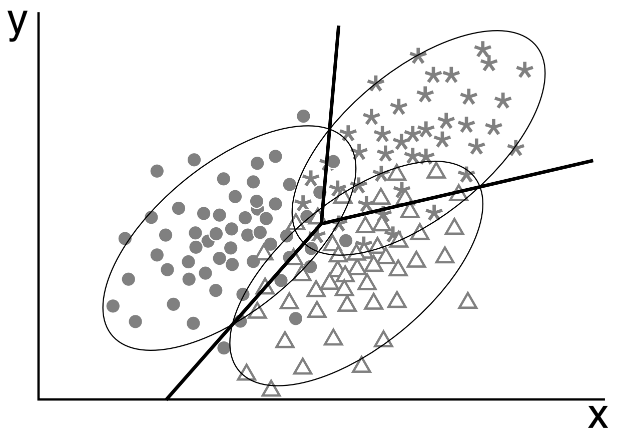

Figure 11: Discriminant analysis of three classes with equal covariance matrices leads to linear discriminant boundaries.

Consider a dataset of a large number of $N$-dimensional data $X$, which belong to one of $K$ classes. For example, $X$ might be a set of geochemical data (e.g., $SiO_2$, $Al_2O_3$, etc) from basaltic rocks of $K$ tectonic affinities (e.g., mid ocean ridge, ocean island, island arc, ...). We might ask ourselves which of these classes an unknown sample $x$ belongs to. This question is answered by *Bayes' Rule*: the decision $d$ is the class $G$ ($1 \leq G \leq K$) that has the highest *posterior probability* given the data $x$:

$$d = \max_{k=1,\ldots,K} Pr(G = k|X = x) \tag{56}$$

This posterior distribution can be calculated according to Bayes' Theorem:

$$Pr(G|X) = Pr(X|G)Pr(G) \tag{57}$$

where $Pr(X|G)$ is the *likelihood* of the data in a given class, and $Pr(G)$ the *prior probability* of the class, which we will consider uniformly distributed; i.e., $Pr(G = 1) = Pr(G = 2) = \ldots = Pr(G = K) = 1/K$. Therefore, plugging Equation 57 into Equation 56 reduces Bayes' Rule to a comparison of likelihoods. We now make the simplifying assumption of multivariate normality:

$$Pr(X = x|G = k) = \frac{\exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right)}{\sqrt{(2\pi)^N|\Sigma_k|}} \tag{58}$$

Where $\mu_k$ and $\Sigma_k$ are the mean and covariance of the $k^{\text{th}}$ class and $(x - \mu_k)^T$ indicates the transpose of the matrix $(x - \mu_k)$. Using Equation 58 and taking logarithms, Equation 56 becomes:

$$d = \max_{k=1,\ldots,K}\left[-\frac{1}{2}log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right] \tag{59}$$

Equation 59 is the basis for *quadratic discriminant analysis* (QDA). Usually, $\mu_k$ and $\Sigma_k$ are not known, and must be estimated from the training data. If we make the additional assumption that all the classes share the same covariance structure (i.e., $\Sigma_k = \Sigma$ for all $k$), then Equation 56 simplifies to:

$$d = \max_{k=1,\ldots,K} \left[ x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k \right] \tag{60}$$

This is the basis of *linear discriminant analysis* (LDA), which has some desirable properties. For example, because Equation 60 is linear in $x$, the decision boundaries between the different classes are straight lines (Figure 11). Furthermore, LDA can lead to a significant reduction in dimensionality, in a similar way to *principal component analysis* (PCA). PCA finds an orthogonal transformation $B$ (i.e., a rotation) that transforms the centered data ($X$) to orthogonality, so that the elements of the vector $BX$ are uncorrelated. $B$ can be calculated by an eigenvalue decomposition of the covariance matrix $\Sigma$. The eigenvectors are orthogonal linear combinations of the original variables, and the eigenvalues give their variances. The first few principal components generally account for most of the variability of the data, consituting a significant reduction of dimensionality (Figure 12).

Like PCA, LDA also finds linear combinations of the original variables. However, this time, we do not want to maximise the overall variability, but find the orthogonal transformation $Z = BX$ that maximizes the *between class* variance $S_b$ relative to the *within class* variance $S_w$, where $S_b$ is the variance of the class means of $Z$, and $S_w$ is the pooled variance about the means (Figure 12).

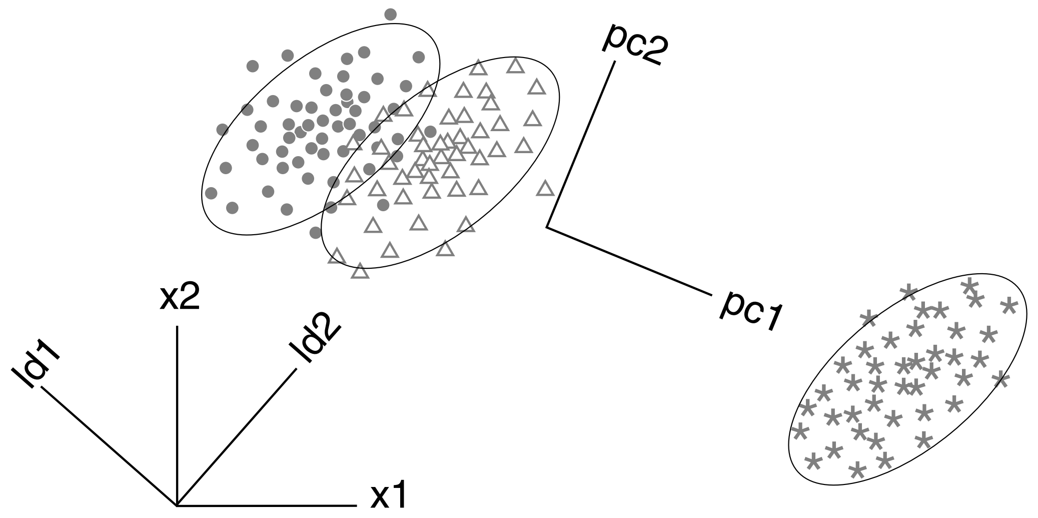

Figure 12: Similarities and differences between linear discriminant and principal component analysis. x1 and x2 are the original variables, pc1 and pc2 are the principal components and ld1 and ld2 are the linear discriminant functions.

1. Let us apply linear discriminant analysis to Fisher's iris dataset:

```
library(MASS)
r <- lda(Species ~ . ,data=iris)
```

   where "`.`" is equivalent to "`Sepal.Width + Sepal.Length + Petal.Width + Petal.Length`".

2. The discriminants are linear functions of the original variables that maximise the between-group variance. Display the first of these against each other on a scatterplot, with the different flowers marked in different colours:

```
plot(r,col=as.numeric(iris[,'Species']))
```

   The three groups are pulled apart more clearly than using any binary combination of the raw variables (e.g., Section 8.3.2).

3. Let us now guess the species of a new flower with 5 cm sepal length, 5 cm sepal width, 3 cm petal length and 3 cm petal width:

```
newflower <- data.frame(Sepal.Length=5,Sepal.Width=5,
                        Petal.Length=3,Petal.Width=3)
pred.lda <- predict(object=r,newdata=newflower)
```

   The posterior probabilities are given by:

```
> pred.lda$posterior
       setosa versicolor virginica
1 8.45939e-14  0.1316704 0.8683296
```

   which means that there is 13% probability that the new flower is Versicolor and 87% chance that it is Virginica. The flower is therefore classified as Virginica:

```
> pred.lda$class
[1] virginica
```

4. Let us assess the misclassification rate of the iris data:

```
> pred.iris <- predict(r,data=iris)
> table(pred.iris$class,iris[,'Species'])
             setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         48         1
  virginica       0          2        49
```

   So only 3/150 of the flowers were classified incorrectly. This looks like a great result. But unfortunately it is overly optimistic.

5. The problem is that we are using the training data to test the performance of our discriminator. It is much better to use an *independent* dataset to test the predictive power of our method. The resulting misclassification rate is usually more pessimistic but more realistic than the one obtained under exercise 4.

```
# select 75 random numbers between 1 and 150:
train <- sample(1:150,75)
r2 <- lda(Species ~ .,data=iris,subset=train)
pred.test <- predict(object=r,newdata=iris[-train,])
table(pred.test$class,iris[-train,'Species'])
```

## 9.2 Classification and Regression Trees

Suppose that we have $N$ $J$-dimensional data points $X^n = \{x_1^n, \ldots, x_j^n, \ldots, x_J^n\}$ (for $1 \leq n \leq N$) belonging to one of $K$ classes: $Y^n = c_1|\ldots|c_k|\ldots|c_K$. For example, $\underline{X} = \{X^1, \ldots, X^n, \ldots, X^N\}$ might represent $J$ features (e.g., various major and trace element concentrations, isotopic ratios, color, weight, ...) measured in $N$ samples of basalt. $c_1, \ldots, c_K$ might then be tectonic affinity, e.g., "mid-ocean ridge", "ocean island", "island arc", ... . The basic idea behind classification and regression trees (CART) is to approximate the parameter space by a piecewise constant function, in other words, to partition $\underline{X}$ into $M$ disjoint regions $\{R_1, \ldots, R_m, \ldots, R_M\}$. An example of such a partition

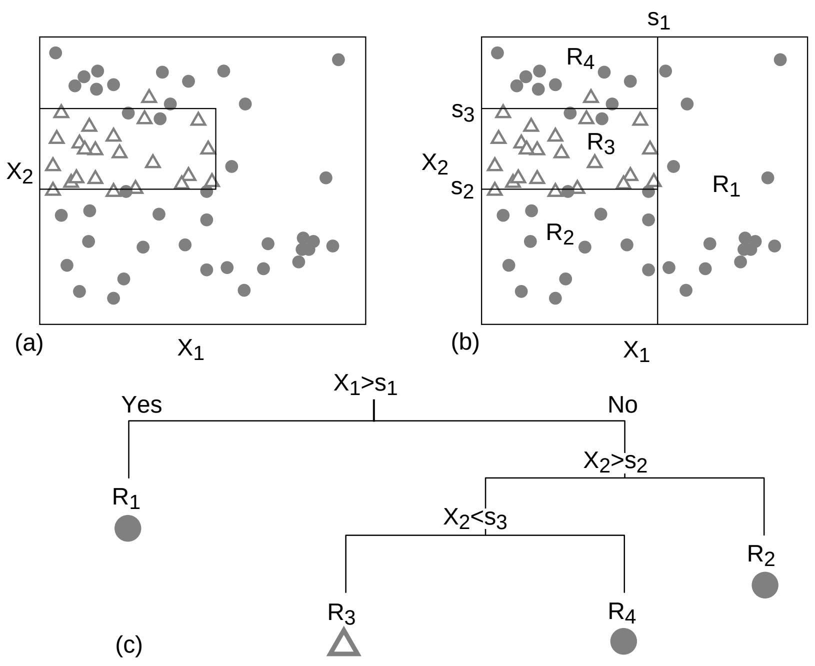

Figure 13: An example of a bivariate $(X_1, X_2)$ classification tree. Classification trees approximate the data space with a piecewise constant function (a). To ensure computational feasibility, a recursive binary partitioning method is used (b). Such partitions have the added advantage that the results can be visualized as a two-dimensional graph or 'tree' (c). In this and subsequent trees, left branches mean 'Yes' and right branches 'No'.

is given in Figure 13.a.

Because it is impossible to describe all possible partitions of the feature space, we restrict ourselves to a small subset of possible solutions, the recursive binary partitions (Figure 13.b). Surprising as it may seem, considering the crudeness of this approximation, trees are one of the most powerful and popular data mining techniques in existence. One of the reasons for this is that besides assuring computational feasibility, the recursive partitioning technique described next also allows the representation of the multi-dimensional decision-space as a two dimensional tree graph (Figure 13.c).

At any point in the recursive process, a partition is defined by two quantities: the split variable $j$ $(1 \leq j \leq J)$ and the split point $s$ $(-\infty < s < \infty)$. For any given partition $R_m$ of a tree $T$, there are at most $N \times J$ possible binary subpartitions, which can be exhaustively searched. We choose the one that minimises the 'node impurity' $Q_m(T)$. Let $\hat{p}_{mk}$ be the proportion of class $k$ observations in node $m$, then

$$Q_m(T) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{61}$$

This particular form of $Q_m(T)$ is called the 'Gini index of diversity', but alternatives exist. Note that if we simply used the 'misclassification error' (i.e., the number of misclassified data points resulting from a candidate split) as a measure of node impurity, it would be impossible to partition the data set shown in Figure 13 because any initial split would misclassify all the triangles, yielding the same misclassification error (=23/63 in this case). The recursive partitioning process continues until all the end-nodes are 'pure', i.e. all belong to the same class.

The maximum sized tree thus obtained perfectly describes the training data. In other words, it has zero bias. However, for the purpose of prediction, this tree is not optimal, because it overfits the training data, causing high variance. The tree with optimal *predictive* power will be smaller than the largest possible tree, and can be found by 'pruning' the tree. Define the "cost-complexity criterion" of a tree $T$ as

$$cp_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T| \tag{62}$$

with $|T|$ the number of terminal nodes in $T$, $N_m$ the number of observations in the $m^{th}$ terminal node, $Q_m(T)$ the 'node impurity' defined by Equation 61 and $\alpha$ a tuning parameter. For a given $\alpha \geq 0$, it is possible to find the subtree $T_\alpha \subset T_0$ that minimizes $cp_\alpha(T)$ over all possible subtrees of the largest possible tree $T_0$

$$T_\alpha = \min_{T \subset T_\circ} cp_\alpha(T) \tag{63}$$

Repeating this procedure for a range of values $0 \leq \alpha < \infty$ produces a finite nested sequence of trees $\{T_0, T_{\alpha_1}, ..., T_{\alpha_{max}}\}$. Except for $T_0$, these trees will no longer have only pure end-nodes. Impure end-nodes are assigned the class that dominates in them. We then choose the value $\alpha^*$ that minimizes an estimate of future prediction error, for example by '$V$-fold cross-validation'. The training data are randomly divided into $V$ (e.g., ten) fractions of equal size. We then grow $V$ overly large trees $T_0^{(v)}$, each time using all but the $v^{\text{th}}$ sample fraction. Each of these trees then goes through the pruning procedure described before, yielding $V$ nested sequences of trees $\{T_0^{(v)}, T_{\alpha_1}^{(v)}, \ldots T_{\alpha_{max}}^{(v)}\}$ $(1 \leq v \leq V)$. The trees $T_\alpha^{(v)}$ were constructed without ever seeing the cases in the $v^{\text{th}}$ fraction. Sending the $v^{\text{th}}$ fraction down $T_\alpha^{(v)}$ for each $v = 1, \ldots, V$ thus yields an independent estimate of the misclassification error.

A plot of these cross-validated (CV) prediction errors versus the number of nodes in each of the nested subtrees shows a minimum at some point. As discussed before, trees with fewer nodes tend to have large bias, while the increased CV-cost of overly large trees is caused by their inflated variance. There typically exist several trees with CV-costs close to the minimum. Therefore, a '1-SE rule' is used, i.e., choosing the smallest tree whose CV misclassification cost does not exceed the minimum CV-cost plus one standard error of the CV-cost for the minimum CV-cost tree.

1. A simple example using the `iris` dataset and default cost-complexity settings:

```
library(rpart)
tree <- rpart(Species ~ ., data=iris, method="class")
plot(tree)
text(tree, use.n=T) # use.n adds the misclassification rates
```

2. To predict the class of a new flower:

```
newflower <- data.frame(Sepal.Length=5,Sepal.Width=5,
                        Petal.Length=3,Petal.Width=3)
pred.tree <- predict(object=tree,newdata=newflower)
```

Querying the results:

```
> pred.tree
  setosa versicolor virginica
1      0 0.02173913 0.9782609
```

Confirms with 98% confidence that the new flower is a Virginica.

3. A more complex example using cost-complexity analysis and pruning:

```
library(rpart)
# set the cross-validation to 10 and the complexity parameter to zero
my.control <- rpart.control(xval=10, cp=0, minsplit=1)

# fit a decision tree to all the data
tree.unpruned <- rpart(Species ~ ., data=iris,
                       method="class",control=my.control)

# look at the sequence of unpruned trees to decide which
# value of cp gives the optimal tree:
printcp(tree.unpruned)
plotcp(tree.unpruned)

# get the optimal subtree
tree.pruned <- prune(tree.unpruned, cp=.01)

# plot this tree
plot(tree.pruned)
text(tree.pruned, use.n=T)
```

## 10 Compositional data

The normal distribution (Section 5.1) plays a key role in linear regression (Section 7), PCA (Section 8.1), Discriminant Analysis (Section 9.1) and many other standard statistical techniques. Although Gaussian distributions are common in nature, it is dangerous to assume normality for all datasets. In fact, more often than not, the normality assumption is invalid for geological data. Ignoring this non-normality can lead to counter-intuitive and plainly wrong results.

### 10.1 Ratio data

To illustrate the dangers of blindly assuming normality, let us consider the simple case of *ratio data*, which are quite common in the Earth Sciences. Take, for example, the ratio of apatite to tourmaline in heavy mineral analysis, which has been used to indicate the duration of transport and storage prior to deposition. In this part of the tutorial, we will investigate the statistics of ratio data using a synthetic example.

1. Create two vectors $A$ and $B$, each containing 100 random numbers between 0 and 1:

```
ns <- 100
A <- runif(ns)
B <- runif(ns)
```

Intuitively, given that $\underline{A/B = 1/(B/A)}$ and $B/A = 1/(A/B)$, we would expect the same to be true for their means $\overline{(A/B)}$ and $\overline{(B/A)}$. However, when we define two new variables for the (inverse) of the (reciprocal) mean ratios:

```
AB.mean <- mean(A/B)
inv.BA.mean <- 1/mean(B/A)
```

then we find that AB.mean≠inv.BA.mean. So $\overline{(A/B)} \neq 1/\overline{(B/A)}$ and $\overline{(B/A)} \neq 1/\overline{(A/B)}$! This is a counterintuitive and clearly wrong result.

2. Calculate the standard deviation of $A/B$ and multiply this by two to obtain a '2-sigma' confidence interval for the data:

   ```
   AB.sd <- sd(A/B)
   LL <- AB.mean - 2*AB.sd
   UL <- AB.mean + 2*AB.sd
   ```

   then we find that $LL < 0$, which is nonsensical since $A$ and $B$ are both strictly positive numbers and their ratio is therefore not allowed to take negative values either. Herein lies the root of the problem. The sampling distribution of A/B is positively skewed, whereas the normal distribution is symmetric with tails ranging from $-\infty$ to $+\infty$. Geologists frequently encounter strictly positive numbers. *Time*, for example, is a strictly positive quantity, expressed by geochronologists as 'years before present (BP)', where 'present' is equivalent to zero.

3. The problems caused by applying normal theory to strictly positive data can often be solved by simply taking logarithms. The transformed data are then free to take on any value, including negative values, and this often allows normal theory to be applied with no problems. For example, when we calculate the (geometric) mean after taking the logarithm of the ratio data:

   ```
   logAB <- log(A/B)
   logBA <- log(B/A)
   AB.gmean <- exp(mean(logAB))
   inv.BA.gmean <- 1/exp(mean(logBA))
   ```

   then we find that `AB.gmean` = `inv.BA.gmean`, which is a far more sensible result. Note the similarity between this example and the fuel consumption calculation of Section 4.2.

4. Calculating the 2-sigma interval for the log-transformed data:

   ```
   LL <- exp( mean(logAB) - 2*sd(logAB) )
   UL <- exp( mean(logAB) + 2*sd(logAB) )
   ```

   also produces strictly positive values, as expected.

## 10.2 The logratio transformation

Like the ratios of the previous section, the chemical compositions of rocks and minerals are also expressed as strictly positive numbers. They, however, do not span the entire range of positive values, but are restricted to a narrow subset of that space, ranging from 0 to 1 (if fractions are used) or from 0 to 100% (using percentage notation). Compositions are further restricted by a constant sum constraint:

$$\sum_{i=1}^{n} C_i = 1$$

for an $n$-component system. Consider, for example, a three-component system $\{x, y, z\}$, where $x+y+z = 1$. Such compositions can be plotted on ternary diagrams, which are very popular in geology. Well-known examples are the Q-F-L diagram of sedimentary petrography, the A-CN-K diagram in weathering studies, and the A-F-M, Q-A-P and Q-P-F diagrams of igneous petrology. The very fact that it is possible to plot a ternary diagram on a two-dimensional sheet of paper already tells us that it really displays only two and not three dimensions worth of information. Treating the ternary data space as a regular Euclidean space with Gaussian statistics leads to wrong results, as illustrated by the following example.

1. Read a compositional dataset containing the major element composition of a number of synthetic samples:

```
ACNK <- read.csv('ACNK.csv',header=TRUE,
                 row.names=1,check.names=FALSE)
```

   where the optional `check.names` arguments tells `R` not to substitute the special characters (such as '+') in the header row.

2. Calculate the arithmetic mean composition and 95% confidence limits for each column of the dataset:

```
mu <- colMeans(ACNK)
sig <- apply(ACNK,MARGIN=2,FUN='sd')
```

   and construct the 2-sigma confidence confidence bounds:

```
LL <- mu - 2*sig
UL <- mu + 2*sig
```

3. In order to plot the compositional data on a ternary diagram, we will need to first load the `provenance` package into memory:

```
library(provenance)
```

   Now plot the $Al_2O_3$, ($CaO + Na_2O$) and $K_2O$ compositions on a ternary diagram alongside the arithmetic mean composition:

```
plot(ternary(ACNK),pch=20,labels=NA)
points(ternary(mu),pch=22,bg='blue')
```

   where `ternary(x)` creates a ternary data 'object' from a variable `x`, and `pch=20` and `pch=22` produce filled circles and squares, respectively. Notice how the arithmetic mean plots outside the data cloud, and therefore fails to represent the compositional dataset (Figure 14).

4. To add a 2-sigma polygon to this figure, we will need to use one of the functions in the `helper.R` script. We then use the `ternary.polygon()` function in this file to add our desired 'normal' bounds to the ternary diagram:

```
source('helper.R')
ternary.polygon(LL,UL,col='blue')
```

   Note that the polygon partly plots outside the ternary diagram, into physically impossible negative data space. This nonsensical result is diagnostic of the dangers of applying 'normal' statistics to compositional data. It is similar to the negative limits for the ratio data in Section 10.1.

A comprehensive solution to the compositional data conundrum was only found in the 1980s, by Scottish statistician John Aitchison. It is closely related to the solution of the ratio averaging problem discussed in the previous section. The trick is to map the n-dimensional composition to an (n-1)-dimensional Euclidean space by means of a logratio transformation. For example, in the ternary case, we can map the compositional variables $x$, $y$ and $z$ to two transformed variables $v$ and $w$:

$$v = \ln\left(\frac{x}{z}\right), \; w = \ln\left(\frac{y}{z}\right) \tag{64}$$

After performing the statistical analysis of interest (e.g., calculating the mean or constructing a 95% confidence region) on the transformed data, the results can then be mapped back to compositional space with the inverse logratio transformation. For the ternary case:

$$x = \frac{e^v}{e^v + e^w + 1}, \; y = \frac{e^w}{e^v + e^w + 1}, \; z = \frac{1}{e^v + e^w + 1} \tag{65}$$

This transformation is implemented in the `provenance` package. Let us use this feature to revisit the K-CN-A dataset, and add the geometric mean and 95% confidence region to the ternary diagram for comparison with the arithmetic mean and confidence polygon obtained before.

5. Compute the geometric mean composition and add it to the existing ternary diagram as a red square:

   ```
   mug <- exp(colMeans(log(ACNK)))
   points(ternary(mug),pch=22,bg='red')
   ```

   This red square falls right inside the data cloud, an altogether more satisfying result than the arithmetic mean shown in blue (Figure 14).

6. To add a compositional confidence contour, we must re-read `ACNK.csv` into memory using the `read.compositional()` function. This will tell the `provenance` package to treat the resulting variable as compositional data in subsequent operations:

   ```
   ACNK2 <- read.compositional('ACNK.csv',check.names=FALSE)
   ```

   Adding the 95% confidence contour using `provenance`'s `ternary.ellipse` function:

   ```
   ternary.ellipse(ACNK2,alpha=0.05)
   ```

   creates a 95% confidence ellipse in logratio space, and maps this back to the ternary diagram. This results in a 'boomerang'-shaped contour that tightly hugs the compositional data whilst staying inside the boundaries of the ternary diagram (Figure 14).

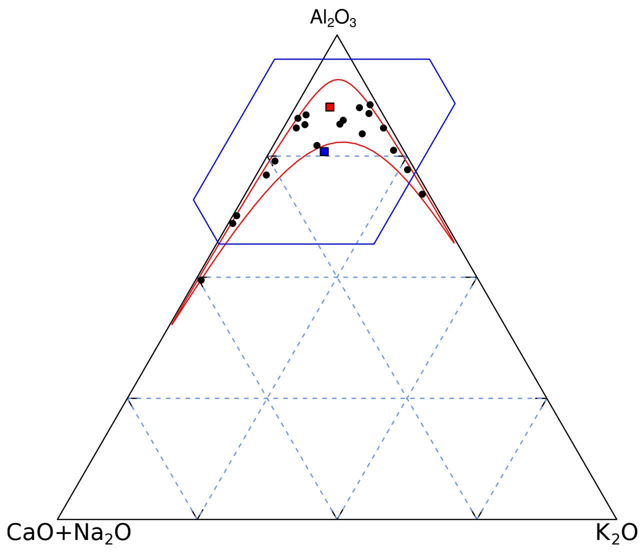

Figure 14: Graphical output of Section 10. Black circles mark 20 synthetic $Al_2O_3$, $(CaO + Na_2O)$ and $K_2O$ compositions, drawn from a logistic normal distribution. The blue square and polygon mark the arithmetic mean and 2-$\sigma$ confidence polygon. The former falls outside the data cloud, whereas the latter plots outside the ternary diagram, in physically impossible negative space. The red square and confidence envelope represent the geometric mean and 95% confidence region calculated using Aitchison's logratio approach.

## 10.3 PCA of compositional data

Consider the following trivariate ($a$, $b$ and $c$) dataset of three (1, 2 and 3) compositions:

$$X = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \left[\begin{array}{ccc} 0.03 & 99.88 & 0.09 \\ 70.54 & 25.95 & 3.51 \\ 72.14 & 26.54 & 1.32 \end{array}\right] \end{array} \quad (66)$$

1. Plot the data on a ternary diagram:

```
X <- matrix(c(0.03,99.88,0.09,70.54,25.95,3.51,
              72.14,26.54,1.32),3,3,byrow=TRUE)
colnames(X) <- c('a','b','c')
library(provenance)
plot(ternary(X))
```

2. It would be wrong to apply conventional PCA to this dataset, because this would ignore the constant sum constraint. As was discussed in Section 8.1, PCA begins by 'centering' the data via the arithmetic mean. Section 10 showed that this yields incorrect results for compositional data. Subjecting the data to a logratio transformation produces:

$$X_a = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cc} \ln(a/c) & \ln(b/c) \end{array} \\ \left[\begin{array}{cc} -1 & 7 \\ 3 & 2 \\ 4 & 3 \end{array}\right] \end{array} \quad (67)$$

which, the observant reader will note, is identical to the example of Equation 51. Applying conventional PCA to the log-transformed data of Equation 67 will yield two principal components that are expressed in terms of the logratios $\ln(a/c)$ and $\ln(b/c)$.

```
Xa <- cbind(log(X[,'a']/X[,'c']),log(X[,'b']/X[,'c']))
colnames(Xa) <- c('ln(a/c)','ln(b/c)')
pc <- prcomp(Xa)
biplot(pc)
```

3. Alternatively, the data of Equation 66 can also be subjected to a different type of logratio transformation. The so-called *centred* logratio transformation (as opposed to the *additive* logratio transformation of Equation 64) maps any set of compositional data vectors $x = \{x_1, \ldots, x_i, \ldots, x_n\}$, $y = \{y_1, \ldots, y_i, \ldots, y_n\}$, ... to the same number of (centred) logratios $u = \{u_1, \ldots, u_i, \ldots, u_n\}$, $v = \{v_1, \ldots, v_i, \ldots, v_n\}$, ..., where:

$$u_i = \ln(x_i) - [\ln(x_i) + \ln(y_i)]/2 \text{ and } v_i = \ln(y_i) - [\ln(x_i) + \ln(y_i)]/2 \quad (68)$$

Applying this transformation to the data of Equation 66 yields a new trivariate dataset:

$$X_c = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} \ln(a/g) & \ln(b/g) & \ln(c/g) \end{array} \\ \left[\begin{array}{ccc} -3 & 5 & -2 \\ 1.33 & 0.33 & -1.67 \\ 1.67 & 0.67 & -2.33 \end{array}\right] \end{array} \quad (69)$$

where $g$ stands for the geometric mean of each row. Subjecting Equation 69 to the same matrix decomposition as Equation 52 yields:

$$X_c = 1_{3,1}\, C + S\ V\ D = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 2 & -2 \end{bmatrix} + \begin{bmatrix} -1.15 & 0 & 0.82 \\ 0.58 & -1 & 0.82 \\ 0.58 & 1 & 0.82 \end{bmatrix} \begin{bmatrix} 3.67 & 0 & 0 \\ 0 & 0.41 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.71 & -0.71 & 0 \\ 0.41 & 0.41 & -0.82 \\ 0.58 & 0.58 & 0.58 \end{bmatrix} \tag{70}$$

Note that, even though this yields three principal components instead two, the standard deviation of the third component is zero. Therefore, all the information is contained in the first two components. The PCA map using the centred logratio transformation looks identical to that using the additive logratio transformation. The only difference is that the loadings are expressed in terms of the three centred logratio variables, rather than the two additive logratio variables. The former are easier to interpret than the latter, which is why the centred logratio transformation is preferred in this context.

```
g <- exp(rowMeans(log(X)))
Xc <- cbind(log(X[,'a']/g),log(X[,'b']/g),log(X[,'c']/g))
colnames(Xc) <- c('a','b','c')
pc <- prcomp(Xc)
biplot(pc)
```

4. The following script applies compositional PCA to a dataset of major element compositions from Namibia using base `R`:

```
# load the major element composition of Namib sand:
Major <- read.csv(file="Major.csv",
                  header=TRUE,row.names=1)
# apply the centred logratio transformation:
cMajor <- log(Major) -
       rowMeans(log(Major)) %*% matrix(1,1,ncol(Major))
# perform PCA of the logratio transformed data:
pc <- prcomp(cMajor)
# plot the results of the PCA analysis:
biplot(pc)
```

5. Alternatively, we can also do this more easily in `provenance`:

```
library(provenance)
# tell R that Major.csv contains compositional data:
Major.comp <- read.compositional('Major.csv')
# perform the principal component analysis:
pc.comp <- PCA(Major.comp)
# create the biplot:
plot(pc.comp)
```

where the `read.compositional` function reads the `.csv` file into an object of class `compositional`, thus ensuring that logratio statistics are used in all `provenance` functions (such as `PCA`) that accept compositional data as input. Also note that the `provenance` package *overloads* the `plot` function to generate a compositional biplot when applied to the output of the `PCA` function.

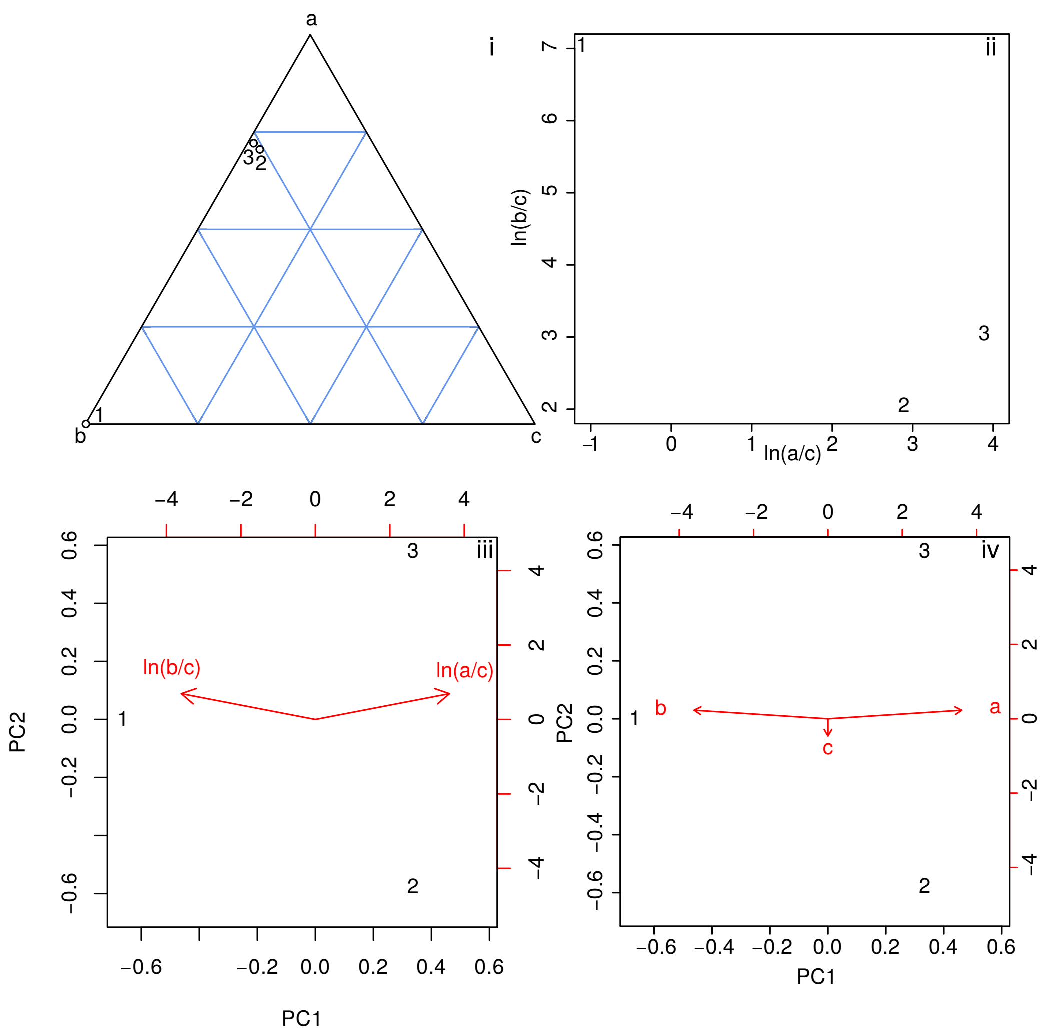

Figure 15: i – the compositional dataset of Equation 69 shown on a ternary diagram; ii – subjecting the same dataset to an additive logratio transformation (alr) produces a configuration of points that is identical to Figure 10.i; iii — as a consequence the PCA biplot of the logratio transformed data looks identical to Figure 10.iii; iv – using a centred logratio transformation (clr) yields the same configuration as panel iii but with more easily interpretable vector loadings.

Other statistical operations, such as (k-means and hierarchical) clustering, discriminant analysis, and classification and regression trees, can also be applied to compositional data after logratio transformation.

## 11 Directional data

Strike and dip, azimuth and elevation, latitude and longitude, ... *directional data* are ubiquitous in the Earth Sciences. And just like the compositional data discussed in the previous section of these notes, the statistical analysis of directional data is fraught with dangers.

1. The `helper.R` script contains a dataset with the orientation (in degrees) of thirty pebbles measured on a fluvioglacial terrace. Using the axiliary function `plot.circ()` function in `helper.R`, we can plot these data on a circle:

```
source('helper.R')
plot.circ(pebbles)
```

The pebble orientations are roughly centered around zero but exhibit significant scatter, from the northwest (276°) to the northeast (79°).

2. The arithmetic mean value of these angles is 189.2°, a nonsensical value:

```
points.circ(mean(pebbles),pch=19)
```

The problem is that directional data are 'wrapped around' a circle. The (arithmetic) mean of 1° and 359° is 180° but should be 0°.

3. A more sensible definition of the 'mean direction' is obtained by taking the vector sum of all the component directions. For example, let $\theta = \{\theta_1, \ldots, \theta_i, \ldots \theta_n\}$ be $n$ angles, then the resultant direction is obtained by summing the horizontal and vertical components of unit vectors pointing in these directions (Figure 16):

$$\overline{\theta} = \arctan\left(\frac{\sum_{i=1}^{n}\sin[\theta_i]}{\sum_{i=1}^{n}\cos[\theta_i]}\right) \tag{71}$$

```
ss <- sum(sin(pebbles)) # sum of the sines
sc <- sum(cos(pebbles)) # sum of the cosines
md <- atan(ss/sc)       # mean direction
points.circ(md,pch='x',cex=2)
```

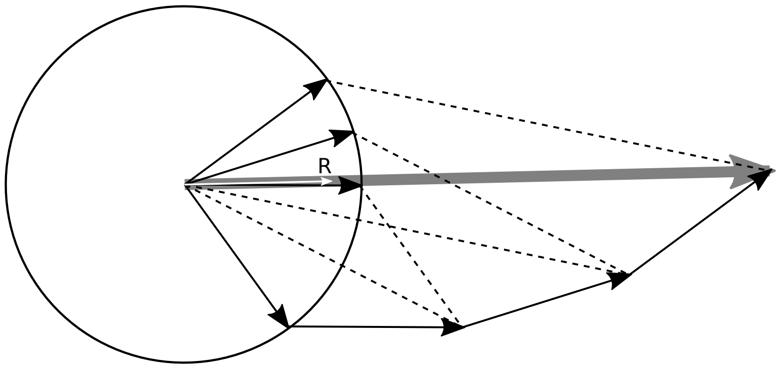

Figure 16: An average direction (grey arrow) is obtained by taking the vector sum of four angular measurements (black arrows). Scaling by the number of measurements (white arrow) yields a measure of concentration ($0 \leq R \leq 1$) that is related with (but not identical to) von Mises' $\kappa$-parameter.

4. The normal distribution is not appropriate for directional data for the same reason why it did not apply to compositional data. Its tails go from $-\infty$ to $+\infty$ and do not fit within the constrained dataspace of the angles. The *von Mises* distribution does not suffer from this problem:

$$f(\theta|\mu,\kappa) \propto \exp[\kappa\cos(\theta-\mu)] \tag{72}$$

where $\mu$ is the location parameter (the mean angle) and $\kappa$ is the *concentration parameter*. As the name suggests, large $\kappa$-values correspond to narrow distributions, and small $\kappa$-values to wide ones. In contrast, the *circular standard deviation* is defined as

$$s_c = \sqrt{\ln(1/R^2)} \tag{73}$$

where

$$R^2 = \left(\sum_{i=1}^{n}\sin[\theta_i]/n\right)^2 + \left(\sum_{i=1}^{n}\cos[\theta_i]/n\right)^2 \tag{74}$$

$R$ is the normalised length of the vector sum (the white arrow in Figure 16). This arrow shortens with increasing scatter and so $R^2$ is a dispersion parameter. In the extreme case where all the measurements point in the same direction, $R = 1$ and $s_c = 1$. If the data are evenly distributed around the circle, $R = 0$ and $s_c = \infty$.

5. These concepts are easily generalised from the 1-dimensional circle to a 2-dimensional sphere. The coordinates of data in this space can be expressed as latitude ($\theta$) and longitude ($\phi$):

$$\begin{cases} x = \sin\theta\cos\phi \\ y = \sin\theta\sin\phi \\ z = \cos\theta \end{cases} \tag{75}$$

as strike (S) and dip (D):

$$\begin{cases} x = -\sin S\cos D \\ y = \cos S\cos D \\ z = \sin D \end{cases} \tag{76}$$

or as dip (D) and azimuth (A):

$$\begin{cases} x = \cos D\cos A \\ y = \cos D\sin A \\ z = \sin D \end{cases} \tag{77}$$

6. The following code plots some palaeomagnetic data (declination and inclination) onto a Schmidt stereonet using the `RFOC` package:

```
library(RFOC)                                       # load the package
d <- read.csv(file='palaeomag.csv',header=TRUE) # load the data
net()                                               # set up the stereonet
focpoint(d$dec,d$inc)                               # plot the data
```

To calculate the 95% confidence ellipse for the average orientation:

```
ci = alpha95(d$dec,d$inc)                # get the confidence interval
addsmallcirc(ci$Dr, ci$Ir, ci$Alph95)   # plot as a circle
```

where `ci$Dr` and `ci$Ir` are the resultant declination and inclination of the mean, which are obtained by vector summation just like in Figure 16.

## 12 Time series

Time is a fundamental parameter in geology. Few people outside the Earth Sciences truly appreciate the vast expanses of Geological Time that have shaped our planet. Understanding and reconstructing the evolution of our planet through time is a key objective of the Earth Sciences. This Chapter will give the briefest of introductions to time series analysis in the context of palaeoclimatology.

1. To illustrate the concept of a frequency spectrum, let us create a synthetic time series by summing two periodic functions `y1` and `y2` that run from 0 to $2\pi$:

```
n <- 200
x <- seq(from=0,to=2*pi,length.out=n)
y1 <- 3*sin(2*x)
y2 <- 2*cos(4*x)
y3 <- y1+y2
matplot(x,cbind(y1,y2,y3),type='l')
```

where `matplot` combines multiple lines on the same plot. Function `y1` has an amplitude of 3 and a frequency of 2. Function `y2` has an amplitude of 2 and a frequency of 4, and is shifted by a phase difference of $\pi/2$ (i.e., 90°) w.r.t. `y1`. The sum of `y1` and `y2` produces a new function `y3`. This is a so-called *Fourier Series*, named in honour of its inventor, the French mathematician Joseph Fournier (1768 – 1830), who also made seminal contributions to the understanding of heat flow. `y3` is more complex than either `y1` or `y2`. It is easy to see that by superimposing additional sines and cosines, a great variety of functions can be created. See `https://youtu.be/qS4H6PEcCCA` for a spectacular example of this.

2. Suppose that we are given `y3` without any information about `y1` and `y2`. We can deconvolve `y3` to its constituents `y1` and `y2` using a *Fourier Transformation*. In `R`:

```
s <- spectrum(y3,plot=FALSE)
plot(s$freq,s$spec,type='l')
```

`spectrum` returns a list (`s`) containing the frequency (`freq`) and spectral density (`spec`) of the component functions. The frequency equals the reciprocal of the sampling interval, whereas the spectral density is related to the amplitude of the component functions.

3. The output of the previous code snippet shows two distinct low frequency peaks. Let us zoom into this part of the spectrum and add vertical lines marking the known frequencies of `y1` and `y2`:

```
plot(s$freq,s$spec,type='l',xlim=c(0,0.1))
usr = par("usr")
lines(rep(2/n,2),usr[c(1,4)],col='blue')
lines(rep(4/n,2),usr[c(1,4)],col='red')
```

where `usr` contains the axis limits of the currently active plot, `2/n` is the frequency of `y1` (blue line) and `4/n` is the frequency of `y2` (red line). You can see that the Fourier analysis has successfully recovered these frequencies, as well as the relative amplitudes of the two component functions.

4. Let us now apply this same procedure to a real dataset of $\delta^{18}$O-measurements in benthic foraminifera (Lisiecki and Raymo, *Paleoceanography*, 2005). Load the data from a `.csv` file and plot them as a time series:

```
LR04 <- read.csv('LR04stack.csv',header=TRUE)
plot(LR04$ka,LR04$d18O,type='l',xlab='time (ka)',ylab='d18O')
```

Before we proceed, it is important to note that the time series is not sampled at a constant rate. Use the `diff()` function to inspect the sampling interval:

```
> diff(LR04$ka)
```

we can see that datapoints 1–601 are spaced at a 1 ka time interval, points 602–1050 are spaced 2 ka apart, points 1051–1651 2.5 ka and 1652–2114 5 ka. This decreasing resolution of the time series reflects the effects of sedimentary compaction and diagenesis on the benthic record.

5. In order to perform a Fourier analysis using R's built-in `spectrum()` function, we need to ensure that our data are sampled at regular intervals. To avoid this problem, we will only consider the first 601 datapoints for the Fourier analysis:

```
s <- spectrum(LR04$d18O[1:601],plot=FALSE)
plot(s$freq,s$spec,type='l',xlim=c(0,0.1),
     xlab='frequency (1/ka)',ylab='spectral density')
```

   The spectrum shows three peaks at frequencies 0.01, 0.024 and 0.043, respectively. These correspond to the three orbital cycles of the Milankovitch series, at 100 ka (ellipticity), 41 ka (obliquity) and 23 ka (precession). Mark these three periods on the power spectrum:

```
usr = par("usr")
lines(rep(1/100,2),usr[c(1,4)],col='blue')
lines(rep(1/41,2),usr[c(1,4)],col='red')
lines(rep(1/23,2),usr[c(1,4)],col='green')
```

## 13 Spatial data

"everything is related to everything else, but near things are more related than distant things."

– Waldo R. Tobler (1969)

The previous Section discussed time as a fundamental parameter in the Earth Sciences. This Section deals with space, which is equally important. More specifically, it introduces a method called *kriging* to interpolate sparse geological measurements in two (or more) dimensions.

1. In the summer of 2018, I carried out some geomorphological observation on the Gong He Plateau, in northeastern Tibet. This part of the world suffers from severe land degradation as a result of overgrazing. This manifests itself as blowout dunes, which consist of saucer-shaped deflation bowls of up to 20 m depth, and deflation lobes of sand that has been transported downwind from the deflation bowl. With the help of students, I measured the thickness of the wind-blown sand at 89 locations in a small deflation lobe. These measurements are provided in a file called `pits.csv`. The following code loads the data and parses the columns into three variables for further manipulation:

```
pits <- read.csv('pits.csv',header=TRUE,row.names=1)
lat <- pits[,'lat']
lon <- pits[,'lon']
th <- pits[,'thickness']
```

   Let us plot these data on a map and colour code them for sand thickness:

```
np <- length(th)                       # number of pits
r <- (th-min(th))/(max(th)-min(th)) # red
g <- 1-r                               # green
b <- rep(0,np)                         # blue
plot(lon,lat,pch=21,bg=rgb(r,g,b))
```

2. This diagram shows that the thickness of the depositional lobe is the greatest in the middle (red), and thins out near the edges (green). However, we do not have any information about the areas between the sample points. Kriging is one way to 'fill in the blanks' and interpolate the measurements between the samples to create a continuous map. Kriging is not part of `R`'s core functionality but there are many packages that implement it. We will use a basic and light-weight package called `kriging` but several more sophisticated options are available as well:

```
library(kriging)
k <- kriging(lon,lat,th)
```

3. Let us have a look at the output of the `kriging()` function:

```
> plot(k)
```

   which produces a so-called *semivariogram*. This plot graphically expresses Waldo's *First Law of Geography*, which was quoted at the beginning of this Section. It shows that the variance between a sample and its neighbours increases with increasing distance. Kriging fits a mathematical function to the semivariogram. By default, the `kriging` package uses a circular function to do so. The intercept of this circle with the y-axis of the semivariogram is called the *nugget*. The variance increases until it reaches a plateau (the *sill*). At distances exceeding the *range*, the variance between samples remains constant.

4. Given the best analytical fit to the semivariogram, the kriging algorithm then interpolates the data between the measurements as follows:

$$z(s_\circ) = \sum_{i=1}^{n} \lambda_i z(s_i) \tag{78}$$

   where $s_i$ is the $i^{\text{th}}$ (out of $n$) sites, $z(s_i)$ is the measured thickness at this location, and $z(s_\circ)$ is the predicted thickness at some new location $s_\circ$. $\lambda_i$ is a weighting parameter that depends on the distance between locations $s_i$ and $s_\circ$, and the spatial correlation at that distance. The latter parameter is captured by the analytical fit to the semivariogram. Applying Equation 78 over a grid of points yields an interpolation map:

```
> image(k)
```

5. Use the `lattice` package to produce a visually more pleasing kriging map:

```
library(lattice)
levelplot(pred ~ x + y, data=k$map)
```

   which adds a colour scale to the interpolation map. Note how the inferred sand thickness outside the sampling area contains negative values. This reflects the fact that the kriging algorithm is intended for interpolation and is less reliable for extrapolation.