Part 3 Section 2: Probability Density Functions

Aaron Robotham

Libraries needed for this chapter: magicaxis, foreach, ellipse, mvtnorm.

```
library(magicaxis, quietly=TRUE)
library(foreach, quietly=TRUE)
library(ellipse, quietly=TRUE)
```

```
##
## Attaching package: 'ellipse'
```

```
## The following object is masked from 'package:graphics':
##
##     pairs
```

```
library(mvtnorm, quietly=TRUE)
```

So everybody running this chapter gets the same outputs, we should first set the seed.

```
set.seed(1)
```

Quick Reminder on the Mean, Median, Mode

Given a distribution of data you should be aware of the three important types of average: mean, median and mode.

The mean of a distribution x is your classic average sum of values divided by number of values, i.e.:

$$\mathrm{E}(x) = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

This can be done explicitly in **R**, but we would usually use the inbuilt **mean** function:

```
x_data = c(1,5,3,7,2,8,4,3,6)
sum(x_data)/length(x_data)
```

```
## [1] 4.333333
```

```
mean(x_data)
```

```
## [1] 4.333333
```

The median is the middle most point of the sorted values of x, or for a given PDF the value of x at which the integral reaches 0.5. There is no completely standardised notation for the median, but one form (which we shall use) is $\mu_{0.5}$, since this can be generalised to other quantiles beyond the 50^{th} percentile.

```
sort(x_data)[5]
```

```
## [1] 4
```

```
median(x_data)
```

```
## [1] 4
```

```
quantile(x_data,0.5)
```

```
## 50%
##   4
```

The mode means two different things in practice. For fairly discretised data it is the most commonly occurring value (so in the case above for x_data this would 3). The other definition is the peak of the PDF formed when smoothing the data with an appropriate kernel. If this seems a bit vague, then that is because it is. We will see an example of this latter definition later though. As for the former, surprisingly **R** does not offer an explicit **mode** function, perhaps because of the ambiguity. Instead, we can use the **tabulate** function to compute the frequency of all occurrences of values, and then find the maximum of this.

```
x_tab =  tabulate(x_data)
x_tab
```

```
## [1] 1 1 2 1 1 1 1 1
```

```
which.max(x_tab)
```

```
## [1] 3
```

Comparing Averages

If we create a deliberately skewed sample distribution, we can see how these averages vary for non-symmetrical data:

```
x_samp = rpois(1e3,lambda=3)^2
x_mean = mean(x_samp)
x_median = median(x_samp)
x_mode = which.max(tabulate(x_samp))
```

```
maghist(x_samp, xlab='x', ylab='Counts')
```

```
## Summary of used sample:
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    4.00    9.00   12.23   16.00  144.00
```

```
## Pop Std Dev: 13.537
```

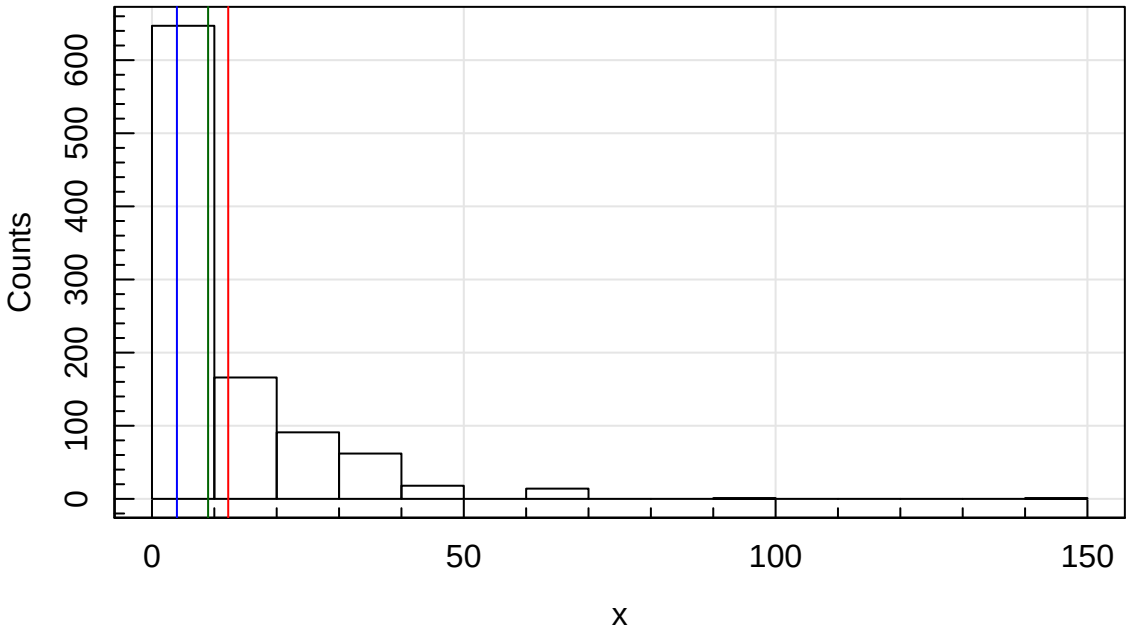
```
## MAD: 10.378
```

```
## Half 16-84 Quan (1s): 12
```

```
## Half 02-98 Quan (2s): 24.5
```

```
## Using 1000 out of 1000
```

```
abline(v=x_mean, col='red')
abline(v=x_median, col='darkgreen')
abline(v=x_mode, col='blue')
```



We see the mean is dragged towards the long tail of the distribution, and the mode is the lowest of the three. The median is usually the middle of the three averages (when they do differ), and it is also usually considered to be the most robust since it is not highly sensitive to outliers, which always affect the mean.

Standard Deviation, Variance and Co-Variance

Once you have computed the mean of a distribution the next most powerful summary statistic is the variance or standard-deviation (which is just the $\sqrt{var}$). The variance is defined as:

$$\mathrm{Var}(x) = \mathrm{E}[(x - \mathrm{E}(x))^2],$$

where $E(x)$ is the sample mean, as above. We can compute this easily in **R**:

```
mean((x_samp - x_mean)^2)
```

```
## [1] 183.056
```

```
var(x_samp)
```

```
## [1] 183.2393
```

Notice the two values differ by a small amount. The reason for this is that the sample mean, whilst unbiased in a global sense (i.e. it will give the population mean) is always a bit biased per sample and is dragged systematically towards where the data happens to be given N samples, rather than where it would be if we have infinite samples (the idealised population). It turns out (i.e. a proof exists, but we do not cover it here) that to correct for this effect with Normally distributed data the results should be scaled by a factor of $N/(N-1)$:

```
mean((x_samp-x_mean)^2)*length(x_samp)/(length(x_samp) - 1)
```

```
## [1] 183.2393
```

In both cases (sample variance or population variance) the standard deviation is just $\sigma_x = \sqrt{Var(x)}$. When using **R** users should be aware that the variance and standard deviation is always corrected for this $N/(N-1)$ factor, i.e. it attempts to return an unbiased population variance or standard deviation. E.g.:

```
sqrt(mean((x_samp-x_mean)^2)*length(x_samp)/(length(x_samp) - 1))
```

```
## [1] 13.53659
```

```
sd(x_samp)
```

```
## [1] 13.53659
```

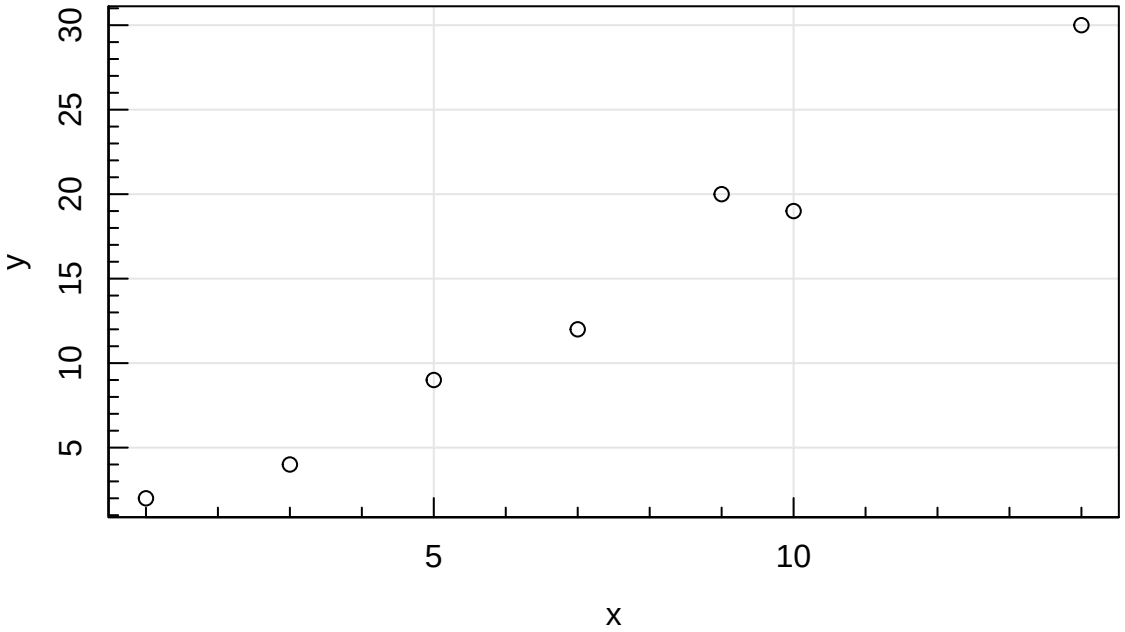
The concept of distributions of sample means is key to the concept of the central limit theorem, which we discuss later in detail.

The final concept to remind ourselves about (perhaps for the first time...) is the sample covariance. This looks like a generalised version of the variance, that includes a second distribution y:

$$\mathrm{cov}(x, y) = \mathrm{E}[(x - \mathrm{E}(x))(y - \mathrm{E}(y))].$$

This means if $x = y$ then you just compute the normal variance. Covariance is really a measure of how much two samples correlate with each other, e,g,:

```
cov_data = cbind(x=c(1,3,5,7,9,10,14),y=c(2,4,9,12,20,19,30))
magplot(cov_data, xlab='x', ylab='y')
```



There is clearly a lot of correlation here (this is something we quantify more rigorously later in the course). Given the above definition, and being careful to scale by the $N/(N-1)$ factor, we can explicitly calculate the covariance:

```
mean((cov_data[,1] - mean(cov_data[,1]))*(cov_data[,2] - mean(cov_data[,2])))*length(cov_data[,1])/(
```

```
## [1] 43.5
```

Or use **R**'s built in **cov** function:

```
cov(cov_data[,1], cov_data[,2])
```

```
## [1] 43.5
```

If $\sigma_x \sigma_y \sim \mathrm{cov}(x, y)$ then the two distribution must be very tightly correlated. This concept of correlation is discussed more later in the course.

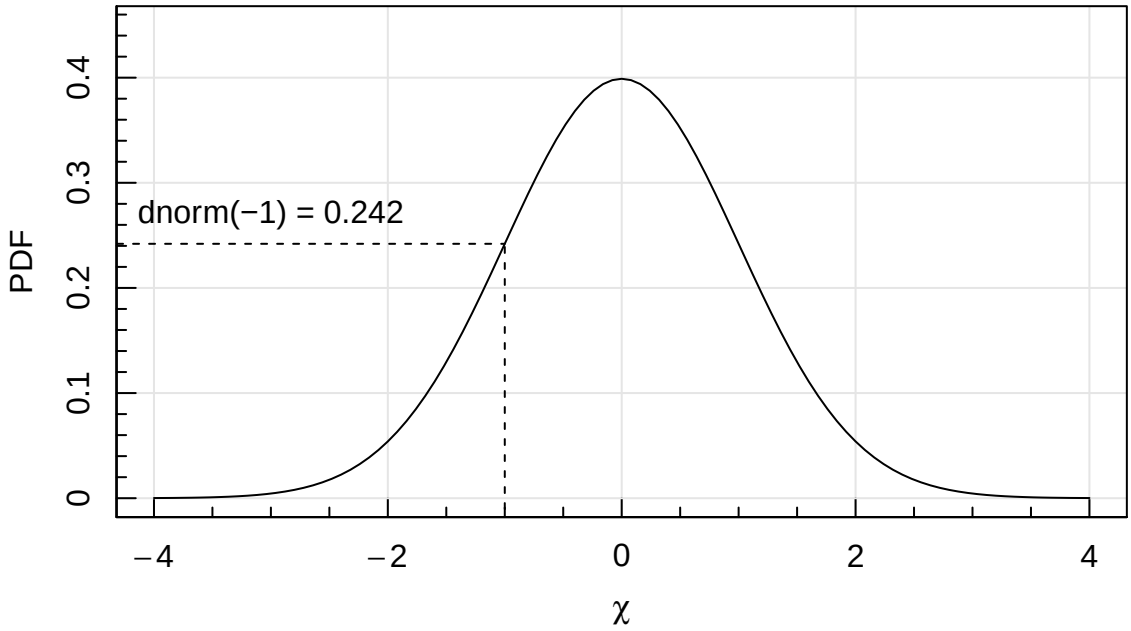
Basics on Distribution Functions

R has *a lot* of support for a huge range of popular statistical distributions, e.g. Uniform, Normal, χ^2, Binomial (we saw this in the last class re coin tosses), Poisson et al. Do '?Distributions' and '??distribution' to see the many available.

Below we discuss a few in detail, but generically **R** offers the same basic 4 functions for interacting with them. These are *d* (density) / *q* (quantile) / *p* (integrated probability) / *r* (random sample) followed by the (usually shortened) name of the distribution (in fact we already saw an example with **dbinom** earlier). No matter this distribution in question, these 4 functions have a consistent meaning in **R**:

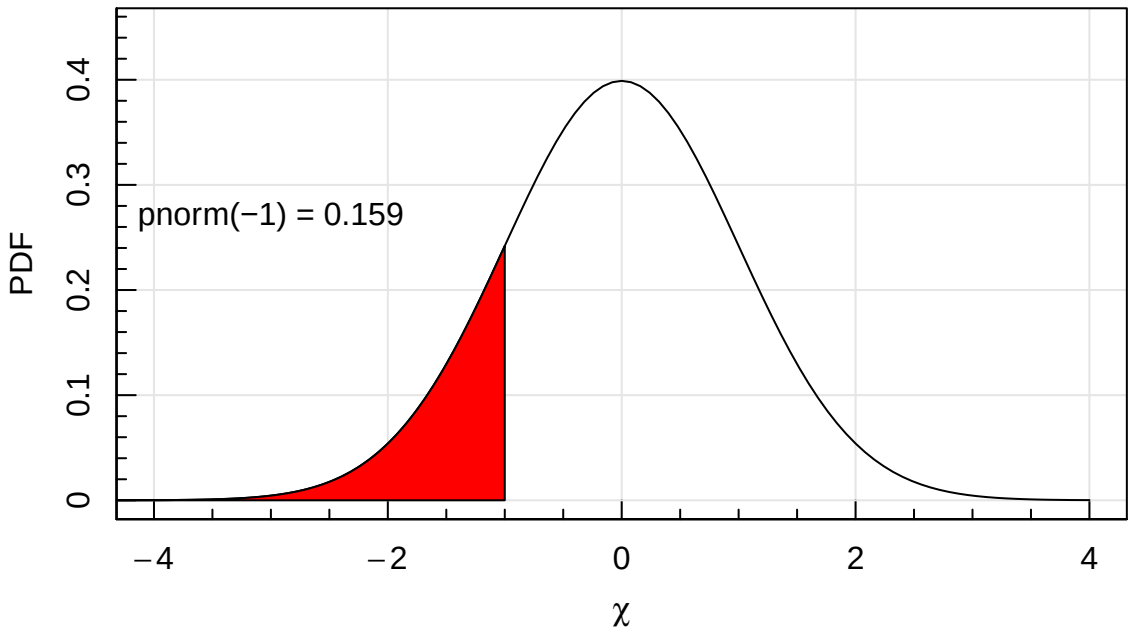
- d???? The instantaneous density at a value (this forms the PDF, where a true PDF must always integrate to 1)

```
magcurve(dnorm, xlim=c(-4,4), ylim=c(0,0.45), xlab=expression(chi), ylab='PDF')
lines(c(-1,-1),c(-1,dnorm(-1)),lty=2)
lines(c(-5,-1),c(dnorm(-1),dnorm(-1)),lty=2)
text(-3,0.27,'dnorm(-1) = 0.242')
```

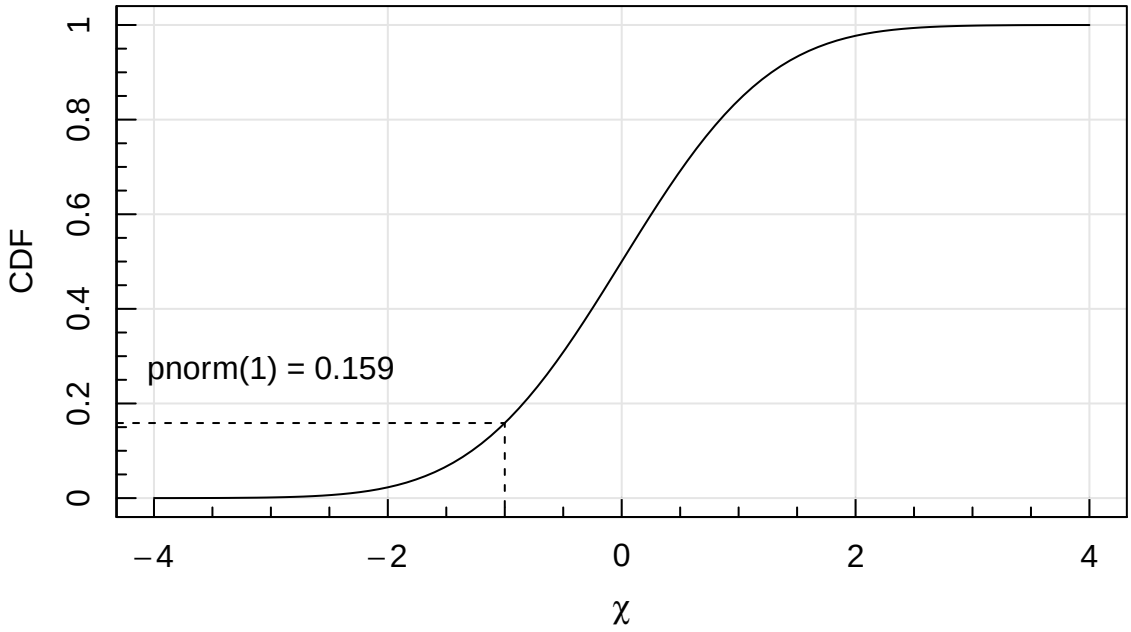


- p???? The integrated probability up to a value (cumulative distribution function, CDF)

```
magcurve(dnorm, xlim=c(-4,4), ylim=c(0,0.45), xlab=expression(chi), ylab='PDF')
polygon(c(-5,-1,seq(-1,-5,by=-0.01),-5), c(0,0,dnorm(seq(-1,-5,by=-0.01)),0), col='red')
text(-3,0.27,'pnorm(-1) = 0.159')
```

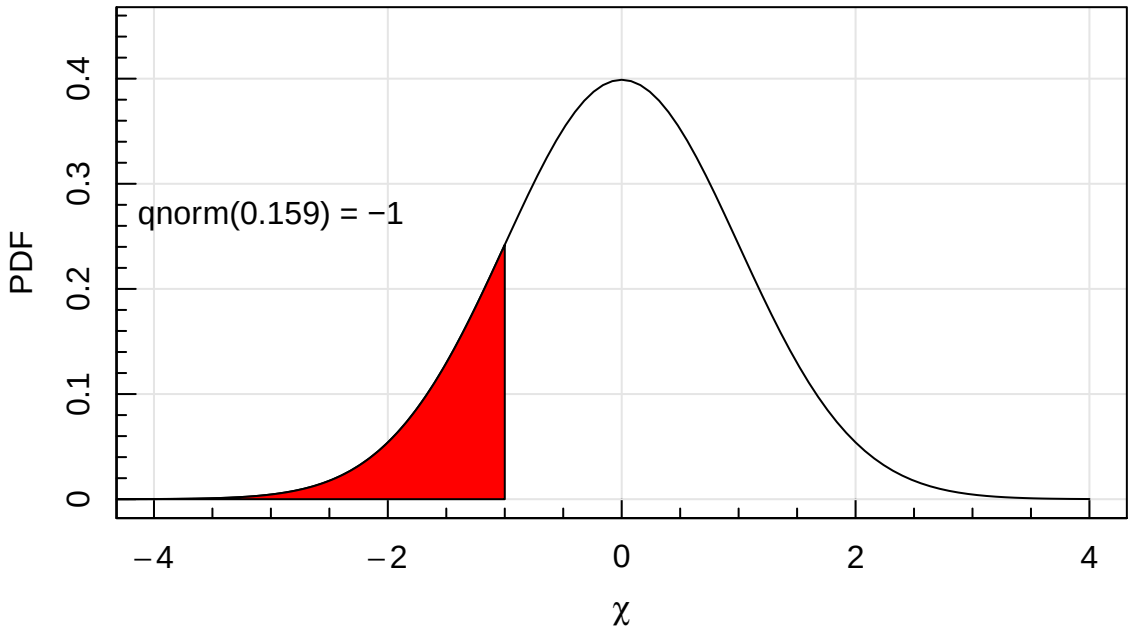


```
magcurve(pnorm, xlim=c(-4,4), ylim=c(0,1), xlab=expression(chi), ylab='CDF')
lines(c(-1,-1),c(-1,pnorm(-1)),lty=2)
lines(c(-5,-1),c(pnorm(-1),pnorm(-1)),lty=2)
text(-3,0.27,'pnorm(1) = 0.159')
```

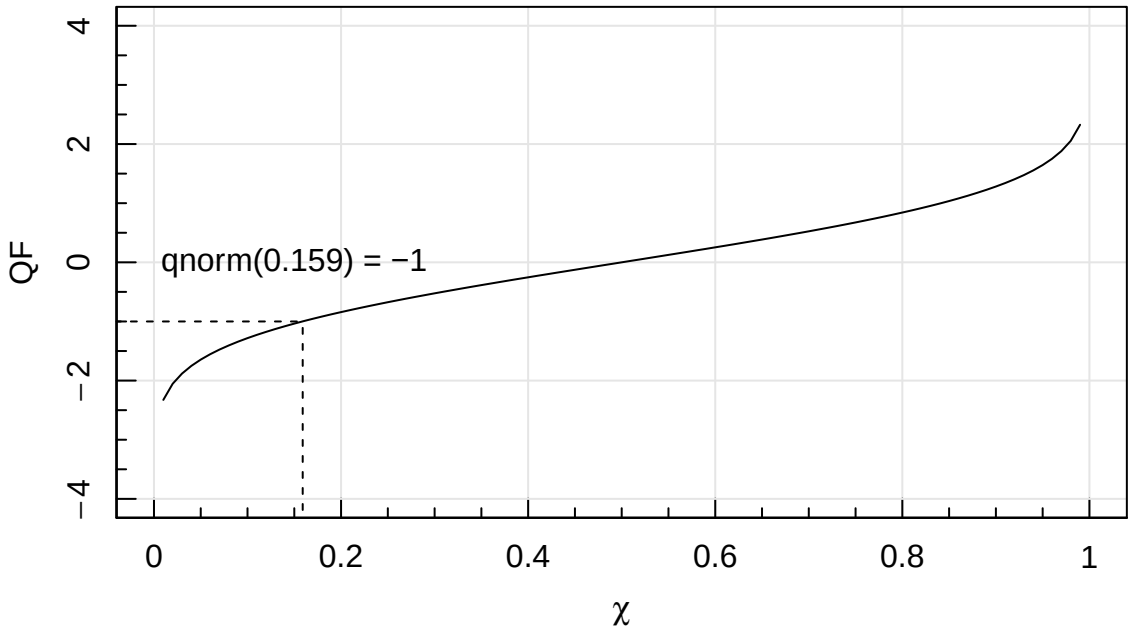


- q???? The value at an integrated probability (aka quantile function, QF)

```
magcurve(dnorm, xlim=c(-4,4), ylim=c(0,0.45), xlab=expression(chi), ylab='PDF')
polygon(c(-5,-1,seq(-1,-5,by=-0.01),-5), c(0,0,dnorm(seq(-1,-5,by=-0.01)),0), col='red')
text(-3,0.27,'qnorm(0.159) = -1')
```

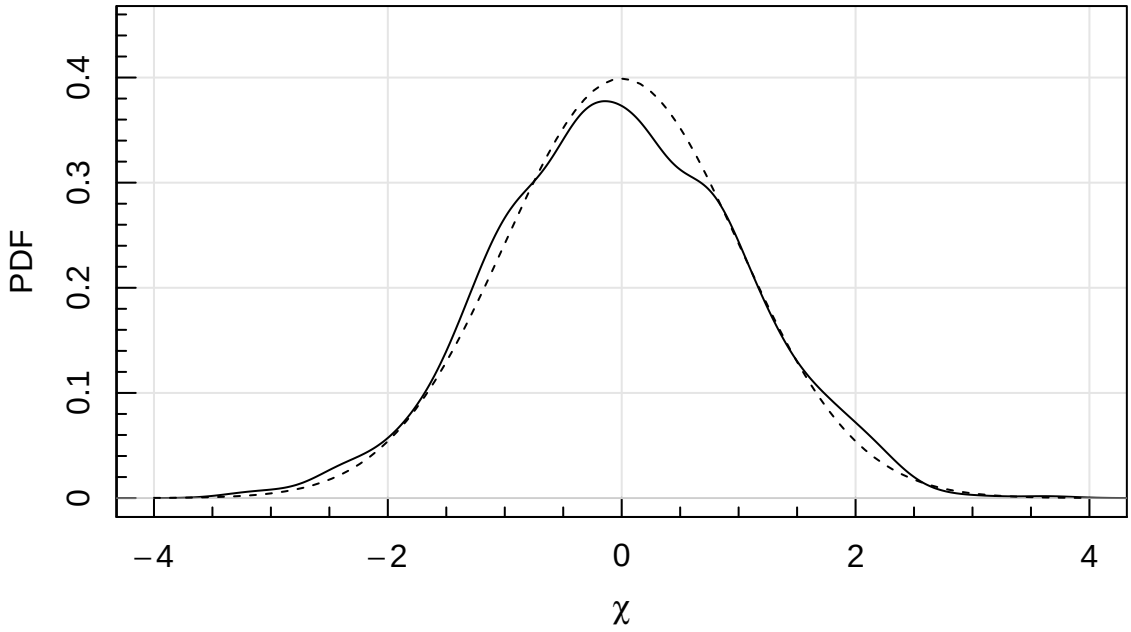



```
magcurve(qnorm, xlim=c(0,1), ylim=c(-4,4), xlab=expression(chi), ylab='QF')
lines(c(0.159,0.159),c(-5,qnorm(0.159)),lty=2)
lines(c(-1,0.159),c(-1,-1),lty=2)
text(0.15,0,'qnorm(0.159) = -1')
```



- r???? A random sampling of the PDF to generate mock data

```
magplot(density(rnorm(1e3)), xlim=c(-4,4), ylim=c(0,0.45), xlab=expression(chi), ylab='PDF')
curve(dnorm, add=TRUE, lty=2)
```



In case it is not completely clear, q???? is the inverse of p????, and vica-versa. There is no generic inverse for d???? since often outcomes would be multi-valued (and it is not a particularly useful quantity to compute).

Base **R** distributions should therefore behave in a predictable manner, with standard options available (such as $\ln(L)$ [log-likelihood] outputs as opposed to linear L [linear-likelihood]). All built in **R** distributions (and as far as I am aware, all extensions) have all 4 function types available, which means there are a lot of powerful tests available to users without coding up often fairly complex (i.e. tedious) functions. They are also very well tested and written in an efficient manner, so can be used with great confidence in general. I have never found a situation where I could not use the base **R** distributions, e.g. I have never needed to recode something in **Rcpp**, so neither should you.

Uniform Distribution

The Uniform distribution is the simplest of all distributions, being a top hat probability distribution between specified limits. It can be specified in a number ways, such as:

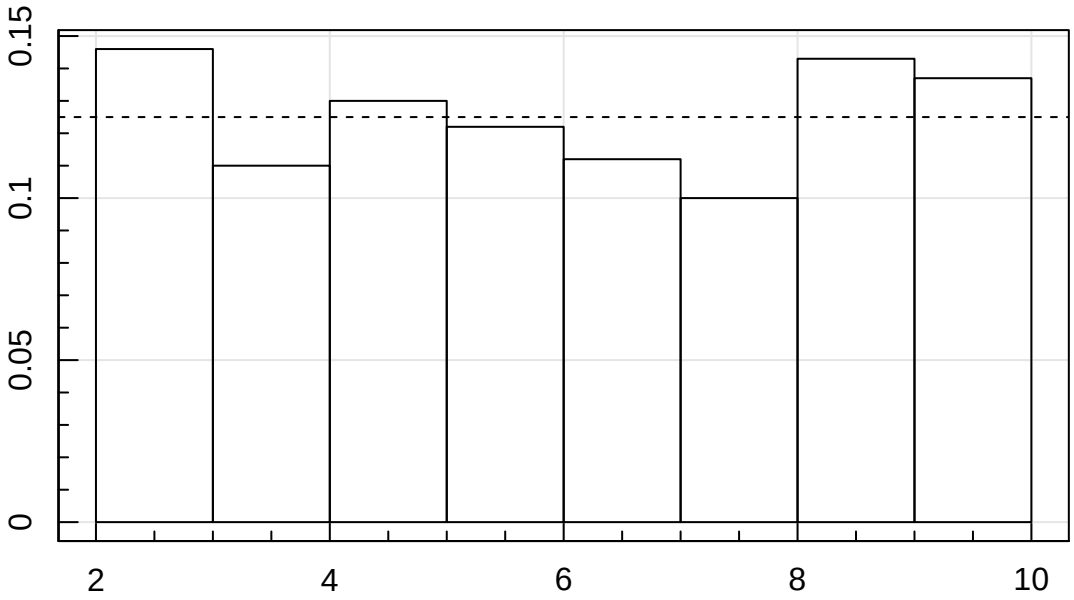
$$p(x) = \frac{1}{b-a} \text{for } a \leq x \leq b,$$

where a is the lower limit and b is the upper limit. In **R** it is used a lot for random number generation, since the default arguments will generate a random number between 0 (min) and 1 (max). Here we will make a different Uniform sample using the built in **runif** function:

```
RanNum = runif(1e3, min=2, max=10)
maghist(RanNum, xlim=c(2,10), freq=FALSE)
```

```
## Summary of used sample:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.009   3.911   5.940   5.988   8.245   9.994
## Pop Std Dev: 2.4108
## MAD: 3.2135
## Half 16-84 Quan (1s): 2.9009
## Half 02-98 Quan (2s): 3.856
## Using 1000 out of 1000 (100%) data points (0 < xlo & 0 > xhi)
```

```
abline(h=1/(10-2), lty=2)
```



Implicitly, the Uniform distribution is probably the most used distribution, since many times by not specifying a decision about a distribution you are actually specifying an improper Uniform distribution (one that was constant probability for all values, but does not integrate to 1, which a true PDF should). We discuss this concept more later in the course when looking at priors, and the philosophy behind them.

Normal Distribution

The Normal distribution is the most important, useful and powerful of all distributions in practical terms. It is abundantly occurring in nature, and should be familiar and fresh in the minds of any budding statistician. It is defined such that

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

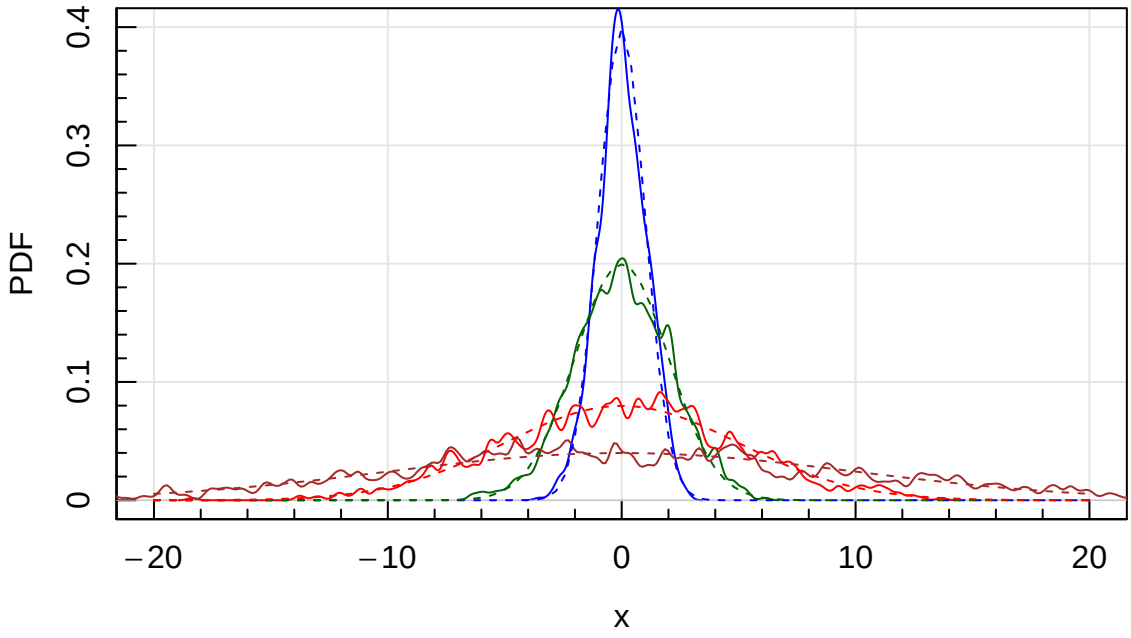
where μ is the mean (or median or mode, they are all the same for the Normal), and σ is the standard deviation (effectively the same type of standard deviation as above, assuming the data is indeed Normally distributed). The part after Euler's number (e) defines the general Gaussian shape of it, the part before correctly *normalises* it (hence the name) so it integrates to 1 over infinite domain. Note in some sloppy fields of research they use 'Normal' and 'Gaussian' as synonyms, but they are not! Whilst the Gaussian function has the same shape, there is no need for it to normalise to any value in particular.

For the Normal distribution we can generate random samples using **rnorm** and compare these to what we would expect using **dnorm**:

```
N_s1 = rnorm(1e3, sd=1)
N_s2 = rnorm(1e3, sd=2)
N_s5 = rnorm(1e3, sd=5)
N_s10 = rnorm(1e3, sd=10)
```

We can then compare our toy data with the Normal built into **R** (**dnorm** function).

```
magplot(density(N_s1, bw=0.2), col='blue', xlim=c(-20,20), ylim=c(0,0.4), xlab='x', ylab='PDF')
lines(density(N_s2, bw=0.2), col='darkgreen')
lines(density(N_s5, bw=0.2), col='red')
lines(density(N_s10, bw=0.2), col='brown')

curve(dnorm(x, sd=1), from=-20, to=20, col='blue', lty=2, add=TRUE)
curve(dnorm(x, sd=2), from=-20, to=20, col='darkgreen', lty=2, add=TRUE)
curve(dnorm(x, sd=5), from=-20, to=20, col='red', lty=2, add=TRUE)
curve(dnorm(x, sd=10), from=-20, to=20, col='brown', lty=2, add=TRUE)
```



The 4 distribution functions we mentioned earlier are powerful tools, especially for the Normal distribution. When testing significance we often want to know how likely it is to observe an event as extreme as x. For instance if we assume our parent distribution is Normal with $\mu = 0$ and $\sigma = 1$, then what is the chance we observe a value of $x < -2$? For this we can use **pnorm**:

```
pnorm(-2)
```

```
## [1] 0.02275013
```

I.e. 2.2750132% of the time we would expect such an observation. Similar we can ask the inverse question, i.e. at what low quantile would we expect to find 5% of the data? To calculate this we can use **qnorm**:

```
qnorm(0.05)
```

```
## [1] -1.644854
```

A useful quantity that astronomers often care about is the range of data that contains $\sim 68\%$ of the data. The reason is that this equates to the integrated probability between $-\sigma$ and σ for the Normal distribution:

```
pnorm(1) - pnorm(-1)
```

```
## [1] 0.6826895
```

Whilst this is only true for the Normal, if the errors in some given results are non-Normal then using the 68% range at least gives an error that is intuitively similar to the common standard deviation type of errors that are assuming Normality.

Central Limit Theorem

The central limit theorem (CLT) is a recurring feature of statistical distributions. The basic thesis is surprising, but we will demonstrate that it is indeed true. In short it states that many measurable quantities of a sample distribution will itself be distributed Normally if we make measurements of multiple independent sample distributions.

As an example we will measure the mean of a Normal sample (size 10) 1000 times and check the distributions:

```
mean_test = foreach(i=1:1e3, .combine='c')%do%{mean(rnorm(10))}
```

Now we can view the output:

```
maghist(mean_test, xlab='x', ylab='Counts')
```

```
## Summary of used sample:
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.95793 -0.22178 -0.01288 -0.01488  0.19931  0.95741
```

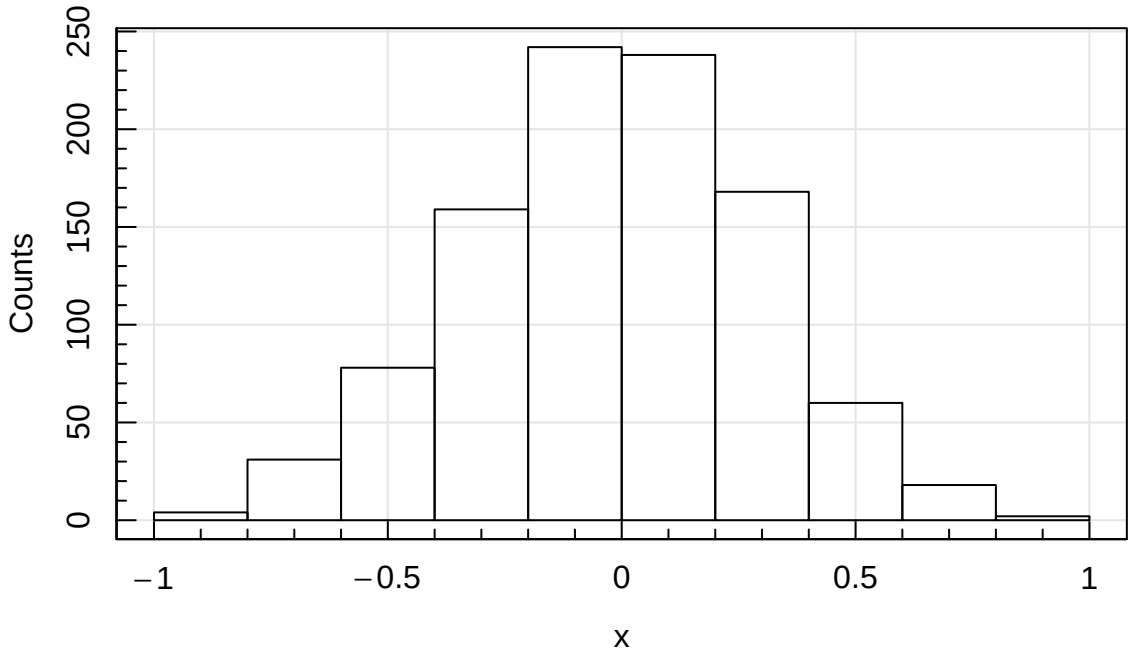
```
## Pop Std Dev: 0.3073
```

```
## MAD: 0.31284
```

```
## Half 16-84 Quan (1s): 0.31819
```

```
## Half 02-98 Quan (2s): 0.61943
```

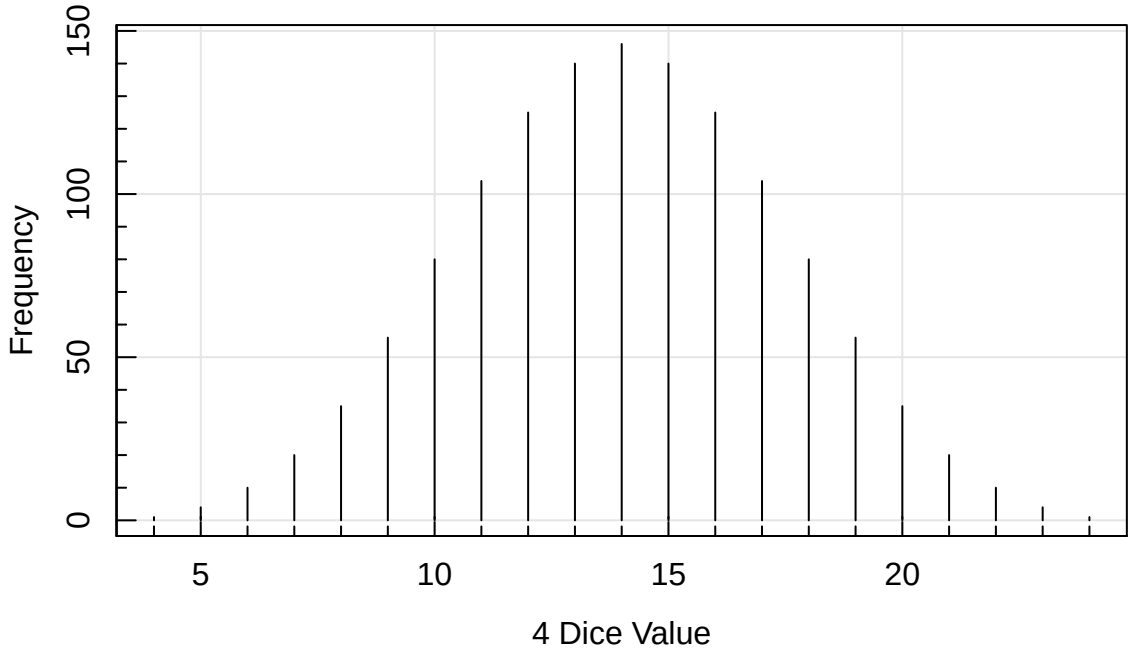
```
## Using 1000 out of 1000
```



There are 3 factors that affect this distribution of sample means: the parent distribution mean and standard deviation, and the number of samples per experiment. Simply put, you would expect that if you make lots more samples you are more likely to measure means and standard deviations nearer to the intrinsic parent distribution parameters ($\mu_p = 0$ and and $\sigma_p = 1$ respectively in this case). The mean of the distribution of means is expected to be 1, but CLT dictates that the standard deviation will actually behave as $\sigma_p/\sqrt{N_s}$. Since $\sigma_p = 1$ and $N_s = 10$ this means we should expect the above distribution to have a standard deviation of $1/\sqrt{10} = 0.316$. We actually measure ~ 0.31.

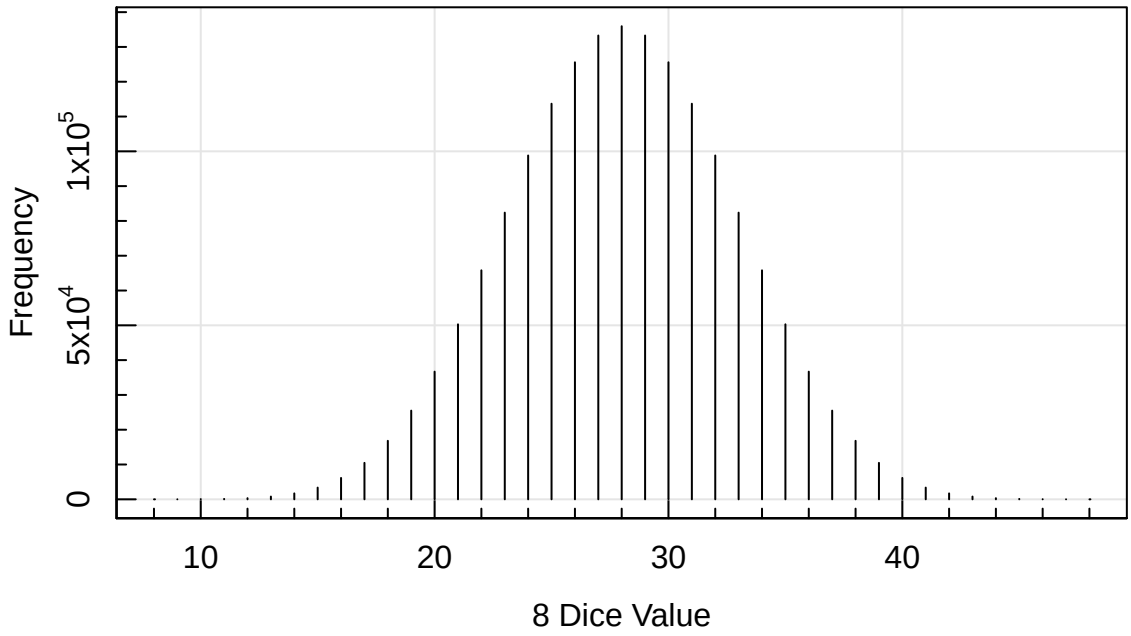
In fact we have seen the consequences of the central limit theorem already in the case of tossing coins and rolling dice. The distribution of likely outcomes, whilst discrete, clearly following a Normal distribution, e.g. to repeat out earlier example:

```
magplot(table(rowSums(expand.grid(1:6,1:6,1:6,1:6))), xlab='4 Dice Value',
        ylab='Frequency')
```



And cranking it up to 8 dice, the Normality becomes very clear:

```
magplot(table(rowSums(expand.grid(1:6,1:6,1:6,1:6,1:6,1:6,1:6,1:6))),
        xlab='8 Dice Value', ylab='Frequency')
```



The concept of the central limit theorem, and tests related to it, will come up at various points in this course. The amazing thing is the more complicated the system, the *more* likely it is for CLT to become apparent. Take for instance the heights of adults in a typical population:

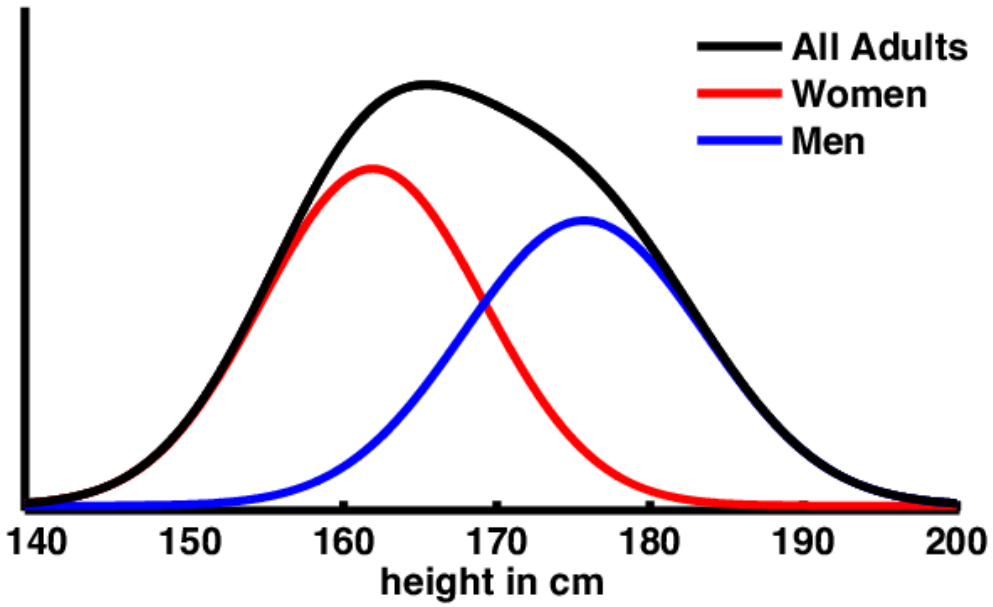


Figure 1: www.researchgate.net/publication/220427369_Disentangling_Gaussians

Despite the incredibly complicated processes involving parent DNA, cellular biology, hormones and environmental effects, the end result is that adult heights are Normally distributed. This is ultimately thanks to CLT being true. In this case, if asked to simulate heights from a population you would do better just to sample from the appropriate Normal than to attempt to model all aspects of physics, chemistry and biology governing the heights of people. As such, in astronomy it is very common that we take a similar short-cut and just parametrise complicated systems as being Normal (e.g. line-of-sight velocity distributions in dark matter halos). Luckily, for good statistical reasons, this approximation is more often

than not close to being true.

Multivariate Normal Distribution

It is very common to see the concept of the Normal distribution expanded out to higher dimensions. In fact it is often a desirable feature for errors and model parameters to possess covariate Normal behaviour.

In its most compact form a sample from the multivariate Normal (MvtN) distribution can be written as:

$$X = (X_1, ..., X_k)^T, X \sim N_k(\mu, \Sigma),$$

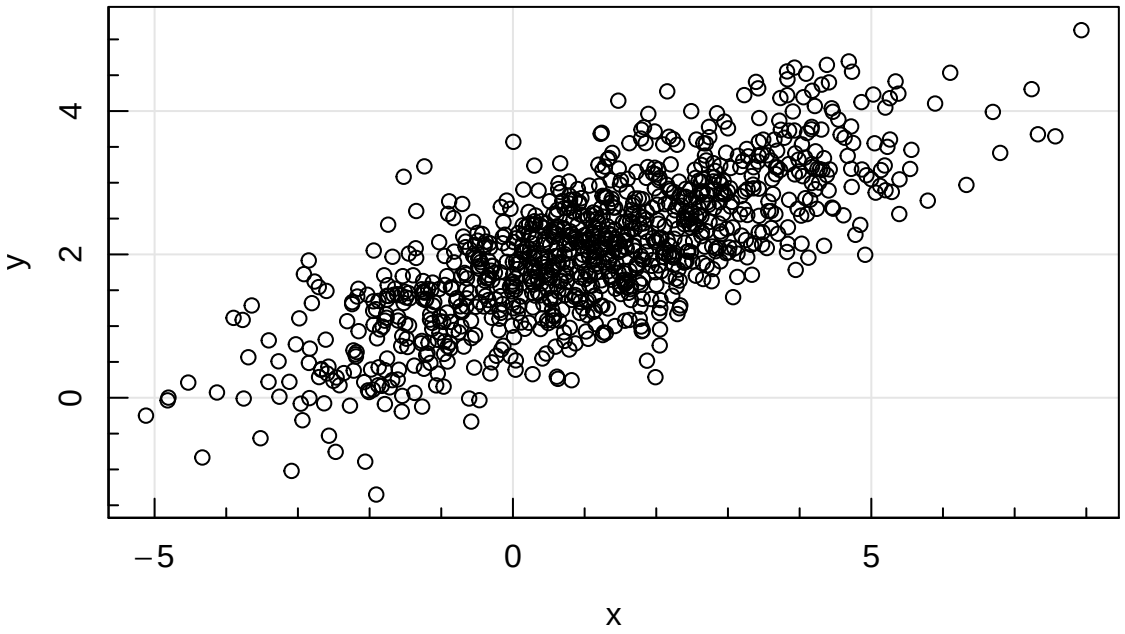
where X is then a random sample of the MvtN for a given vector of length k of means (μ), and a covariance matrix of size $k \times k$ (Σ). We discuss the concept of a covariance matrix in a lot of detail elsewhere in this course, but the key concept is it communicates how broad the Normal distribution is across each dimension (the diagonal elements) and how correlated the different dimensions are (the off-diagonal terms, as we saw in the early example of correlated data).

R does not come with base functions that deal directly with the MvtN distribution, but here we make use of the **mvtnorm** package, which contains all of our standard distribution function routines. Using this we can easily make 1,000 samples of the MvtN in 2D:

```
covmat = cbind(c(4,1.5),c(1.5,1)) #take a look at covmat yourself
tempcov = rmvnorm(1e3, mean=c(1,2), sigma=covmat)
```

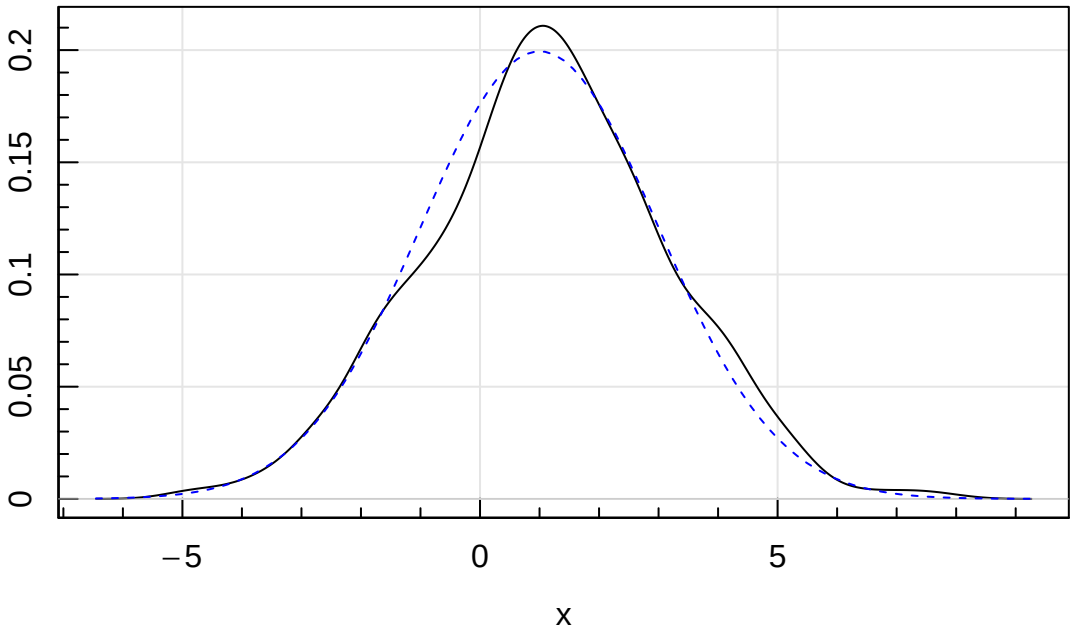
We can plot the above to see the distribution we have sampled in 2D:

```
magplot(tempcov, xlab='x', ylab='y', asp=1)
```

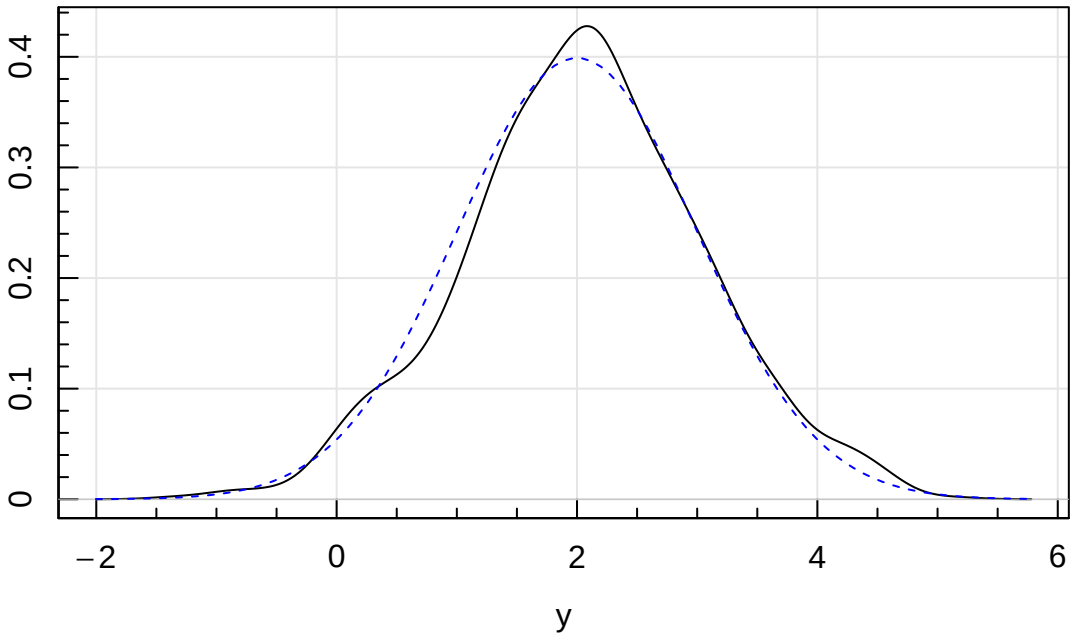


As per the statement above, projections of the MvtN will return what looks like the Normal distribution with specified μ and σ (which will be the square-root of the diagonal variance provided, so 2 and 1 in x and y):

```
magplot(density(tempcov[,1]), xlab='x')
curve(dnorm(x, mean=1, sd=2), col='blue', lty=2, add=TRUE)
```



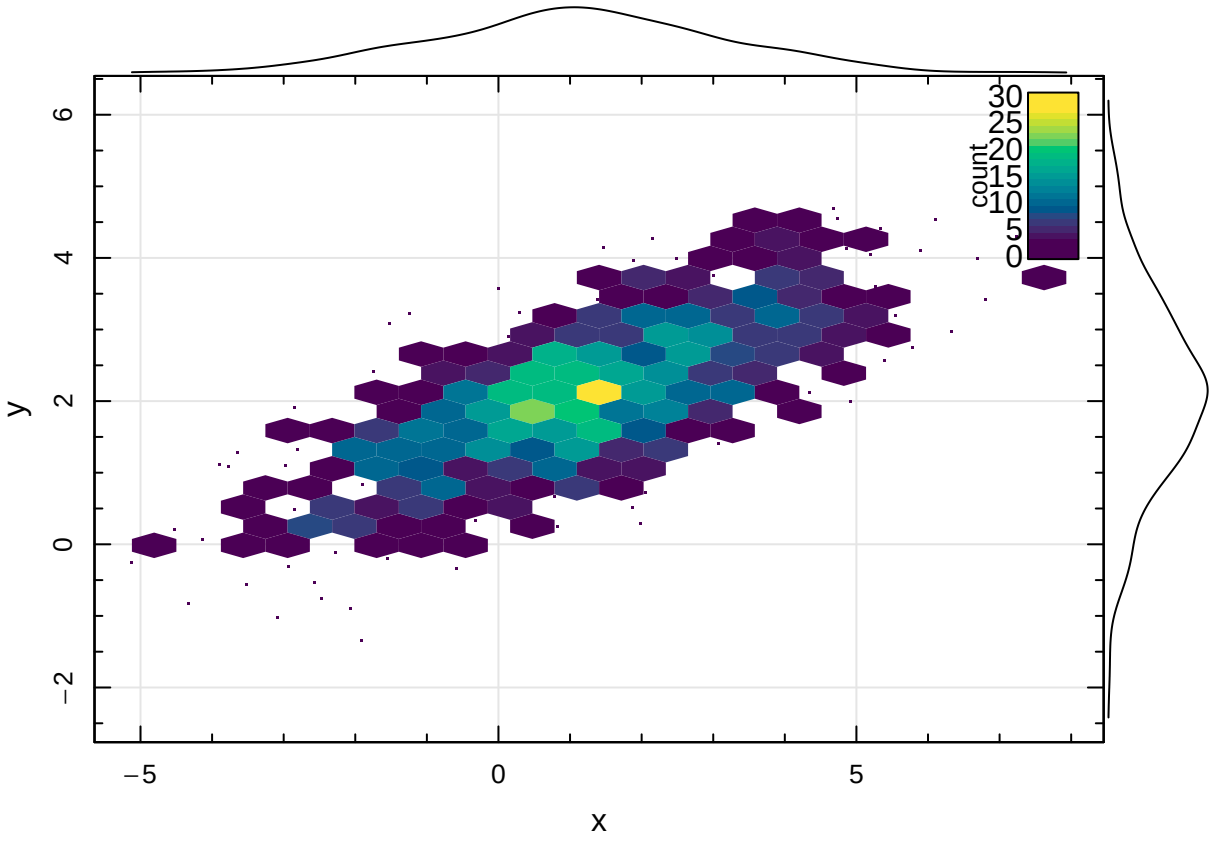

```
magplot(density(tempcov[,2]), xlab='y')
curve(dnorm(x, mean=2, sd=1), col='blue', lty=2, add=TRUE)
```



Given the above, it should be clear that in the case of zeros for the off-diagonal terms, the MvtN behaves the same as making two completely independent samples from the Normal in each dimension.

A convenience plot in the **magicaxis** package helps to reveal this by showing the density of 2D points on a binned colour scale (hexagonal cells by default) and the x/y projections simultaneously:

```
magbin(tempcov, xlab='x', ylab='y', asp=1, projden=TRUE, Nbin=21)
```



Student t Distribution

A concept closely related to the central limit theorem is the Student t distribution. This better explains the expected distribution of sample means when our individual sample sizes are small. Interesting fact, it was popularised by William Gosset under the pseudonym "Student" when he was working for the Guinness brewery. They wanted to better understand variances found in manufacturing processes (say between different days), where you effectively had distributions of averages (one average on each day for a given number of samples). For very large numbers of samples things look Normal (no surprise) but for smaller sample sizes things can look very different.

It is quite a complex distribution, but powerful in application. First of all let us define a new statistics t where for Normally distributed data we have

$$t = \frac{\bar{x} - \mu}{s/\sqrt{N}},$$

where μ is the true population mean, $\bar{x}$ is our sample mean s is our estimator of the population standard deviation (as we have discussed previously, with the $1/\sqrt{N-1}$ correction) and N is the number of observations. t is basically then the distribution of our best guesses of the mean versus the true means scaled by our best guess of the standard deviation.

The Student's t distribution is then defined as

$$p(t; N) = \frac{\Gamma(\frac{1}{2}N)}{\sqrt{\pi(N-1)}\Gamma\left[\frac{1}{2}(N-1)\right]}\left(1 + \frac{t^2}{N-1}\right)^{-N/2}$$

where Γ is the Gamma function, N is the number of observations. Often this is parametrised by the degrees-of-freedom (often written as ν, df, or dof), which for the Student t distribution is given by $\mathrm{df} = \nu = N - 1$.

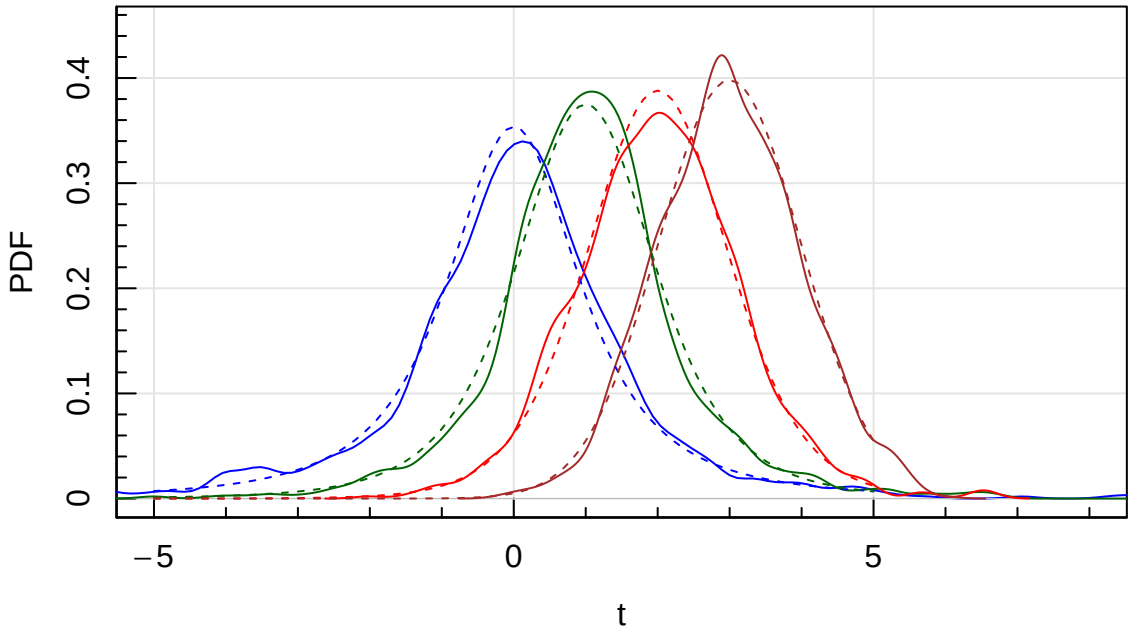
It might not be obvious looking at the above, but the effect is that when N is large this converges towards the Normal distribution, and when N is small it converges towards the Lorentzian (AKA Cauchy) distribution with very broad wings.

This is probably not intuitive from the above, but it is easy to see in action. First we make some mock data:

```
t_N3 = foreach(i = 1:1e3, .combine='c')%do%{temp=rnorm(3); mean(temp)/(sd(temp)/sqrt(3))}
t_N5 = foreach(i = 1:1e3, .combine='c')%do%{temp=rnorm(5); mean(temp)/(sd(temp)/sqrt(5))}
t_N10 = foreach(i = 1:1e3, .combine='c')%do%{temp=rnorm(10); mean(temp)/(sd(temp)/sqrt(10))}
t_N100 = foreach(i = 1:1e3, .combine='c')%do%{temp=rnorm(100); mean(temp)/(sd(temp)/sqrt(100))}
```

We can then compare our mock data with the Student-t distribution built into **R** (**dt** function). We will offset the 4 distributions to make them easier to differentiate, but by definition the t is actually always symmetrically distributed around 0.

```
magplot(density(t_N3, bw=0.2), col='blue', xlim=c(-5,8), ylim=c(0,0.45), xlab='t', ylab='PDF')
lines(density(t_N5+1, bw=0.2), col='darkgreen')
lines(density(t_N10+2, bw=0.2), col='red')
lines(density(t_N100+3, bw=0.2), col='brown')

curve(dt(x, df=2), from=-5, to=5, col='blue', lty=2, add=TRUE)
curve(dt(x-1, df=4), from=-5, to=5, col='darkgreen', lty=2, add=TRUE)
curve(dt(x-2, df=9), from=-5, to=5, col='red', lty=2, add=TRUE)
curve(dt(x-3, df=99), from=-5, to=5, col='brown', lty=2, add=TRUE)
```



The above is noisy (keen students should repeat the above for much larger numbers of realisations), but we can certainly see the effect that lower N samples produce fatter tails for the t statistic.

Such is the nature of real life observations, it is often the case that errors are better represented by Student t statistics. An important feature is that the population variance of a t distribution is related to the degrees of freedom by $\mathrm{Var}(x) = \frac{\nu}{\nu-2}$ for $\nu > 2$.

Z Distribution

A simplified form of the t distribution is the Z distribution, which basically assumes the samples a large enough such that Normality can be assumed once translation and scaling has been applied, and no adaptation needs to be made for the degress-of-freedom in terms of the shape of the PDF. This makes it computationally cheaper to calculate, but these days it is rarely used since computational expense is not a driving factor usually.

χ^2 Distribution

Again related to an aspect of the central limit theorem is the χ^2 distribution. This is the distribution you should expect when summing random squared samples from a Normal distribution. This is a classic quantity used when trying to estimate goodness of fit, when it is formally known as the Pearson χ^2 statistic (we will see this is more detail later in the course). It also tells you have multivariate errors combine to create contingent errors (since you add variances, which are squared errors). First we make a new statistic χ^2:

$$\chi^2 = \sum_{i=1}^{N} \frac{(x_i - \mu)^2}{\sigma^2},$$

where μ and and σ have their usual meaning for a Normal distribution. For the χ^2 distribution the degrees of freedom is given by $\nu = N$. The χ^2 distribution is then defined as (where we set $C = \chi^2$):

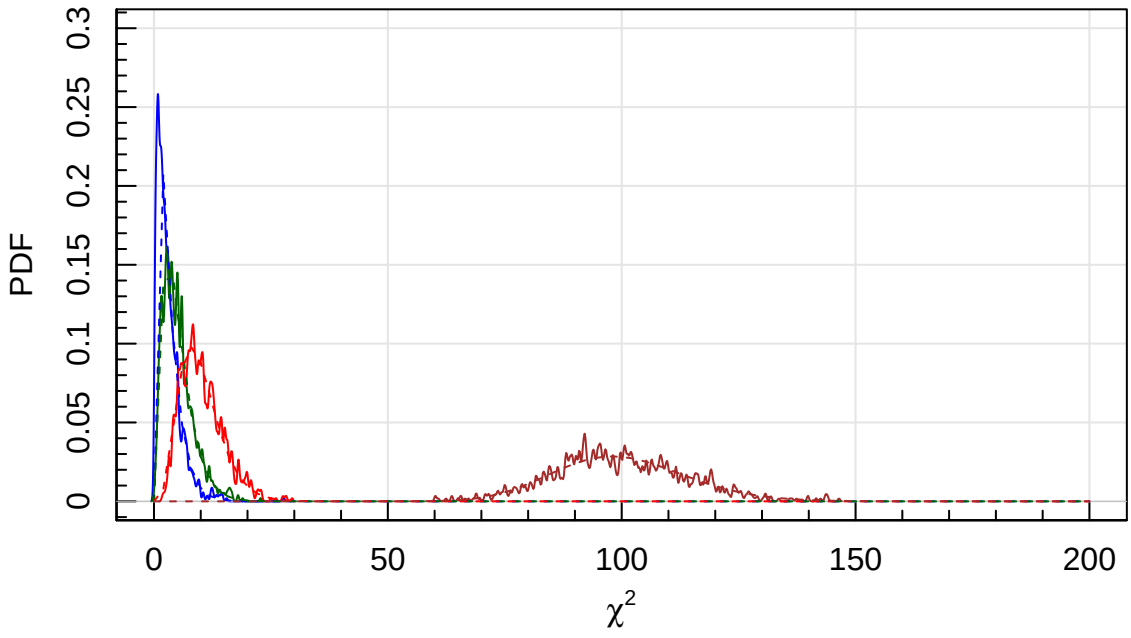
$$p(C;\nu) = \frac{1}{2^{\nu/2}\Gamma(\nu/2)} C^{\nu/2-1} e^{-C/2}$$

As before, it is easiest to see this in action. First make some mock data:

```
chi_N3 = foreach(i = 1:1e3, .combine='c')%do%{sum(rnorm(3)^2)}
chi_N5 = foreach(i = 1:1e3, .combine='c')%do%{sum(rnorm(5)^2)}
chi_N10 = foreach(i = 1:1e3, .combine='c')%do%{sum(rnorm(10)^2)}
chi_N100 = foreach(i = 1:1e3, .combine='c')%do%{sum(rnorm(100)^2)}
```

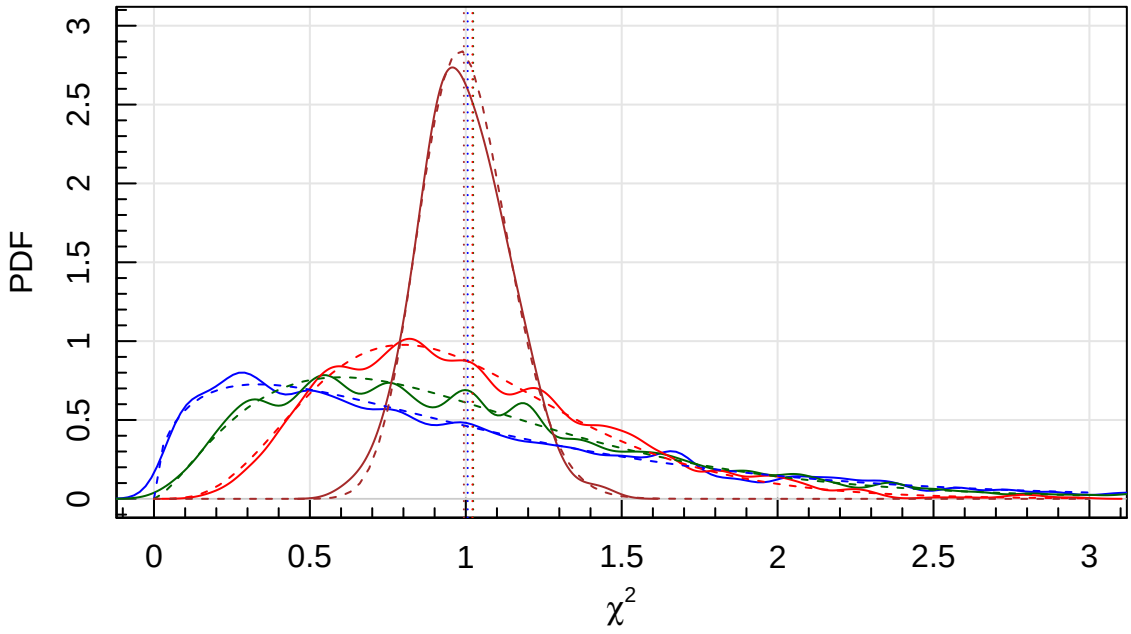
We can then compare our mock data with the χ^2 distribution built into **R** (**dchisq** function).

```
magplot(density(chi_N3, bw=0.2), col='blue', xlim=c(0,200), ylim=c(0,0.3), xlab=expression(chi^2), y
lines(density(chi_N5, bw=0.2), col='darkgreen')
lines(density(chi_N10, bw=0.2), col='red')
lines(density(chi_N100, bw=0.2), col='brown')

curve(dchisq(x, df=3), from=0, to=200, col='blue', lty=2, add=TRUE)
curve(dchisq(x, df=5), from=0, to=200, col='darkgreen', lty=2, add=TRUE)
curve(dchisq(x, df=10), from=0, to=200, col='red', lty=2, add=TRUE)
curve(dchisq(x, df=100), from=0, to=200, col='brown', lty=2, add=TRUE)
```



In astronomy there is a common quantity that is calculated called the "reduced χ^2", which is defined as χ^2/ν. From the below plot it is clear that the mean (so expectation) of this quantity is near to 1.

```
magplot(density(chi_N3/3, bw=0.05), col='blue', xlim=c(0,3), ylim=c(0,3), xlab=expression(chi^2), yl
lines(density(chi_N5/5, bw=0.05), col='darkgreen')
lines(density(chi_N10/10, bw=0.05), col='red')
lines(density(chi_N100/100, bw=0.05), col='brown')

curve(dchisq(x*3, df=3)*3, from=0, to=3, col='blue', lty=2, add=TRUE)
curve(dchisq(x*5, df=5)*5, from=0, to=3, col='darkgreen', lty=2, add=TRUE)
curve(dchisq(x*10, df=10)*10, from=0, to=3, col='red', lty=2, add=TRUE)
curve(dchisq(x*100, df=100)*100, from=0, to=3, col='brown', lty=2, add=TRUE)

abline(v=mean(chi_N3/3), col='blue', lty=3)
abline(v=mean(chi_N5/5), col='darkgreen', lty=3)
abline(v=mean(chi_N10/10), col='red', lty=3)
abline(v=mean(chi_N100/100), col='brown', lty=3)
```



Many papers will include vague statements suggesting that $\chi^2/\nu \sim 1$ is a good thing to see when calculating fitting residuals, but we can show a stricter truth. The mean of the χ^2 for a large number of samples is indeed ν, but the mode is $\nu - 2$ and the median is $\nu\left(1 - \frac{2}{9\nu}\right)^3$. If you have attempted a maximum likelihood fit then it stands to reason the comparison should be with the mode and not the mean, so the more accurate statement will be $\chi^2/(\nu - 2) \sim 1$. Most astronomers do not know this!

Another subtle point is that people often attempt to justify "worse" fits, with $\chi^2/\nu \sim 1$ preferred to $\chi^2/\nu << 1$ and/ $\chi^2/\nu >> 1$. Doing this without invoking the χ^2 distribution itself is a duct tape solution, but it will naturally drop out when correctly applying the χ^2 statistic, i.e. when $\nu = 100$ we would naturally be sceptical of results that produced $\chi^2 = 50$ or $\chi^2 = 150$.

F Distribution

Closely associated with the Student t distribution and the χ^2 distribution is the F distribution.

Conceptually the F distribution can be thought of in two ways: in one it is the ratio of mean squares of independent samples of standard Normals with n_1 and n_2 samples; in the other it is the ratio of two independent χ^2 variates each divided by their degrees of freedom. In either case the F distribution can be used as a test of shared Normality between two independent samples of the same parent distribution.

First we can create our F statistic for Normally distributed data with sample sizes n_1 and n_2, and degrees of freedom $d_1 = n_1 - 1$ and $d_2 = n_2 - 1$:

$$F = \frac{s_1^2}{s_2^2}.$$

The F distribution is then defined as:

$$p(F; d_1, d_2) = \frac{\sqrt{\frac{(d_1 F)^{d_1} d_2^{d_2}}{(d_1 F + d_2)^{d_1 + d_2}}}}{F B\left(\frac{d_1}{2}, \frac{d_2}{2}\right)},$$

where B is the Beta function, defined as $B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a,b)}$. Do not worry, I will not expect you to remember that (and I certainly do not, that's why I wrote these notes).

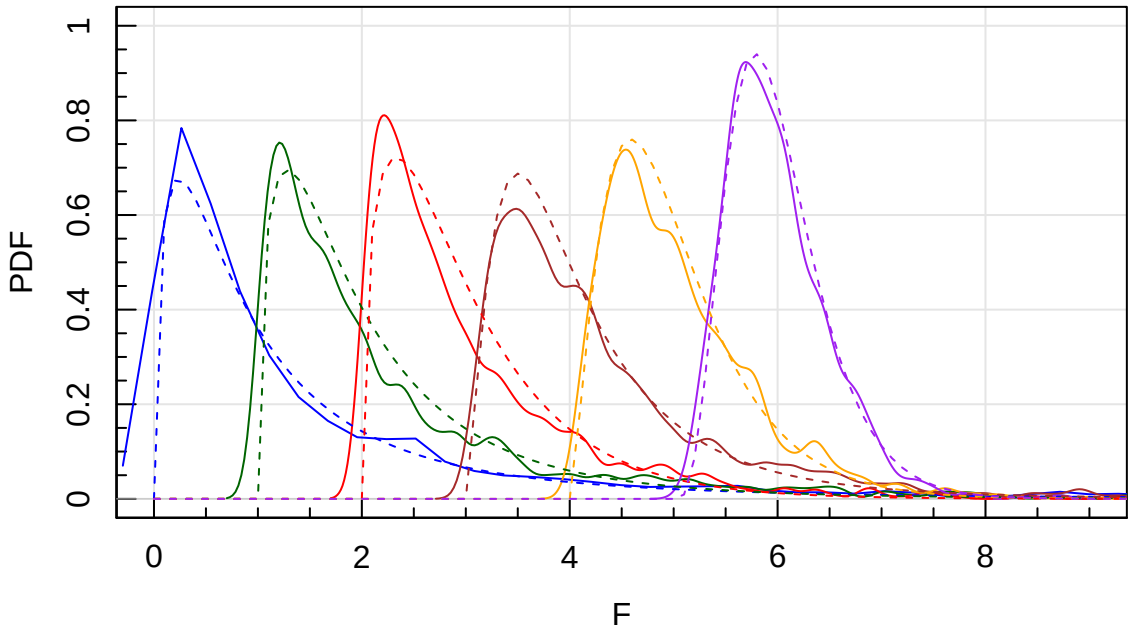
Again, it is easiest to see this in action. First make some mock data:

```
var_N3 = foreach(i = 1:1e3, .combine='c')%do%{var(rnorm(3))}
var_N5 = foreach(i = 1:1e3, .combine='c')%do%{var(rnorm(5))}
var_N10 = foreach(i = 1:1e3, .combine='c')%do%{var(rnorm(10))}
var_N100 = foreach(i = 1:1e3, .combine='c')%do%{var(rnorm(100))}
```

```
F_N3N5 = var_N3/var_N5
F_N3N10 = var_N3/var_N10
F_N3N100 = var_N3/var_N100
F_N5N10 = var_N5/var_N10
F_N5N100 = var_N5/var_N100
F_N10N100 = var_N10/var_N100
```

We can then compare our mock data with the F distribution built into **R** (**df** function). Again, we will offset things for clarity.

```
magplot(density(F_N3N5, bw=0.1), col='blue', xlim=c(0,9), ylim=c(0,1), xlab='F', ylab='PDF')
lines(density(F_N3N10+1, bw=0.1), col='darkgreen')
lines(density(F_N3N100+2, bw=0.1), col='red')
lines(density(F_N5N10+3, bw=0.1), col='brown')
lines(density(F_N5N100+4, bw=0.1), col='orange')
lines(density(F_N10N100+5, bw=0.1), col='purple')

curve(df(x, df1=3, df2=5), from=0, to=10, col='blue', lty=2, add=TRUE)
curve(df(x-1, df1=3, df2=10), from=0, to=10, col='darkgreen', lty=2, add=TRUE)
curve(df(x-2, df1=3, df2=100), from=0, to=10, col='red', lty=2, add=TRUE)
curve(df(x-3, df1=5, df2=10), from=0, to=10, col='brown', lty=2, add=TRUE)
curve(df(x-4, df1=5, df2=100), from=0, to=10, col='orange', lty=2, add=TRUE)
curve(df(x-5, df1=10, df2=100), from=0, to=10, col='purple', lty=2, add=TRUE)
```



The F distribution is used for a distribution comparison test known as the F test, which is discussed in detail later in this course. In short it can tell us whether the ratio between two sample variances is unusually high given the null hypothesis that they share the same parent distribution. The F test is then in itself part of the analysis of variance (ANOVA) omnibus testing procedure that attempts to discover significant Normality departures for samples when conducting a number of experiments. This is useful when trying to determine if a particular treatment is significantly effective given the expected level of sample-to-sample variance when conducting fairly small trials of e.g. drugs.

It might be obvious already to the keen eyed student, but the squared t statistic (t_m^2) is a subset of the F distribution, where $d_1 = m$ and $d_2 = 1$.

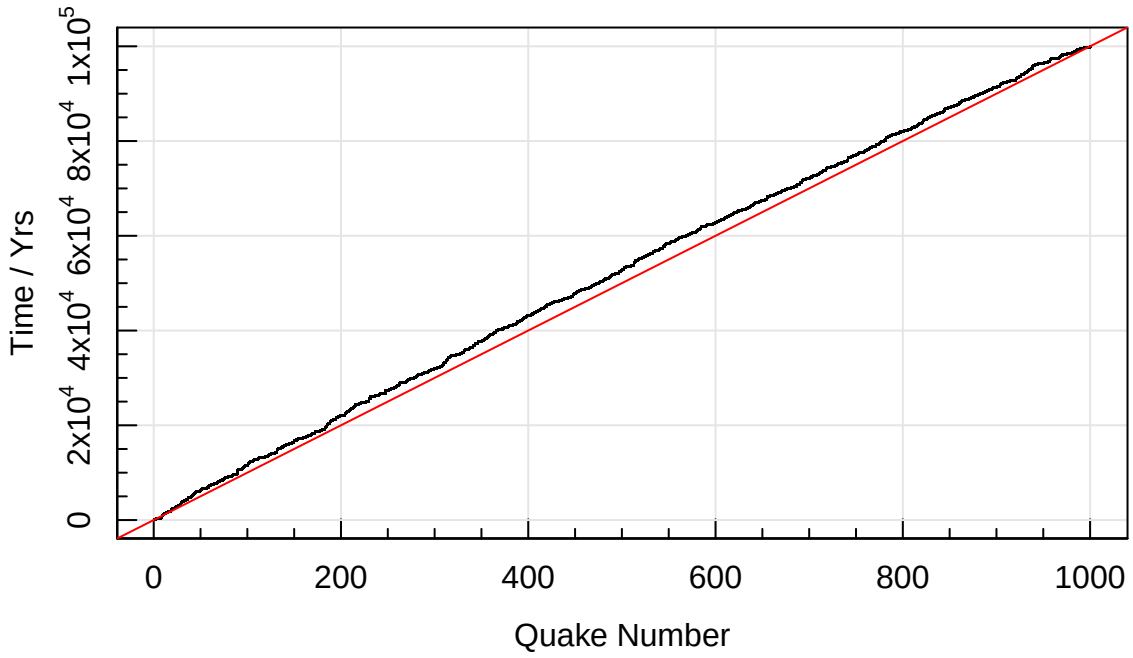
Exponential Distribution

A very important distribution that is relevant to randomly occurring rare events is the exponential distribution. Imagine we have a rare event like an earth quake, where we know the mean rate observed is one every 100 years. This can be trivially simulated by sampling and ordering the results of a Uniform distribution:

```
quakes = sort(runif(1e3, min=0, max=1e5))
```

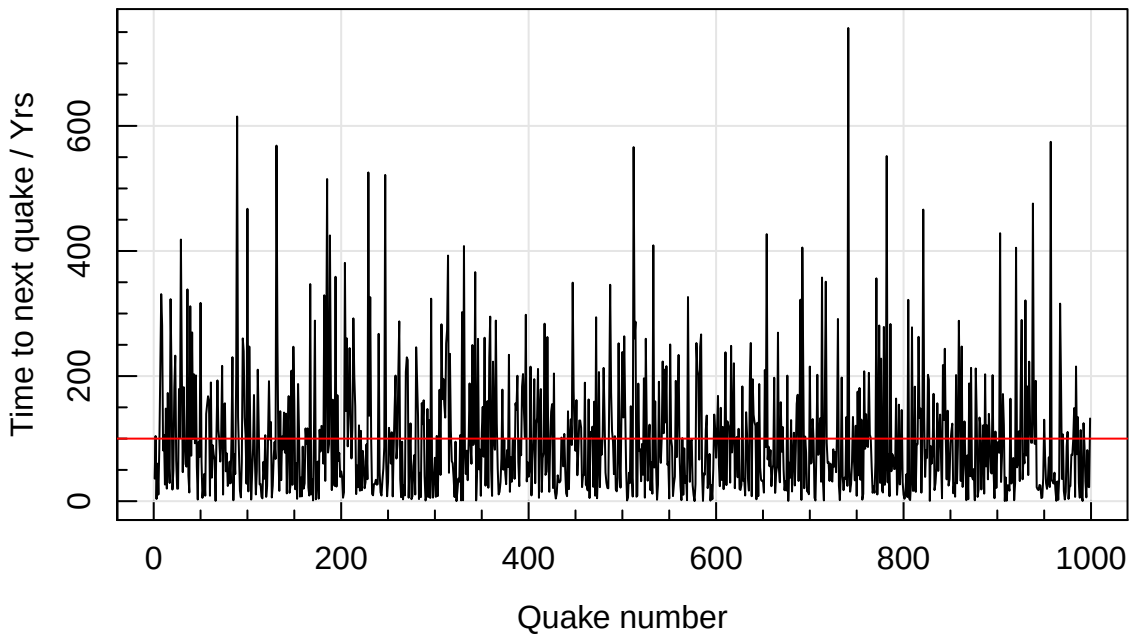
We can see the dates of each quake visually:

```
magplot(quakes, pch='.', xlab='Quake Number', ylab='Time / Yrs')
abline(0,100,col='red')
```



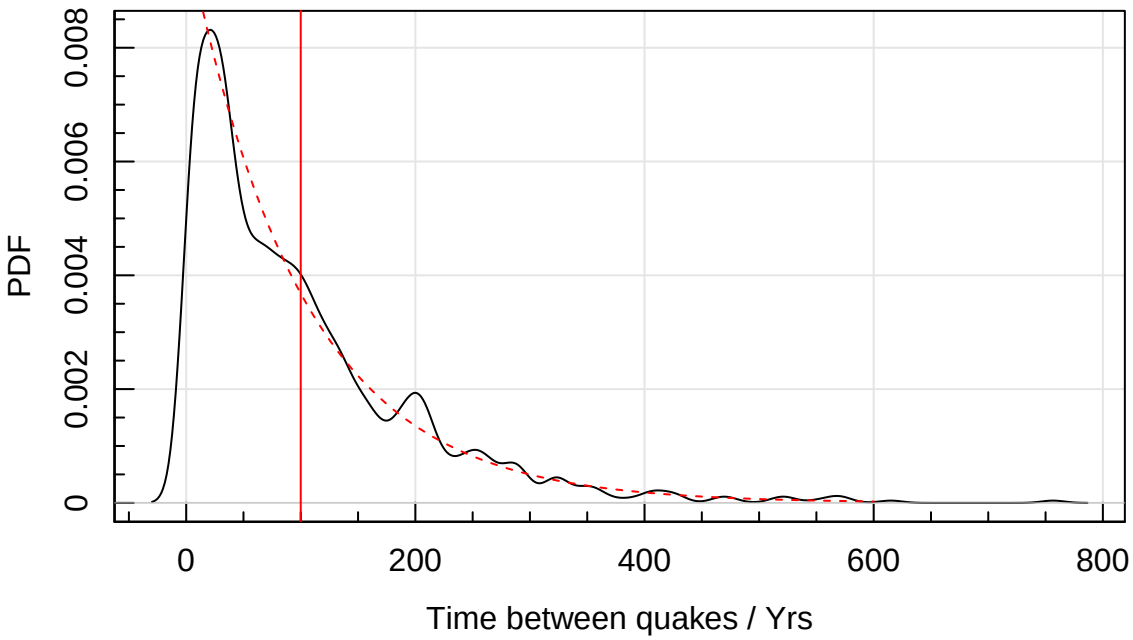
There are obviously departures from quakes occurring at precise 100 year intervals, which we should expect since this is still a random process. We can look at the time intervals between adjacent quakes:

```
magplot(diff(quakes), xlab='Quake number', ylab='Time to next quake / Yrs', type='l')
abline(h=100, col='red')
```



It is easy to compute the distribution of time intervals between events, and it is this statistic that follows the exponential distribution (with a 'rate' parameter defined as the number of events expected per unit time):

```
magplot(density(diff(quakes), bw=10), xlab='Time between quakes / Yrs', ylab='PDF')
abline(v=mean(diff(quakes)) ,col='red')
curve(dexp(x, rate=1/100), from=0, to=600, col='red', lty=2, add=TRUE)
```



The vertical mean line is clearly very close to 100, which is what we should expect by construction. The above exponential shape is also the answer to the question "why do you wait ages for a bus and then two come at once?", basically when events are random in time with a certain mean rate, they will be naturally more clustered to shorter intervals. In fact 63.2% ($1 - e^{-1} = 0.632$) of the time two buses will arrive before the mean wait time (and half will arrive within 70% of the mean wait time).

More formally we define the exponential distribution as:

$$p(n; \lambda) = \lambda e^{-\lambda n}$$

where n is our time between events, λ is the 'rate' used above.

Geometric Distribution

It is reasonable to wonder why this above is true for the exponential distribution. To intuit this better it is worth invoking the discrete analogue of the exponential distribution, namely the geometric distribution.

Taking the above situation, we can slightly recast the problem discretely by asking what is the chance that from a given time we observe a quake in the first year after one has just happened (assuming they truly are independent events, which in detail they will not because they are driven by complex and slightly correlated geological activity)? This will be very close to $\frac{1}{100}$. How about in year 2? Well, this will independently be $\frac{1}{100}$. And in year 3? Again, $\frac{1}{100}$. It is this property that makes the geometric (and by extension its continuous analogue the exponential distribution) 'memoryless'. It does not matter when you start the experiment, there is always the same chance that an event will occur in a given window of time.[1]

What does the above mean to the longer term delay distribution? Well it means the chance of observing a 1 year gap between quakes will be $\frac{1}{100}$, a 2 year gap will be $\frac{1}{100}\left(1 - \frac{1}{100}\right)$ (since it explicitly requires us not to have observed an event it year 1), a 3 year gap will be $\frac{1}{100}\left(1 - \frac{1}{100}\right)^2$ and an n year gap will be $\frac{1}{100}\left(1 - \frac{1}{100}\right)^n$. This is exactly the geometric distribution, with a discrete probability mass function (PMf, as opposed to PDF for continuous functions) given by:

$$P(n; p) = p(1 - p)^n,$$

[1] This is actually a vitally important concept in Earth quake safety analysis- whether they are truly memoryless events, or whether information about past quakes can inform us to the likelihood that one might occur soon. If Earthquake are the former (memoryless, with exponential time delay distributions) then there is no reason not to move to San Francisco today, but if they are non-exponential delay characteristics then you should keep your distance!

where p here is the chance of observing a quake in the window of time of interest, and n is how many multiples of this window we are interested in.

Now we can ask what continuous distribution could be integrated over window of n to give the above behaviour? It should be clear that the CDF form of the geometric can be given by:

$$F(n;p) = 1 - (1-p)^n.$$

In the limit of $p \to 0$ (i.e. rare events) $(1-p)^n = e^{-np}$ (test this yourself, but it is one of those handy convergence identities since $\ln(1-p) \sim -p$ for small p because the gradient near to $p = 0$ is 1 by the definition of $\ln(x)$ at $x = 1$, i.e. this is why Euler's number 'e' is special). Differentiating this and replacing p with λ (which has the same meaning) we get:

$$p(n;\lambda) = \lambda e^{-\lambda n},$$

I.e. we get back to our exponential distribution again.

Poisson Distribution

Closely related to the exponential distribution is the Poisson distribution. This asks a slightly different question of the same situation: what is the distribution of events expected in a certain amount of time. This has a PMF of the form:

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

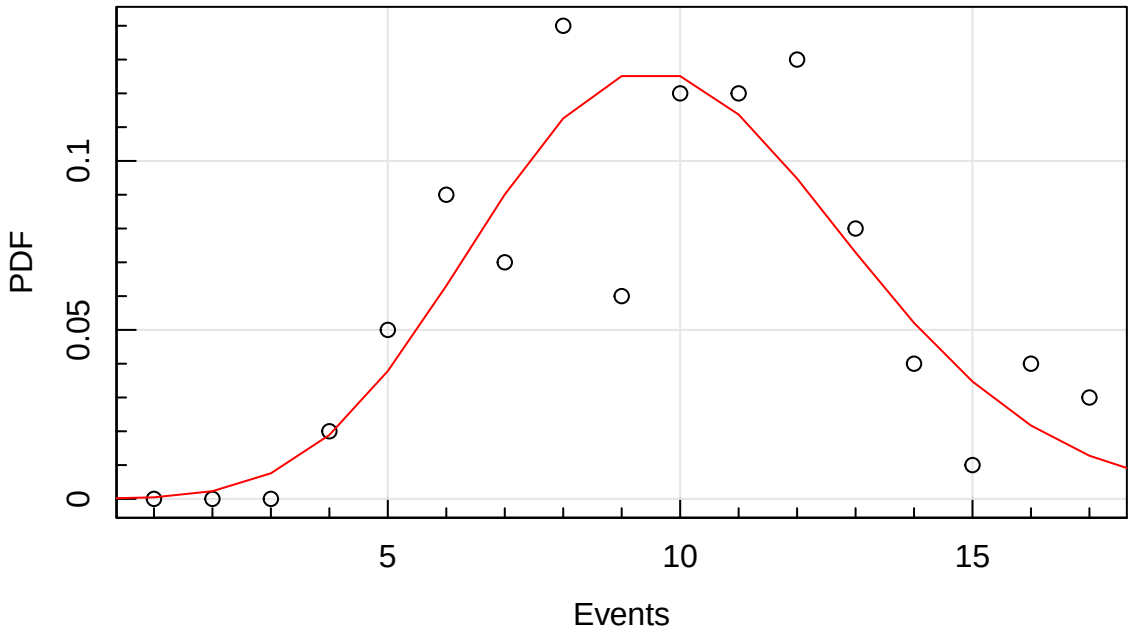
where k is the number of events in a target interval, and λ is the expectation of the number of events (not simply the rate, as for the exponential distribution above, since it depends how long we are observing for).

If we take the quakes data from above, we can see that we should expect on average 10 events every 1,000 years.

```
quakes_pois = foreach(i = seq(0,1e5-1e3,1e3), .combine='c')%do%{length(quakes[quakes>=i & quakes<=i+
```

We can now plot this and compare to the Poisson distribution with $\lambda = 10$ (which reflects the mean expected number of events):

```
magplot(1:length(tabulate(quakes_pois)), tabulate(quakes_pois)/1e2, xlab='Events', ylab='PDF')
lines(0:30, dpois(0:30, lambda=10), col='red')
```



The Poisson distribution is extremely important in astronomy, since it is more often the number of events in a fixed amount of time that various instruments (say UV and X-ray telescopes) are actually measuring. As it happens, many other effectively independent events also have Poisson distribution characteristics. Any property that is generated Uniformly will have Poisson distributions for set subsets, e.g. often the number of objects in an area is close to Uniformly random, hence the number in subsets will also have a Poisson distribution.

Gamma Distribution

The Gamma distribution asks a similar question to the Poisson statistic, but in the other direction. Rather than measuring how many events should occur in a fixed window of time, it describes the distribution of how long you would need to wait to observe a certain number of events given a certain event rate. It has a PDF form given by:

$$p(x;\alpha,\beta) = \frac{\beta^{\alpha} x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)},$$

where α is the number of events we wish to observe (known also as the shape parameter), and β is the rate (again, events per unit time of interest).

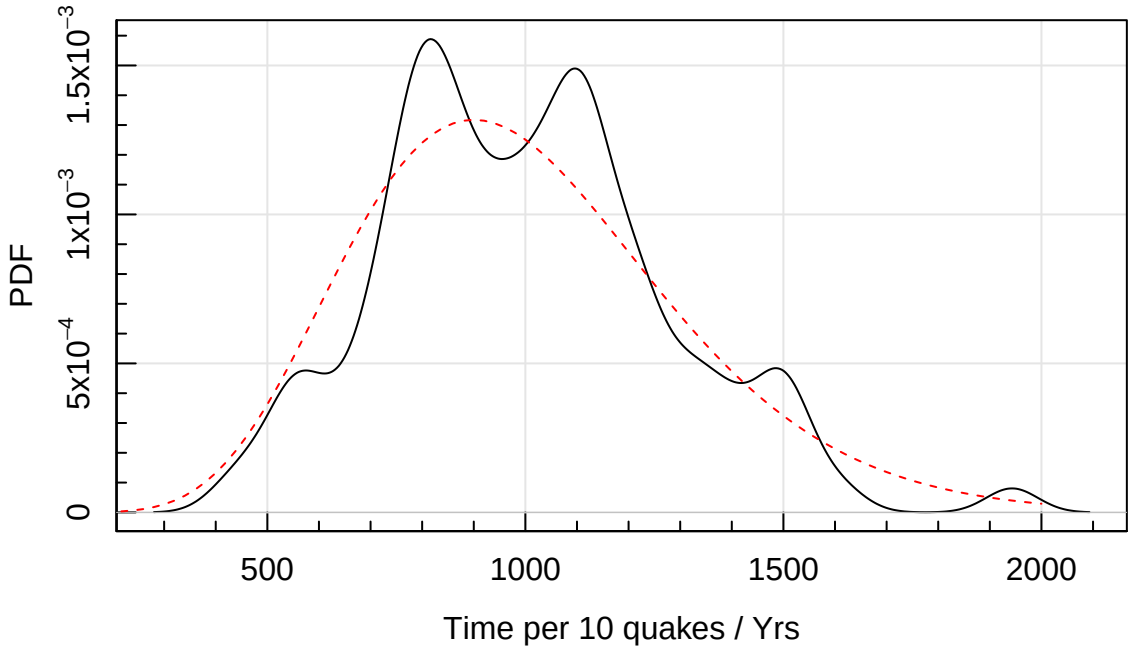
If we take the quakes data from above, we can see that we should expect 10 events to take on average 1,000 years.

We can extract the timing statistic that the Gamma distribution describes easily:

```
quakes_gamma = foreach(i = seq(1,1e3-10,10), .combine='c')%do%{diff(range(quakes[i:(i+10)]))}
```

We can now plot this and compare to the Gamma distribution with shape = 10 (which is α in the PDF definition above) and rate = 1/100 (which is β in the PDF definition above):

```
magplot(density(quakes_gamma, bw=50), xlab='Time per 10 quakes / Yrs', ylab='PDF')
curve(dgamma(x, shape=10, rate=1/100), from=0, to=2000, col='red', lty=2, add=TRUE)
```



The above examples should make it clear how close the relationship is between the Uniform, Normal, Poisson, exponential and Gamma distributions. All are very important, and the last 3 are part of the "exponential family" of distributions. In astronomy we make very regular use of the Poisson distribution in particular, since so many instruments integrate arrivals (e.g. of photons) over time. Also the Schechter function (which we will make use of later in the course) is in fact a re-parametrisation of the Gamma distribution.

Weibull Distribution

An important extension of the exponential distribution is the Weibull distribution. This basically allows a simple modification to purely random event histories where you either expect things to partially clustered (e.g. time to next sub 10s 100m run, since they tend to occur at infrequent competitive events) or have upper limits on lifetime (time to car breaking down- no car will keep working for 100 years, but it may be truly random for the next 10 years). It can be defined as

$$p(n;\lambda,k) = \frac{k}{\lambda}\left(\frac{n}{\lambda}\right)^{k-1} e^{-(n/\lambda)^k},$$

where k is the shape parameter, and λ is the scale parameter (which is the inverse of the rate, so confusingly the inverse of the meaning of λ in the exponential distribution case). If $k = 1$ then the Weibull distribution reduces to the exponential, if $k < 1$ then events are more likely to be closely associated with each other (earlier failure), and if $k > 1$ then they are more likely to be distant (ageing failure). The Weibull distribution functions can be accessed in base **R** via **dweibull** etc.

Exponentiated Weibull Distribution

There are extensions to the Weibull distribution too, e.g. the exponentiated Weibull distribution allows two shape terms that control the early and late time behaviour, e.g. you can have a model where (e.g.) magnetic hard drives will typically either break very early due to manufacturing defects or last until a certain age when components physically wear out. This turns out to be a very useful model for insurance purposes, because things like computers, motors and hard drives tend to either fail very early, or last pretty consistently to old age, e.g. if you buy a new hard disk it will either fail in the first week or after 3 years (the warranty period of course!) with not much in the middle.

In practical terms this often mean there is little difference between one and three year warranties (they all pick up things failing very early) but extended warranties can be genuinely useful. This type of failure

mode is often called the bathtub model: starts high, drops to constant baseline and then shoots back up again.

Generalised Gamma Distribution

One of the most powerful supersets of exponential distributions is the generalised Gamma distribution. We will not discuss it in detail here, but it contains the log-Normal, Weibull, exponential and Gamma distributions for certain combinations of parameters. In practice it is often used to help select which distribution from the exponential family should be used, since it is famously hard to fit directly (the parameters are highly degenerate unless care is taken with the parametrisation used).

Pareto (Power-Law)

The Pareto (often called power-law) distribution is a common distribution in the astronomy literature. It was originally developed to model wealth inequality, given origin to the famous 80-20 (or 'Pareto Law' of life)- 80% of a company's profit comes from 20% of its customers etc. It can be specified as:

$$p(x) = \frac{\alpha x_m^\alpha}{x^{\alpha+1}}, \text{for } x \geq x_m, 0 \text{ otherwise.}$$

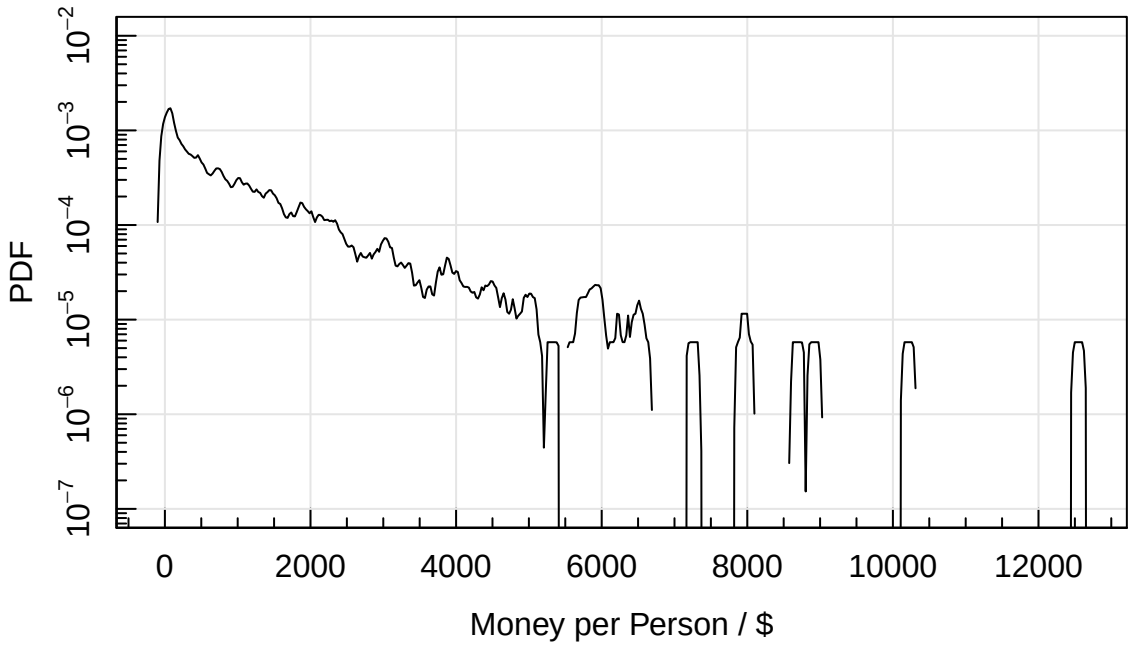
It is surprising how many processes naturally tend towards a power law distribution (see for instance Benford's Law, see below). For instance, there is a simple wealth inequality simulation that we can run. We start with 1,000 people each of whom have $1,000. We then choose two people and take a random fraction of one person's money and give it to another, and repeat this 100,000 times.

```
people = rep(1000,1000)
for(i in 1:1e5){
  select = sample(1000,2) #select two people
  exchange = runif(1,0,people[select[1]]) #compute a Uniformly random fraction of their money
  people[select[1]] = people[select[1]] - exchange #one person loses this
  people[select[2]] = people[select[2]] + exchange #the other gains
}
```

The above produces a power-law like distribution with a skew to lots of people with little money:

```
magplot(density(people, kern='rect', bw=50), xlab='Money per Person / $', ylab='PDF',
        log='y', ylim=c(1e-7,1e-2))
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 80 y values <= 0 omitted from
## logarithmic plot
```



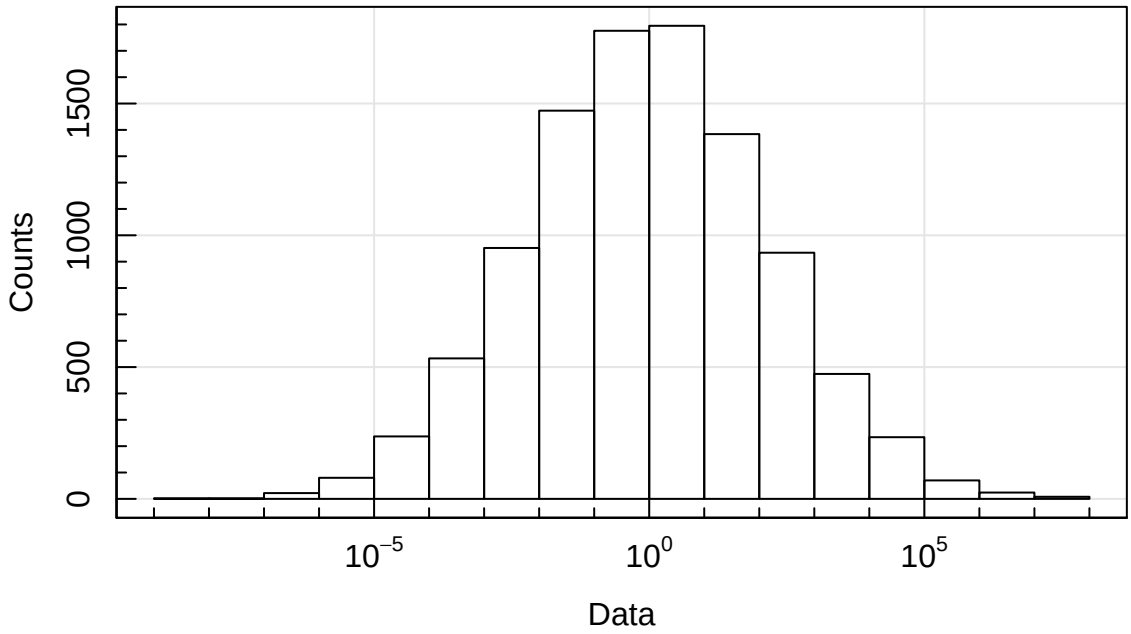
This is why Pareto was interested in it. It turns out it is very hard to produce an 'equal' society without fairly aggressive intervention. Why does this happen? Well, imagine you have a poverty threshold of $2,000 and you are rich (say you currently have $8,000). Most of your interaction will be with poorer people and there is a good chance you will lose a lot of money quickly, and not many opportunities to build it up (you can only take a fraction of what the poorer person has after all). So in noisy financial interactions with no skill involved (the average market speculator, basically), a few people will blindly get rich. They will probably sell popular books about their astute financial acumen too.

Benford's Law

Generate some data that naturally spans many orders of magnitude:

```
data = exp(rnorm(1e4,sd=5))
maghist(data, xlab='Data', ylab='Counts', log='x')
```

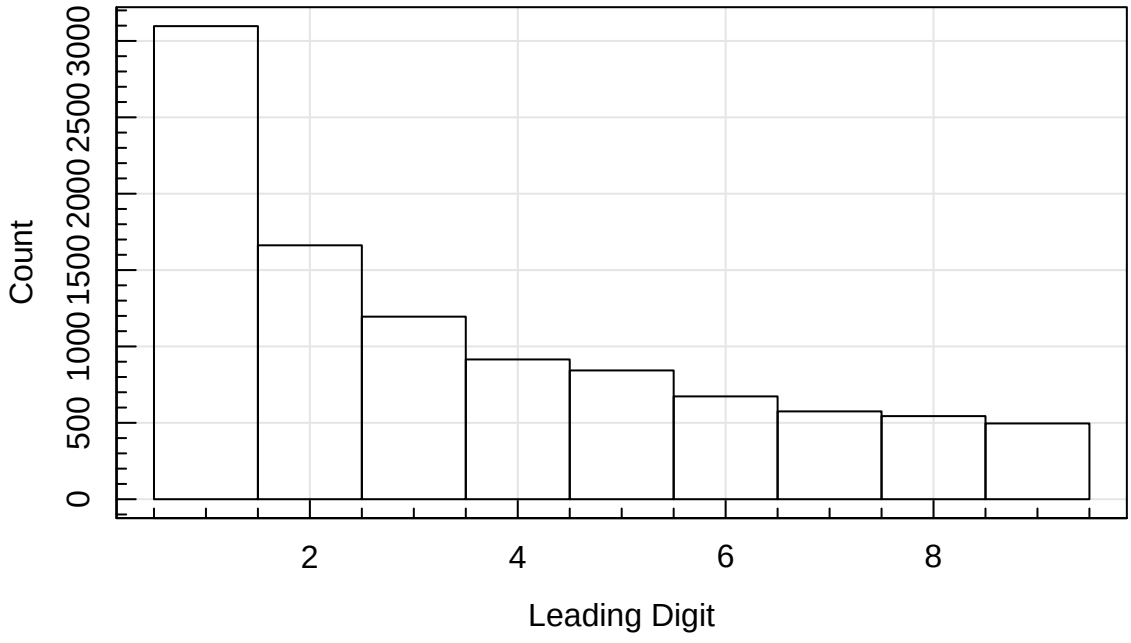
```
## Summary of used sample:

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -8.74810 -1.51687 -0.03802 -0.03920  1.42866  7.97909

## Pop Std Dev: 2.175

## MAD: 2.184

## Half 16-84 Quan (1s): 2.1613

## Half 02-98 Quan (2s): 4.3394

## Using 10000 out of 10000
```



What's your expectation of the distribution of the leading significant digits of these numbers? Most people intuit that there should be an equal chance of each digit 1-9 appearing equally often, but this is what we find:

```
leading_digit = as.integer(substr(formatC(data, format='e'), 1, 1))
maghist(leading_digit, xlab='Leading Digit', ylab='Count', breaks=seq(0.5,9.5))
```

```
## Summary of used sample:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   3.000   3.476   5.000   9.000
## Pop Std Dev: 2.5058
## MAD: 2.9652
## Half 16-84 Quan (1s): 3
## Half 02-98 Quan (2s): 4
## Using 10000 out of 10000 (100%) data points (0 < xlo & 0 > xhi)
```



The leading digit 1 appears much more often!

```
table(leading_digit)/1e4
```

```
## leading_digit
##      1      2      3      4      5      6      7      8      9
## 0.3097 0.1662 0.1195 0.0915 0.0843 0.0673 0.0575 0.0544 0.0496
```

These numbers are in fact logarithmically distributed, i.e. in linear space:

```
total = integrate(function(x){1/x},1,10)$value
for(i in 1:9){
  print(integrate(function(x){1/x},i,i+1)$value/total)
}
```

```
## [1] 0.30103
## [1] 0.1760913
## [1] 0.1249387
## [1] 0.09691001
## [1] 0.07918125
## [1] 0.06694679
## [1] 0.05799195
## [1] 0.05115252
## [1] 0.04575749
```

Which is uniform intergral limits in log space:

```
total = log(10)
for(i in 1:9){
  print((log(i+1) - log(i))/total)
}
```

```
## [1] 0.30103
## [1] 0.1760913
## [1] 0.1249387
## [1] 0.09691001
## [1] 0.07918125
## [1] 0.06694679
## [1] 0.05799195
## [1] 0.05115252
## [1] 0.04575749
```

So what's going on? Well, the above makes sense when we think about numbers in terms of their dynamic range. The relative range between 1 and 2 is clearly a factor of 2, but the relative range between (e.g.) 5 and 6 is only 20%. This means a small relative value change is very likely to change the leading digit when our current leading digit is large (say, 9) and quite unlikely to change the leading digit when the current leading digit is small (say, 1). This reality if known as Benford's Law, and it is actually used to detect for 'realistic' number in the real work (tax fraud etc).

It only work when the native dynamic range of the random data is high and ideally natively power law distributed (which many natural phenomena are). So it works well for the stellar masses of galaxies (since they span so many orders of magnitude) but poorly when the native range is small like the heights of adults (the dynamic range is only a factor of couple at the extreme).

Binomial and Multinomial Distribution

The final types of distributions that are of some use in astronomy that we will discuss here are the binomial and multinomial.

Binomial distribution

We saw the binomial earlier for calculating the likely outcomes for a fixed probability event (e.g. chance of tossing a head etc with biased or unbiased coin). The PMF form can be be written as

$$P(k; n, p) = \frac{p^k(1-p)^{n-k}n!}{k!(n-k)!},$$

where k is the number of successes (e.g. chance of getting 2 heads), n is the number of trials (e.g. we toss the coin 3 times) and p is chance of success (e.g. the coin is biased and we get a head 60% of the time). For the coin example the probability of two heads would therefore be:

```
dbinom(2, size=3, prob=0.6)
```

```
## [1] 0.432
```

Since we have already played with the binomial distribution earlier in the course, we will leave the discussion there.

Multinomial distribution

The logical extension to the binomial distribution is the multinomial distribution, which can deal with more than one type of outcome (so events like rolling dice). We will not parametrise the PDF here, but we can see how it works with a simple example. To simulate 10 rolls of two dice (say in a two dice game like Monopoly) we would use the built in **rmultinom** function and run:

```
Monop_10 = rmultinom(n=10, size=2, prob=rep(1/6,6))
Monop_10
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    1    0    0    0    0    1    0    1    1     0
## [2,]    0    0    1    1    0    0    0    0    0     0
## [3,]    0    0    1    0    1    1    1    0    0     0
## [4,]    0    1    0    0    0    0    1    0    1     2
## [5,]    0    0    0    0    1    0    0    1    0     0
## [6,]    1    1    0    1    0    0    0    0    0     0
```

Here n is the number of turns, *size* is the number of dice we want to roll each turn, the *prob* is the vector of outcomes (here unbiased dice with an equal 1/6 chance of all 6 outcomes, but note the *prob* is internally normalised to 1, so we do not actually need to worry about the sum). What we get in return is a *size* $\times$ n matrix of outcomes, where the row number indicates the number thrown. The numbers in the matrix then indicate how often each number appears, so columns all sum to 2, and sometimes we might see two rolls produce the same number.

We can easily do a bigger simulation of rolling 5 dice (like in Yahtzee):

```
Yahtzee = rmultinom(n=1e6, size=5, prob=rep(1,6))
```

And then check the chance of all 5 outcomes being the same (so getting Yahtzee on one roll):

```
length(which(Yahtzee==5))
```

```
## [1] 781
```

Given we played one million times, this means the chance of a single roll Yahtzee is $\sim 800/10^6 = 8 \times 10^{-4}$, so 0.08%.

We can check this using the **dmultinom** function (we check for one type of Yahtzee, in this case 0,0,0,0,0, and then times by 6):

```
dmultinom(c(5,0,0,0,0,0), size=5, prob=rep(1,6))*6*100
```

```
## [1] 0.07716049
```

Similarly, we can check for full houses (3 and 2 of a kind) by checking for a particular Full House (1s full of 2s) and multiplying by 30 to get all combinations (5 pairs for each 3-of-a-kind, and 6 possible 3-of-a-kind):

```
dmultinom(c(3,2,0,0,0,0), size=5, prob=rep(1,6))*30*100
```

```
## [1] 3.858025
```

So there is a bit less than a 4% chance of directly rolling a Full House.

The keen student should try to calculate the chance of rolling a Yahtzee after three turns, where in each turn you re-roll as many dice as you like (in practice you would leave alone the most frequent dice value, and roll the rest).