Interactive demonstrations for statistics

The file "interact_fun.R" contains a number of R functions which demonstrate statistical concepts, allowing you to change the parameters to see what happens graphically. On Windows systems, a pull down menu allows you to select the demonstration you want.

Some of these functions require additional packages:

- tcltk
- scatterplot3d

You can download these from the internet using the 'Packages > Install package(s)…' menu item in R.

To get started with the demos, download the file "interact_fun.R" to a convenient place on your computer (eg the desktop), then drag the icon to the R console and drop it. Alternatively, you can use 'File > Source R code…' and select 'interact_fun.R'.

All the functions generate simulated data when you start them with no parameters: eg. interact.power(). Some of them you can use to display and experiment with your own data, as described below.

Most of the functions use a 'TK widget' with sliders and checkboxes to control the graphics, while a few operate with mouse-clicks on the graphic itself.

Interactive histogram

Select 'Interact > Histogram' from the pull-down menu, or type `interact.hist()` at the `>` prompt in the console. A TK widget and a histogram will appear, using simulated data which are *not* normally distributed (they are skewed or possibly bimodal).

At the bottom of the plot, below the histogram, is a 'rug' of tickmarks corresponding to the actual data points. The checkboxes enable you to display or hide:

- a kernel density estimate (red curve): effectively a smoothed histogram of the data;
- a normal density function with the same mean and standard deviation as the data set.

You can play with the number of bins and the upper and lower limits of the histogram to see the effects. (I was surprised at the large effect of changing the upper and lower limits, which are usually just set at some convenient round number without a moments reflection!)

Using real data: Your data points need to be in a vector (not a frequency table). You can then type `interact.hist(mydata)` at the `>` prompt. Try it with the built-in dataset 'faithful', where 'faithful$eruptions' gives the length in minutes of eruptions of the geyser Old Faithful at Yellowstone National Park; use `interact.hist(faithfull$eruptions)`. For more details of the dataset use `?faithful`.

The result will be saved in the object 'interact.hist.out', which is of the same class as output from R's usual 'hist' function. Use `plot(interact.hist.out)` to see a 'normal' R histogram using the number of bins and the upper and lower limits you selected.

Law of Large Numbers

One advantage of large samples is that the sample mean is close to the population mean, which is reflected in narrow confidence intervals.

Select 'Interact > Law of Large Numbers' from the pull-down menu, or type `interact.lln()` at the `>` prompt in the console. A TK widget and a graphic will appear, using simulated data from a normal distribution with mean 0 and s.d. 1.

The black curve shows the sample mean, starting at the left with a sample of n = 1 drawn randomly from the population and successively adding more items to the sample up to a maximum of n = 50. Click on the 'New sample' button to draw a new set of samples from the population. The sample mean is often a long way from the population mean (dashed red line) for small samples, but pretty close for large samples.

The solid blue lines show the confidence interval calculated from the sample s.d. from n = 2 upwards. This is generally very wide for small samples. You will also see that it does not always include the population mean. You can adjust the confidence interval from 50% to 99% with the slider.

The Probability Interval (dashed green lines) is similar to the confidence interval but is calculated from the known s.d. of the population, not from the sample s.d. The Confidence Interval slider also controls the Probability Interval. For a 95% PI, 95% of all possible sample mean fall within the Probability Interval.

Using real data: If you have a sufficiently large data set (say >50), you can use this as the 'population', and see the effect of drawing smaller sub-samples. Type `interact.lln(mydata)` at the `>` prompt. The dashed red line now represents the mean of the whole sample. The Probability Interval can't be calculated, as we do not know the population mean and s.d. You can try is with the built-in dataset chickwts; use `interact.lln(chickwts$weight)`. Use `?chickwts` to see details of the dataset.

Central Limit Theorem

A second advantage of large samples is that – according to the Central Limit Theorem – the distribution of the sample means will be nearly normal even if the distribution of the population is very far from normal.

Select 'Interact > Central Limit Theorem' from the pull-down menu, or type `interact.clt()` at the `>` prompt in the console. A TK widget and a graphic will appear, showing four pink histograms. These use simulated data from a normal distribution, an exponential distribution, a uniform distribution and a beta distribution. The second and third and especially the last are very different from normal distributions. The black curve shows a normal distribution with the same mean and s.d. as the population.

The graphs change when you increase the sample size above 1 with the slider. (The means of samples of n = 1 have exactly the same distribution as the population itself.) The blue histogram shows the distribution of the means of a large number of samples, and the black curve now represents a normal distribution with the same mean and s.d. as the sample means.

Even with a sample size of n = 2, you can see that:

- the sample means for the uniform distribution are already normally distributed;
- the sample means for the exponential and beta distributions are still a long way from normal but moving in the right direction;
- the distribution of sample means for the normally distributed population is normal, but sample means are clustering closer to the population mean.

With samples of n = 20, even the sample means for the beta distribution are close to a normal distribution, and all are clustered quite close to the mean of the population. (If you find the pink histogram of the population in the background distracting, hide it by unchecking the "Show Population Distribution" checkbox.)

Many statistical methods only work if the sample means are normally distributed, for example, the calculation of confidence intervals from the sample s.d. This condition is met if the samples are big enough.

You can't use real data with this function. It would have to be completely rewritten to do that.

Confidence intervals

Suppose we weighted a sample of squirrels and inferred that the mean weight was 94g with a 95% confidence interval of 70g to 118g. A common *mis*interpretation of this is: "We can be 95% confident that the true population mean is between 70g and 118g." Some statisticians[1] get very excited about this mistake, as the true interpretation is different and more subtle.

Select 'Interact > Confidence Intervals' from the pull-down menu, or type `interact.confint()` at the **>** prompt in the console. A TK widget and a graphic will appear, using simulated data drawn from a normally distributed population with mean 100 and s.d. 10.

Each of the 100 horizontal lines corresponds to a sample of n = 25 drawn from this population. The vertical tick in the middle of the line is the sample mean and the ends of the line correspond to the confidence interval calculated from the sample s.d. The line is black if the confidence interval includes the true mean (which we know is 100), otherwise it is red or blue.

The real interpretation of the 95% confidence interval is that, if we take a very large number of samples, 95% of the lines on the graph will be black, ie they include the true mean. Press the "New samples" button and you will see that this is approximately right. (You'll rarely get exactly 5 coloured lines, any more than you'd expect to get exactly 5 heads when you toss a coin 10 times.

You can use the sliders to experiment with different values for the confidence interval and different sample sizes.

You can't use real data for this, as it requires huge numbers of samples from a population with known mean and s.d.

The Power of a Hypothesis Test

When we conduct a hypothesis test, we may accept or reject the null hypothesis, which may or may not be true. There are four possible situations:

	Null hypothesis true	Null hypothesis false
Null hypothesis rejected	Type I error	Correct decision
Null hypothesis accepted	Correct decision	Type II error

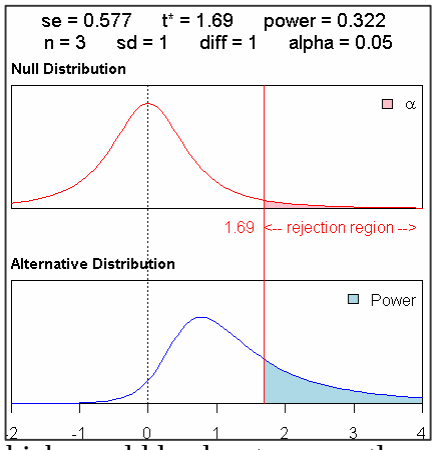
We are usually very careful to avoid Type I errors: we don't want to mistakenly infer that, say, a specific medication cures a disease or that birds with larger territories raise more offspring, too often. We set significance levels at 5% or lower to limit the probability of making such errors. But sometimes Type II errors can be important, as when we infer incorrectly that a pollutant has no effect on fish mortality or that amnesic patients' scores on a particular memory test are no different from normal

[1] Only some. If you approach the analysis from a Bayesian perspective, you can calculate a 'credibility interval'. With an uninformative prior, the credibility interval has (with rare exceptions) the *same* limits as the confidence interval. And the correct interpretation of the credibility interval is: "We can be 95% sure that the true population mean is between 70g and 118g."

people. We will look at an example where we use a one-sided t-test to test the null hypothesis that the difference in means is zero.

Select 'Interact > Power' from the pull-down menu, or type `interact.power()` at the `>` prompt in the console. A TK widget and a graphic with two plots will appear. The example shown is

The upper plot shows the distribution of the sample means based on the null hypothesis that the difference in means is 0 and the population s.d. is 1 (which you can change with the slider). The curve is the t-distribution, and the shape depends on the population s.d. and the sample size. The area of the pink section under the curve corresponds to the significance level, alpha, which we choose: typically 5%, sometimes 1% or 0.1%. The vertical red line and the red figures indicate the critical value; we will reject the null hypothesis if our sample mean is greater than this value.



The lower plot represents the alternative distribution. Note that we cannot discuss Type II errors without deciding what size of effect we are looking for. There will always be a difference between two samples, even if it is only a few nanometres or milligrams, so in that trivial sense the null hypothesis is always false and Type II errors are impossible! When you start the power demo, the effect size (or true difference according to our alternative hypothesis) is set to 1 and the lower curve is a non-central t-distribution with a mean of 1. Notice that most of the area of the curve lies to the left of the vertical red line indicating the critical value. That means that, even if the true difference is 1, most of the samples we could get would lead us to accept the null hypothesis – and make a Type II error. The proportion of samples which would lead us to correctly reject the null hypothesis, represented by the blue area in the lower plot, is the power of the study. And in this case, it's pretty low, just 0.322 or 32%.

The power depends on the sample size, the population s.d., the true difference (effect size) you want to detect, and the level of significance, alpha. You can try changing these with the sliders to see how they affect power.

Unless the power of the study is high, you run a large risk of a Type II error if you assert that the null hypothesis is true. It is important to distinguish between lack of evidence for a difference (ie the null hypothesis *might be* true) and evidence of no difference (ie the null hypothesis *is* true). If you can show high power for a given effect size, then you can make a statement such as, "the difference is less than X." Otherwise the inference should be restricted to "no evidence of a difference".

Using real data: I tried to adapt this function to use with real data, but it uses the non-central t-distribution, which produces huge numbers of warnings in R 2.3.0. That doesn't matter in a demonstration like this, but would a problem if it gave wrong results with real data. Maybe I'll look at that again when the bugs are sorted out. In the meantime, R has it's own power.t.test function: use `?power.t.test` to see details.

Correlation and correlation coefficients

The correlation between two random variables is a measure of the degree to which high values of one are associated with high values of the other. Note that this does not presuppose that one variable is dependent on the other, or that one can be used to predict values of the other.

Select 'Interact > Correlation' from the pull-down menu, or type `interact.correlation()` at the `>` prompt in the console. A TK widget and a scatterplot will appear, using simulated data drawn from two normally distributed populations with mean 0 and s.d. 1 and 0 correlation. The statistics shown are Pearson's product-moment correlation coefficient, r, and the coefficient of determination, r^2.

You can move either of the sliders to see effects of the changing r or r^2 on the scatterplot.

Using real data: You cannot use real data with this function, as it relies on wholesale changes in the two variables. You can use the next function, regression, with real data, as it also calculates r and r^2. However, uncheck the 'LS Line' checkbox: the LS line is a regression line of *y* on *x*, which is meaningful only if *x* is the predictor and *y* the response variable; this would not be the case, for example, if *x* and *y* were the length and width of turtle shells.

Regression

Regression is appropriate where there is a 'one-way relationship' between two variables, so that one (the predictor variable, *x*) might cause changes in the other (the response, *y*), but the response cannot affect the predictor. For example, if the abundance of a species varies with elevation, we might suspect that elevation (the predictor) is the cause of the change in abundance (the response); it is not plausible to imagine that the abundance of an organism affects the elevation.

Select 'Interact > Regression' from the pull-down menu, or type `interact.regression()` at the `>` prompt in the console. A blank plot with *x* and *y* axes will appear on the left of the graphics window, and a number of controls on the right.

Click in the blank plot area to add points. Once you have created two points, a least-squares regression line will be fitted through the points, and the slope and intercept will appear at the top of the plot. Also at the top is r^2, the coefficient of determination, which is the proportion of the variation in *y* which is explained by the regression on *x*. (r is also displayed, but is not very meaningful for a regression.)

Experiment with adding, deleting and moving points to see the effect of outliers or the effect of having points more or less closely clustered around the line.

Click on the 'Exit' box to finish. The coordinates of the points you created are returned invisibly, so you can assign them to an object (you can also get them back with `.Last.value` if you forgot to assign the output to an object before running interact.regression).

Using real data: Type `interact.regression(myx, myy)` at the `>` prompt, where 'myx' is the predictor variable and 'myy' the response. Try the built-in dataset 'cars', which gives data for the distance needed to stop and the speed of the car when the brakes were applied: clearly speed is the predictor variable and distance to stop is the response. Use `interact.regression(cars$speed, cars$dist)`and check `?cars` to see details of the dataset. When you use real data like this, you will see green crosses indicating the positions of the original points; these do not affect the slope, intercept or r^2 calculations, which are based only on the black circles. Also note that you can add points outside the edges of the plot by clicking in the margin; the plot will then be resized to include the new point.

Principle Components Analysis (PCA)

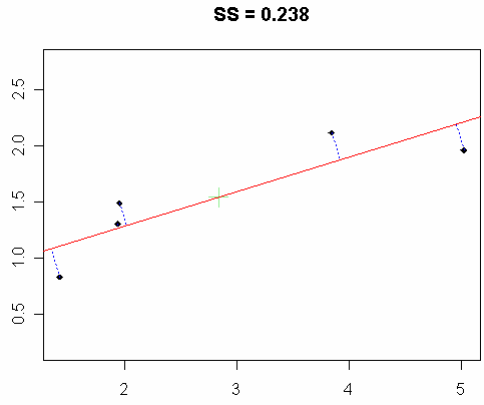
Often you measure a large number of variables which are correlated with each other, but with no clear predictor/response relationships. For example, you may have a data for habitat variables, such as elevation, aspect, light levels, soil pH and depth, …, or you may have measurements of animal specimens: head-and-body length, tail length, interorbital distance, length of hind foot, etc. PCA enables you to reduce the number of variables by combining those which are closely related.

Although PCA is useful when there are many variables, this demonstration will use just two[2], and we'll also keep to a small number of points.

Select 'Interact > PCA' from the pull-down menu, or type `interact.PCA()` at the `>` prompt in the console. A scatterplot appears with five points whose coordinates have be generated randomly from normal distributions. The green "+" indicates the mean of the *x* and *y* values for the five points and is the "centroid" of the cluster of points.

[2] I tried a version with three dimensions using 3-D graphics, but it wasn't clear what was happening, so we'll stick to 2-D.

The first stage is to find the best position for the first principle axis. Click anywhere in the figure and a red line will appear passing through the point you clicked and the centroid (+). Dotted blue lines will also appear joining each point to the red axis; note that the blue lines are perpendicular to the red axis, they are not vertical, as they would be for a regression analysis. At the top of the screen you will see "SS = …": this is the sum of the squares of the lengths of the dotted blue lines. Click again in the figure and the red axis will move, and the dotted blue lines and the value for SS will change too. Your task now is to keep clicking until you find the position for the red axis which is 'closest' to all the points as measured by SS, that means minimizing SS. The diagram shows my best attempt for one arrangement of the 5 points. When you have got the minimum value for SS, right-click in the graphics window and select 'Stop'.



The first principle axis is now fixed (and has turned blue), and open blue circles on the axis indicate the projection of each point, ie. where the blue dotted lines intersect. Click next to these open circles to see the score for each point on the 1st principle axis: these scores measure how far the projected point is from the centroid, positive on one side of the centroid, negative on the other. When you've done, right click and select 'Stop'.

Now we turn to the 2nd principle axis. All the axes pass through the centroid and all must be orthogonal (ie at right angles to each other), so we have no choice about the position of the last axis. You can probably visualize working in 3 dimensions, with 3 variables: you would position the 1st axis by minimizing the SS distance in the same way, then you would be able to choose the position of the 2nd axis, rotating it around the first. We'd then have no choice about the position of the 3rd axis.

The 2nd principle axis is plotted automatically, through the centroid and at right angles to the first. We positioned the 1st axis as close as possible to the points, and the 2nd is much further away: the value for SS is much higher than for the 1st axis. Click near the open red circles to show the scores for each point on the 2nd axis.

That's it. The original coordinates of the points together with the new scores on the principle axes will appear in the console.

Using real data: You can put in your own x and y values, and also a character vector to indicate which group the points belong to. You can try it with R's built in 'iris' dataset, which has 50 measurement for each of 3 species of iris:

```
summary(iris)
attach(iris)
sps <- rep(c("s","c","g"), each=50)
res <- interact.PCA(Sepal.Length, Sepal.Width, sps)
```

Maximum Likelihood Estimators

The concept of likelihood is basic to a whole range of statistical theory. In fact it is so basic – so deeply embedded below the statistical methods we use every day – that it is rarely mentioned in conventional statistics courses. It comes into the open when we try to fit models to our data and need to estimate model parameters. Two examples illustrate the concept of the maximum likelihood estimator (MLE). We'll start with a Poisson model.

Fitting a Poisson distribution:

Select 'Interact > Max Likelihood Est > Poisson' from the pull-down menu, or type `interact.mle.poiss()` at the `>` prompt in the console. A TK widget and graphics window with two plots will appear, using simulated data. In the upper plot, the 10 blue points along the bottom of the graph might represent the number of car accidents on a busy highway for each of 10 weeks; they

are of course exact whole numbers, but they are spread out in the graph so that you can see them. The Poisson model says that the probability of getting *x* accidents in one week is:

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

(You can use 'Interact > Distributions > Poisson' to see what this distribution looks like.) Each of the blue lines in the bar graph represent the probability of getting that number of accidents in a week. The likelihood of getting that particular set of 10 numbers in 10 weeks for a particular value of is found by multiplying the 10 individual probabilities together. Since they are all less than 1 – typically less than 0.3 – the answer will be quite small. For such small numbers, it's easier to use the log rather than the number itself, and the log likelihood is calculated and displayed at the top.

Your job now is to adjust the slider to find the value of λ which gives the maximum likelihood. (Remember that for negative numbers -2 is *bigger* than -3.) The lower plot in the graphics window is a graph of likelihood versus λ.

The value of λ which maximizes the likelihood of getting the actual data observed is called the 'maximum likelihood estimate' or 'MLE' of λ.

Once you are sure that you've found the MLE for λ, try this: add up the ten numbers and divide by 10; that is, find the mean number of accidents per week. You should get the same answer: taking the mean of the observations is a quick way of finding the MLE for λ for a Poisson model. You probably knew that already, and now you know *why* the mean of the data is the right value for λ.

Using real data: Type `interact.mle.poiss(mydata)` at the `>` prompt, where 'mydata' is a vector containing your data. During surveys of Batang Ai NP in Sarawak, we surveyed 10 sites and recorded the following numbers of barking deer: 0, 0, 1, 1, 2, 2, 4, 4, 4, 5. You can put these data in like this:

```
deer <- c(0, 0, 1, 1, 2, 2, 4, 4, 4, 5)
interact.mle.poiss(deer)
```

Fitting a Normal (Gaussian) distribution:

Many biological measures (at least for birds and mammals) fit a bell-shaped distribution described by the Gaussian equation:

$$P(X) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{X-\mu}{\sigma}\right)^2}$$

This distribution turns up so frequently[3] that it is considered "normal". To fit the normal curve to our data, we need to adjust two parameters, μ (mu) and s (sigma).

Select 'Interact > Max Likelihood Est > Normal' from the pull-down menu, or type `interact.mle.norm()` at the `>` prompt in the console. A TK widget and graphics window with three plots will appear, using simulated data. In the upper plot, the 10 black points along the bottom of the graph might represent a measure such at the lengths of the tails of 10 randomly selected adult male sparrows.

Now adjust the two sliders to find the values of μ and s which give the maximum likelihood. The lower plots in the graphics window are graphs of likelihood versus μ and s. When you've found the MLEs for μ and s, look at the Console: R has calculated the mean and s.d. of the ten numbers and these should be the same as your values for the MLEs.

[3] For another reason to call this 'normal', look back at the Central Limit Theory. The normal curve derives from our convention of calculating means by *adding up* the items: the *arithmetic* mean of 2, 3 and 4 is (2 + 3 + 4)/3 = 3. Pythons might find it obvious that the mean of 2, 3 and 4 is 2.88, since $\sqrt[3]{}$(2 x 3 x 4) = 2.88 (the *geometric* mean), and the pythons' "normal" curve would be our log-normal curve.

Likelihood and MLEs do not get more emphasis in basic statistics courses because quick shortcuts produce the MLEs for simple models. And the reason they are important for more advanced statistics is that quick shortcuts don't exist for sophisticated models.

Using real data: Type `interact.mle.norm(mydata)` at the > prompt, where 'mydata' is a vector containing your data. During a study of flying foxes in Sarawak, 10 fully-grown bats were caught and their wingspan measured in cm. You can put these data in like this:

```
bats <- c(98, 113, 119, 112, 115.5, 126, 105, 120, 124, 124.5)
interact.mle.norm(bats)
```

The relationship between Probability and Likelihood

In the previous section we glossed over the difference between 'probability' and 'likelihood'. You may have noticed though that we used probability when talking about individual data points and likelihood for the full data set. This demonstration shows the difference.

Select 'Interact > Probability vs Likelihood' from the pull-down menu, or type `interact.pbllh()` at the > prompt in the console. A dialogue box opens asking how many trials you want to use: leave the number at 10 and press OK. A TK widget and a graphics window appear.

Begin with all the "Show…" check-boxes unchecked. You will see a bar graph in the lower part of the graphics window.

- For a fair coin, the probability of a head in a single throw is $p = 0.5$. Move the p slider to 0.5 and the bar graph shows the probability of getting 0, 1, 2,…10 heads if you throw 10 coins.
- For a fair die, the probability of throwing a six is $p = 1/6$ or 0.167. Move the p slider to 0.16 and the bar graph shows the probability of getting 0, 1, 2,…10 sixes if you throw 10 dice.
- Suppose the probability of recapturing an individual marked rat in a mark-recapture study is $p = 0.3$. Move the p slider to 0.3 and the bar graph shows the probability of recapturing 0, 1, 2,…10 marked rats if there are 10 in the study area. (Of course, the problem is that you don't usually know the value for p!)

Check the box next to "Show 3d Probability". A 3-D graphic appears which looks the same as the Probability bar graph. Use the p slider to change the value of p and you will see that the position of the bar graph changes to match the value of p shown on the right-hand axis (going "into the page").

Now check the box next to "Show 3d curve", so you have two boxes checked. A series of blue curves appears, one for each bar of the bar graph. Play with the p slider and you will see that the blue curves correspond to the heights of the bars.

Notice that moving the "Number of successes (y)" slider makes no difference to anything so far.

Now check the third box. One of the blue curves in the 3-D graphics turns pink: which one depends on the setting of the y slider. The same pink curve is shown in two dimensions on the right of the graphic window.

The Probability graph (bottom left) shows all possible values of y for a single value of p, while the Likelihood graph (top right) shows all possible values of p for a single value of y.

Think of travelling in New York, where the east-west roads are called "avenues" and the north-south roads are "streets". You can move along the avenues or along the streets, or you can stand at an intersection, when you are in an avenue *and* a street. Similarly, you can move along a Probability Avenue and look at the probabilities of different results (y), or you can move along a Likelihood Street and view the likelihoods for different values of p. Or you can stand at an intersection, where both y and p are specified, and it then doesn't matter much if you talk about probability or likelihood because they

are equal[4]. The intersection of 5th Avenue and 42nd Street is the same whether you arrive there along 5th Avenue or along 42nd Street – but that doesn't mean that 5th Avenue is the same as 42nd Street!

Probability functions and likelihood functions are different, but they intersect. And at an intersection – where both *p* and *y* are specified – likelihood = probability and we can use the usual probability formulae to calculate likelihood.

Distributions

The last item on the menu is a kind of Appendix which you may want to refer to from time to time: it displays a variety of probability distributions and allows you to change the parameters and see how they affect the shape of the curve.

For some of the distributions you can also change the maximum and minimum values of the *x* or *y* axes. After typing in new values, click on the 'Refresh' button to redraw the plot.

Acknowledgements

Many of the functions included here are based in functions in the 'TeachingDemos' package for R by Greg Snow, some of them with few changes. Those which I have written are inspired by Greg's efforts. Many thanks to him and to all the core group and others who have worked to develop R over the years and perhaps especially to Peter Dalgaard, who produced the 'tcltk' package which provides the interface to the TK widgets.

[4] Strictly speaking, the likelihood is *proportional* to probability, ie. likelihood = probability x constant, but all statisticians agree to use 1 as the constant.