Maximum likelihood estimation is a procedure to determine the parameters of a statistical distribution that are most likely given a specific set of data. It is also typically used to compare multiple models and determine which model is the best fit for that data.

There are three steps involved in the model comparison procedure: 1) writing the likelihood function, 2) optimizing the function to determine the maximum likelihood estimator for the parameters, and 3) comparing maximum likelihood estimates for multiple models using the Akaike Information Criterion (AIC).

**Part 1: Writing a likelihood function**

Remember that likelihood is the chance of particular model parameters being true, given a certain observed set of data. You can think of it as the converse of probability, which is the chance of observing a certain set of data given particular model parameters.

It’s easier to work with the log-likelihood (the natural logarithm of the likelihood function). Log-likelihood is calculated as the sum of the logarithm of the probability density function evaluated at each observation (given specified model parameters). Written another way, it is the log-probability of the first value in the data occurring, plus the log-probability of the second value occurring, and so on until you reach the last value in the data set.

R has many built-in probability distributions, such as the normal, log-normal, gamma, binomial, Poisson, etc.; each is applicable for certain types of data, depending on the underlying processes that generated that data. You used the F distribution when writing the Hotelling T2 function (the pf command).

Each probability distribution has four commands: d (for probability density), p (for the area under the curve less that a given quantile), q (for the quantile or value corresponding to a specified area under the curve), and r (to generate random points). For example, you could use `dnorm()`, `pnorm()`, `qnorm()`, or `rnorm()` for the normal distribution.

The likelihood function will use `dnorm` (or `dlnorm`, `dbinom`, `dpois`, etc.) because you will need to calculate the probability or probability density of each data observation.

Finally, you will need to write the commands as a function, contained within braces `{ }` and with the input parameters that you want to fit. For example, a normal distribution has a mean and sd, so those are the parameters that your function must require. **Your function must output the negative log-likelihood.**

Here’s an example function to calculate the negative log-likelihood for a normal distribution.

```r
norm.ll <- function(mu, sigma) {
  #normal has mean mu and sd sigma
  normprob <- dnorm(data, mu, sigma)
  #the probability density of each data observation given the mean mu and sd sigma
  -sum(log(normprob))
  #the negative sum of the logarithm of probability densities
}
```
Because of the way the later optimization step works, you’ll have to make sure the name of the “data” value within the function matches the name of the actual data variable (you can’t include it as a parameter in the function).

**Part 2: Optimizing likelihood function and determining model fit**

There are a few functions for maximum likelihood estimation in different packages, but I recommend the bbmle package. You’ll need to download the code from the internet using `install.packages("bbmle")`. That only has to be done once, but you’ll need to load the package into your R session each time, using `library(bbmle)`.

The function to perform the MLE optimization is called `mle2()`.

Warning: technical details (skip this digression if you’re not interested). There is also a function called `mle()` but `mle2()` seems to be more stable. Optimization finds the minimum negative log-likelihood value by calculating the Hessian matrix. The Hessian matrix is a matrix of second-order partial derivatives, so it describes the curvature of a surface in multidimensional space and can therefore be used to find the minimum point on a multidimensional likelihood function surface. The `mle2()` function has better ways of dealing with problems that sometimes happen when inverting the Hessian matrix.

Back to the `mle2()` function! It requires two inputs: 1) a log-likelihood function and 2) a list of starting values for the parameters to be fit.

You will want to store the output of this function as a variable, because it will be used as input in the next step (calculating AIC).

In the example of the normal log-likelihood function given earlier, `mle2()` would be formatted like this:

```r
mle2(norm.ll, start = list(mu = 4, sigma = 3))
```

I randomly chose 4 for the starting mean and 3 for the starting standard deviation. It’s best if the starting points are in the ballpark of the expected maximum likelihood estimators, but it shouldn’t really matter what you choose as long as they are reasonable. If your chosen starting points are really weird, the function will either fail to converge on the solution (and it will say so), or it will just give you an error.

You will almost certainly get warning messages about NaN values being produced, so you can just ignore them.

**Part 3: Calculating Akaike Information Criterion (and weights)**

Note: this step is only meaningful if you compare multiple likelihood models that were all calculated from the same data! Do not compare models that are based on different data.
Also note that this step will tell you which of the models is the best fit, but it only compares the models you chose and is unable to assess models that you didn’t include. The best model of the selected group may not actually be an objectively good fit and it is always possible that some other (un-investigated) model is actually far superior.

The Akaike Information Criterion (AIC) is a measure of model fit that penalizes more complex models, because models with more parameters always have greater likelihood but also have more uncertainty (each parameter is estimated so has some error).

You should use the function to calculate the small-sample-size corrected AICc value. AICc is the same as AIC when sample size is large, but is better when sample size is small, so there is no drawback to using it. The function is `AICctab()` and it requires a series of maximum likelihood model results as the input. I recommend using the `weights=T` option, to display the Akaike weights (a measure of proportional model support) in addition to the differences between AICc values. You will also need to indicate the number of observations (i.e., the number of data points in the original variable) using `nobs=…)

For example (if you have some MLE fits called `mle_fit1` and `mle_fit2`):

```r
AICctab(mle_fit1, mle_fit2, weights=T, nobs=length(data))
```

### Part 4: Likelihood functions for linear regression fits

Maximum likelihood estimation is perhaps more commonly used for model selection (also called variable selection) in linear regression. The goal is to determine which independent predictor variable (or combination of predictors) best explains a dependent outcome variable.

Unlike in the previous example, the slope and intercept of a regression line aren’t probability density functions. This approach takes advantage of the fact that the residuals of data around a regression line follow (or should follow) a normal probability distribution. The residuals should be centered on the regression line (the mean of the residuals distribution should be the predicted value of the dependent variable) and should have a constant standard deviation at all points.

Here’s an example for a single independent variable. If you have a model with multiple independent variables, you will have multiple slope parameters (int + slope1*indep1 + slope2*indep2 + …).

```r
lm.ll <- function(int, slope, sigma) {
  predict_dep <- int + slope * indep
  residprob <- dnorm(dep, mean=predict_dep, sd=sigma)
  -sum(log(residprob))
}
```