# *In-Class Computing Task 8*

## *Math 253: Statistical Computing & Machine Learning*

## *Estimation and Likelihood*

Likelihood is the probability of the observed data given the model. In the previous in-class activity, you computed the likelihood for a set of models: exponential distributions with various values for the `rate` parameter. When considering a fixed data set, at each single value of `rate`, there is just one likelihood value. Calculating the likelihood for a set of `rates` allows you to find the value of `rates` that maximizes the likelihood. For the purpose of estimating the "best" value of a parameter like `rate` on the data set, the parameter with the maximum likelihood is usually taken as the estimate.

In this activity, you are going to use functions that find the maximum for you, without necessarily making a graph. This is particularly useful when the parameters for a model are a set of numbers, so graphing the likelihood as a function of all the parameters is not possible.

### *Task 1 — Fitting a line with likelihood*

1. Create a data frame called `My_data` that has two components:
   - `x` a set of 100 random numbers uniformly distributed over the interval 20 to 70.
   - `y` the values `5 + 3*x + 2*rnorm(100)`
2. Plot `My_data` to check that it looks as you expect.
3. Write a function `LL_line(params)`, where the argument `params` will be a vector of three numbers. The first of these represents `m` (a slope), the second represents `b` (an intercept) and the third represents `sigma` (a standard deviation). The function will calculate the log-likelihood of the observations `y`. The core of the function will look like

   ```
   sum(log(dnorm(y - (m * x + b), sd = sigma)))
   ```

4. Create two objects, `testA` and `testB` containing the value calculated by `LL_line()` for two different possible parameters:
   - for A: $m = 3, b = 5, \sigma = 2$
   - for B: $m = 4, b = 1, \sigma = 10$
5. `LL_line()` is a mathematical function of 3 variables: $m$, $b$, and $\sigma$. As such, it's not easy to plot. Instead, we will rely on another optimization function in R, `optim()` that works for functions of several variables. Rather than specifying an interval to search, you specify a starting guess for the argmax. Use it like this:

```
starting_params <- c(4, 1, 10)
```

```
best <- optim(starting_params, LL_line, control = list(fnscale = -1))
```

That awkward looking `control = list(fnscale = -1)` means "find the maximum rather than the minimum." This is how the programmer who wrote `optim()` decided to control such things.

For fitting a line with the assumption that residuals are normally and identically distributed, it's best to use the techniques derived mathematically from maximum likelihood rather than a brute-force simulation. But change the model a bit, e.g. errors with variance that changes with $x$, or errors with an exponential distribution, and the brute force approach is appropriate.

*Task 2 — Taxicab fare structure*

File `Taxi_trips.rda` is a subset of a much larger data set from the New York City Taxi and Limosine Commission about taxicab trips in New York City.

```
load(url("http://tiny.cc/dcf/Taxi_trips.rda"))
```

According to regulations, the `fare_amount` depends on the `trip_distance` in a simple linear way. But the `fare_amount` also depends on how long the taxi was standing still during the trip. This standing-still time can never be negative, so estimating the parameters of the linear relationship between `fare_amount` and `trip_distance` cannot be accomplished with a least-squares approach.

The waiting-time part of the fare is
`waiting_fare <= fare_amount - (base_fare + per_mile*trip_distance)`.

Each individual trip's waiting time is modeled as random. You can choose whatever probability distribution you like, but it should be such that the probability density for negative values is zero (or, if you like, very small). For instance, an exponential distribution might be appropriate.[1]

- Write a function `taxi_likelihood()` that takes as an argument a vector like `c(base_fare = 2.00, per_mile = 4.00, params = 1/3)`.

With `taxi_likelihood()`, use `optim()` to find the maximum likelihood estimate of `base_fare`, `per_mile`, and the `params`. Store the result of `optim()` under the name `best`.

Plot `fare_amount` versus `trip_distance` as a scatter plot. Then add a line showing the linear relationship you found, which you can do with `abline(base_fare, per_mile)`.

Use your fitted model to estimate the component of each trip's fare that's due to standing-still time.

[1] When using an exponential distribution, keep in mind that there is a hard cut-off at an input of zero: the probability of any negative value is exactly zero, meaning "impossible." (Zero likelihood corresponds to `-Inf` log-likelihood. Yet when searching for the maximum likelihood, you may encounter parameter values where `-Inf` is the result. Sophisticated methods of dealing with such boundaries are beyond the scope of this course, but you can always get around the problem by adding a small positive value to the likelihood before taking the log.)