Some "universal" statistical methods

1. **Cross-validation for kernel estimators**
   For the i.i.d. ISI's, $X_1, ...X_n$, let us look more carefully to the kernel estimators of the underlying density f. Choose $K(x) = \exp(-x^2/2)/\sqrt{2\pi}$. The kernel estimator with bandwidth h is
   $$\hat{f}_h(u) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{u - X_i}{h}\right).$$
   (a) Implement the function in R which compute for a given vector u, and a given window h, the value $\hat{f}_h(.)$ for each coordinate of the vector u. Simulate some exponential variables and try different values for h.

      NB: the R command `density` could more or less do the trick but, as far as I understand it, you cannot decide of the u to put in entries and we need that, so we need to reprogram it... Just follow the formula.

   (b) To choose h, we can use cross-validation. To do that, we use the following least-square contrast
   $$C(g) = -\frac{2}{n}\sum_{i=1}^{n} g(X_i) + \int g(x)^2 dx.$$
      Show that $E(C(g))$ is minimal when $f = g$.

   (c) Use the function `integrate` to compute the integral of the kernel estimator to the square and implement a function in R to compute the least-square contrast on a different sample than the one used for computing the estimator.

   (d) Implement the Hold-out estimator, which consists in cutting the data in half at random and to use half for estimation, half for the selection of h. Plot the corresponding estimator.

   (e) Implement the V-fold cross validation estimator as a function of R. Plot the corresponding estimator.

2. **Parametric Bootstrap** (again with ISI's)

   (a) Simulate $n = 50$ exponential variables with parameter $\theta_0 = 2$. This will be considered as your observations.

   (b) Compute the MLE $\hat{\theta}_{obs}$ of θ for $n = 50$ i.i.d. exponential ISI's.

   (c) By simulating $N_{simu} = 10000$ times n exponential variables with parameter $\theta_0 = 2$, compute an approximate classical confidence interval IC_1 on θ with confidence 95% *NB : to do that, you can for instance compute $D_i = |\hat{\theta}_i - \theta_0|$ on each simulation and decide that with probability approximately 95%, the distance between $\hat{\theta}$ and θ_0 is less than the 95% empirical quantile of D_i. NB 2 : you cannot do that on real data since you do not know θ_0*

(d) By parametrically bootstrapping the data, compute a bootstrap confidence interval IC_2. To do so, simulate N times n exponential variables with parameter $\theta = \hat{\theta}$ and do as before with respect to the simulation.

(e) Compare IC_1 and IC_2 for various size of N. Explain why $IC2$ cannot converge to $IC1$ when N tends to infinity, even if it is a pretty good approximation.

(f) Apply it on the data of the package STAR to show that you have a clear difference on neuron 1 of the experiment citronellal before the puff (same data as Tutorial 1), during the puff and after the puff. *NB: do not forget to change the level of the confidence interval for Bonferroni correction.*

3. **Non parametric bootstrap** (with firing rates and not ISI's) Still on the same data of the package STAR, we want to compute non parametric bootstrap confidence intervals on the firing rates. A firing rate is defined by the average number of spikes per second.

   (a) Compute an estimation of the firing rate for each of the three periods for the experiment citronellal and for each of the trials.

   (b) Code a R function for computing bootstrap confidence interval on the mean at a given level α. Test it on simulated data to see if it works ...

   (c) Apply it on the estimated firing rates of each period in a)and see if you can conclude non parametrically on a difference between the three periods in terms of firing rates.

4. **Clustering** Ingrid Bethus recorded (in many different sessions) 86 neurons of the striatum of a rat performing a spatial task. In `action_potential.Rdata`, you will find the average shape of their action potential as recorded by the tetrodes. More precisely, after typing `load(action_potential.Rdata)` you will find in the variable `Record` a list. Then `Record[[i]]` is a matrix with 32 lines and 86 columns (one column=one neuron), the index i referring to the electrode number i of the tetrode. The line j in the matrix corresponds to the value of the action potential at $j * 0.0001$ second after the beginning of the action potential.

   (a) Plot for some neurons the superposition of the 4 forms that are corresponding to the action potential as seen by each of the electrodes. See that the heights of the signal are very differents (this comes from the proximity of the electrodes) and that the highest signal is not always recorded by the same electrode.

   (b) We focus on the best shape, that is the shape with the heighest peak on all electrodes. On this curve, we extract two parameters that represents the action potential : the duration of the peak above half its height, the delay between the time of the valley after the peak and the time of the peak. Extract these two values for each of the neuron.

   (c) Before doing hierarchical clustering, center and renormalize both quantities (as when doing PCA) and put this into a matrix `x` with 2 columns and 86 rows. Then use `plot(hclust(dist(x),method="ward.D2"))` to see the dendogram.

   NB : the dendogram is a visual representation of the data. The two closest points are merged together and represented by their middle (or barycenter) then the algorithm continue and can eventually merge points and barycenters, or two barycenters together until everything is merged. This process of merging is plot as a tree : when two quantities with different heights are merged there are new branches growing and the shortest one correspond to the distance between the centers.

   Find the largest distance between clusters and cut the tree there. Conclude that clearly we have two clusters.

(d) The commands `z=kmeans(x,2)` and `z$cluster` give the cluster label for the different neurons. Plot the neurons in the plane of the two variables with a different color code for the first cluster and second cluster as estimated by k-means. You have identified the "fast spiking interneurons" (the ones with small durations for each delays) and the "medium spiny neurons".