Queen Mary University of London

School of Biological and Chemical Sciences

Coursework Coversheet

Student Details
Place barcode in the dashed box below.

1 7 0 4 8 4 3 6 7

GRIGORIADIS, Dionysios

MARK

BIO782P		PARKER, Joe
STATISTICS FOR BIOINFORMATICIANS		
WEEK 2 ASSESSMENT		
Friday, 1 December 2017, 5:00 PM		

Feedback

If generic feedback is provided on the module webpage then please indicate by ticking here: ☐

General comments

Suggestions for improvement

Declaration [to be completed by student]

I certify that this coursework that I am submitting is my own work, that it has not been copied in part or in whole from any other person, and that any ideas or quotations from the work of other people, published or otherwise, are properly referenced. I have read and understood the School guidelines on plagiarism and I am aware that penalties will be applied for any plagiarism or other poor academic practice.

	30/11/2017

Part 1

1. The function named “predict” takes a numeric value (activation_threshold) and a list of numbers (stimulation_voltage) and returns a list of numbers (activation) of equal length to the input list. Each element of the output list (activation) takes -70 as its value if the corresponding element of the input list is less than or equal to the activation_threshold, while takes 40 as its value if the corresponding element of the input list is greater than the activation_threshold. This function predicts the list of the response voltages of the axon given a list of stimulation voltages and an activation threshold by fitting a model.

“Calculate_errors” function, on the other hand, takes two lists as its argument and returns their Euclidean distance as output (total_errors). The first argument list (predicted) could be a prediction that “predict” function did, while the second argument list (observed) could be the reponse voltages that had been experimentally measured. In this way, this function calculates the total error of a prediction made.

The third function “fit_threshold” takes two lists of numbers (input_values_stimulation & Input_values_response) and a number (threshold), creates a predicted list (predicted_values) of response voltages of the axon given a list of stimulation voltages (input_values_stimulation) and an activation threshold (threshold) by calling the “predict” function. Then it calculates the Euclidean distance between the created list of predicted response values and the given list of observed response values (Input_values_response) by calling the “calculate_errors” function. Eventually, it stores this distance in a variable and returns it.

2. The response variable of the model is the response voltages list, while the explanatory variable is the stimulation voltages list. The given code is trying to predict the response variable given the explanatory variable and the activation threshold. According to the experimental data, for each stimulation voltage value which is below or equal to the activation threshold of the axon, the corresponding response voltage value takes -70 mV as its argument, while for each stimulation with a voltage value greater than the activation threshold of the axon, the corresponding response voltage takes a value near to 40 mV but less. Considering all of the above, the code

writer fitted the following model as follows: For an x simulation voltage value and an activation threshold (t):

$$f(x) = -70 \text{ for } x \leq t$$

$$f(x) = 40 \text{ for } x > t$$

where f(x) corresponds to the response voltage value.

3. The dataset has 15 observations (voltage values) and the above described model has one explanatory variable which is continuous. Thus:

degrees of freedom in the model = 15 -1 = 14

degrees of freedom in the residuals = 14 -1 = 13

4. Each time that the code is executed a random value from a uniform distribution with -100 as lower limit and 40 as upper limit is picked. Then the function fit_threshold is called using as arguments the stimulation_potential list and the response_voltage_control list from the experimental data and the threshold that was randomly selected before and calculates the goodness of fit of the predicted response voltages list with the response_voltage_control list. Eventually, calculates the goodness of fit of the used model with the given threshold. Then the model is optimized with the following loop.

5. A random generation of an activation threshold from a uniform distribution is used as a proposal mechanism for the model optimisation. Each time that the loop generates a random threshold, a new model is fitted to the control stimulation data which predicts the response voltage values. The goodness of fit of each new fitted model is compared to the previous model which had the lowest error so far. Then the process is repeated. The loop is executed for 20 times and the first model that is compared to the new fitted one is the model that was fitted outside of the loop (see question 4).

6 & 7. These questions were answered as comments to the script submitted (cwk2_DG.R).

Part 2

1.

- A docker image should be built in a local machine (due to permission issues on apocrita). To do that, it is essential to create a dockerfile (containing the given pseudocode) in a location in which the essential files are also included (the script). The name of the image should be the same as in jobscript.
- The image is tagged and pushed to the docker hub.
- On apocrita the image is pulled from the dockerhub with the appropriate singularity commands.
- The jobscript and the image are placed in the same directory and the job (that the jobscript describes) is submitted.

2. The jobfile was re-writed below as required. Explanation about the changes can be found as comments on the script below.

```
#$ -cwd
#$ -S /bin/bash
#$ -j y
#$ -pe smp 1
#$ -l h_vmem=2G
#$ -t 1-45

mkdir output
mkdir output/log_files$SGE_TASK_ID #To create a log_files folder for each iteration.

module load singularity
singularity run activationContainer.img input_data.csv -p 10 #The script is running.
#random seed.

mv final.csv final$SGE_TASK_ID.csv
mv final$SGE_TASK_ID.csv output #final.csv with a unique name for each iteration is
#moved to the output directory.

mv fit* output/log_files$SGE_TASK_ID #logfiles from each iteration will be inserted
#into the corresponding directory inside output.
```