Part A

**# Input Data**

stimulation\_potential = seq(-100, 40, by=10)

response\_voltage\_control = c(-70.00000,-70.00000,-70.00000,-70.00000,-70.00000,35.79871,38.75082,39.30423,39.65390,35.55835,35.66762,36.55754,39.51027,35.65274,39.17013)

response\_voltage\_concN\_1 = c(-70.00000,-70.00000,-70.00000,-70.00000,37.88144,35.64345,36.54319,36.18773,36.67542,38.56634,38.53102,37.73421,38.69826,38.67744,37.21896)

response\_voltage\_concN\_2 = c(-70.00000,-70.00000,-70.00000,37.16110,39.28471,36.57506,38.03037,39.37853,39.92131,39.01116,39.46393,36.16073,37.70418,35.88598,38.53137)

response\_voltage\_concN\_3 = c(-70.00000,-70.00000,36.25729,35.70035,35.61478,36.81245,37.01299,39.34139,36.59520,35.00749,37.94440,36.10611,37.15617,35.29182,35.32412)

plot(stimulation\_potential,stimulation\_potential,type='n',xlab='Stimulation, mV',ylab='Excitation, mV')

lines(stimulation\_potential,response\_voltage\_control)

lines(stimulation\_potential,response\_voltage\_concN\_1,col="green")

lines(stimulation\_potential,response\_voltage\_concN\_2,col="orange")

lines(stimulation\_potential,response\_voltage\_concN\_3,col="red")

legend(0,20,"control",fill="black")

legend(0,00,"concentration 1",fill="green")

legend(0,-20,"concentration 2",fill="orange")

legend(0,-40,"concentration 3",fill="red")

**# Data model**

predict = function(activation\_threshold,stimulation\_voltage){

# Takes two inputs: activation threshold value, and a vector of stimulation voltage

# Returns a vector of the predicted axon response at the given threshold for each stimulation voltage.

activation = rep(-70,length(stimulation\_voltage)) # creates a vector of axon voltage at rest

activation[stimulation\_voltage>activation\_threshold] = 40 # if the stimulation to the axon is greater than the activation threshold the axon is excited

return(activation) # returns vector of axon voltage in respone to the stimulation voltage

}

calculate\_errors = function(predicted, observed){

# Takes 2 inputs: predicted values and observed values.

# Returns the sums of squares of the observed from the expected. A measure of how well the prediction fits the observed (error).

total\_errors = sum((observed - predicted)^2)

return(total\_errors)

}

fit\_threshold = function(input\_values\_stimulation,input\_values\_response,threshold){

# Takes three inputs: input stimulation, input response, and the threshold value.

# Predicts a response from the threshold and input\_values\_stimulation.

# Error of predicted response calculated from input\_values\_response.

predicted\_values = predict(threshold,input\_values\_stimulation) # creates vector of predicted axon response given the threshold and input stimulation

fit\_errors = calculate\_errors(predicted\_values,input\_values\_response) # calculates how well the predicted response fits the actual response

return(fit\_errors)

}

# random threshold selected and predicted model error calculated

# compared to means for control and all nicotine concentrations.

fitted\_threshold = runif(1,-100,40)

error\_control = fit\_threshold(stimulation\_potential,response\_voltage\_control,fitted\_threshold)

error\_concN\_1 = fit\_threshold(stimulation\_potential,response\_voltage\_concN\_1, fitted\_threshold)

error\_concN\_2 = fit\_threshold(stimulation\_potential,response\_voltage\_concN\_2, fitted\_threshold)

error\_concN\_3 = fit\_threshold(stimulation\_potential,response\_voltage\_concN\_3, fitted\_threshold)

# H0 = Assume nicotine conc has no effect

H0 = 'Nicotine concentration does not change activation threshold'

# generate new threshold and compare to old

# if new threshold generates smaller error, retain as best.

# for new best threshold calculate if H0 rejected and direction of change

for(i in 1:20){

new\_threshold = runif(1,-100,40)

new\_error\_control = fit\_threshold(stimulation\_potential,response\_voltage\_control,new\_threshold)

new\_error\_concN\_1 = fit\_threshold(stimulation\_potential,response\_voltage\_concN\_1,new\_threshold)

new\_error\_concN\_2 = fit\_threshold(stimulation\_potential,response\_voltage\_concN\_2,new\_threshold)

new\_error\_concN\_3 = fit\_threshold(stimulation\_potential,response\_voltage\_concN\_3,new\_threshold)

if(new\_error\_control < error\_control){

fitted\_threshold = new\_threshold

error\_control = new\_error\_control

error\_concN\_1 = new\_error\_concN\_1

error\_concN\_2 = new\_error\_concN\_2

error\_concN\_3 = new\_error\_concN\_3

test\_threshold = new\_threshold + 10

test\_error\_concN3 = fit\_threshold(stimulation\_potential,response\_voltage\_concN\_3,test\_threshold)

if(error\_control != error\_concN\_3){

if(new\_error\_concN\_3 < test\_error\_concN3){

H0 = 'Increasing nicotine causes decreased activation threshold'

}

if(new\_error\_concN\_3 > test\_error\_concN3){

H0 = 'Increasing nicotine causes increased activation threshold'

}

}

}

}

# best model, calculated errors, and hypothesis

fitted\_threshold

error\_control

error\_concN\_1

error\_concN\_2

error\_concN\_3

H0

# write to file

write.csv(data.frame('fitted value'=fitted\_threshold,'Control errors SS'=error\_control, 'Conc1 errors SS'=error\_concN\_1, 'Conc2 errors SS'=error\_concN\_2, 'Conc3 errors SS'=error\_concN\_3, 'Hypothesis'=H0),file = 'excitation.csv',row.names = F)

1. Describe what the inputs, ouputs and role for each of the functions predict, calculate\_errors and fit\_threshold are.

See commented code.

1. How is the data modelled?

The data is modelled by using a threshold stimulation value. If the axon is stimulated with a voltage below the threshold value, the excitation of the axon is -70mV. When the axon is stimulated with a voltage that is greater than or equal to the threshold value then the excitation of the avon is 40mV.

1. How many degrees of freedom are present in (a) the model, and (b) the residuals?

Model df = 1

Residuals = 13

1. Describe what happens when the code is executed, in statistical terms.

1 – A starting model is formed

A random value between -100 and 40 is chosen (4dp) as a threshold

A prediction is made using the random threshold.

The prediction is compared to the mean (control response) and the error (sums of squares) is calculated.

2 – Model is refined

i – A new model is formed in the same way as the initial model

* A random value between -100 and 40 is chosen (4dp) as a threshold
* A prediction is made compared to the mean and the error is calculated.

ii – The errors of the starting model and new model are compared.

* If the error of the starting model is less than the new model, then the starting model is retained as the “best”. If the error of the new model is less than the starting model, then the new model is retained as “best”

iii – Set repetition of the formation of new threshold and model. The threshold with the prediction giving the lowest error I sretained as "best".

3 – "Best" model (with smallest error compared to mean) retained.

4 – repeat steps 2 and 3 for 20 cycles. Overall “best model retained

1. How is the model fit to the data optimised?

The model is optimised through the use of a proposal mechanism. It generates new parameters to test (in this case a new threshold), and dictates that if the parameters make then model better, then the parameters should be updated.

For this model a new threshold is generated, if the error calculated for the new threshold is less than the original threshold then the threshold is updated as the new threshold is a better fit to the model.

1. How could you improve the optimisation?

The two most simple ways to improve the optimisation would be to decrease the range by which the parameter can be generated (based on the experimental data) and to increase the number of times the optimiser runs. This would increase the chance of a better threshold being randomly generated.

A more complicated method would be for the optimiser to be able to "learn". Narrowing the range by which the random threshold is picked by accounting for previous models and their errors.

1. How would you modify the code to:

a) fit the data given under the three experimental nicotine treatments.

[Added into code]. The prediction is compared to the mean of each tretment (ie, compared to each nicotine concentration response), and the error for each is calculated

b) compare the goodness of fit amongst these fitted models to determine whether each nicotine treatment lowers the activation threshold, compared to the control treatment.

[Added into code]. Alternate hypothesis and the direction of change are tested. When a "better" model if found for the control, the errors of the control and highest nicotine concentration are compared - if they are different the null hypothesis is rejected (ie nicotine changes the activation threshold).

A test threshold is generated (threshold+10). The error for the highest nicotine concentration is calculated from the threshold and the test threshold. If the error for the test threshold is higher than the threshold then nicotine concentration causes the activation threshold to decrease. If the error for the test threshold is lower than the threhold then the nicotine concentration causes the activation threshold to increase.

c) what is the null hypothesis in this case?

H0: Nicotine treatment has no effect and does not change the activation threshold.

Part B

We have modified your code so that it prints each fit out to a logfile as the optimisation progresses. The logfiles are called fit\_01, fit\_02... etc and print out each time optimisation loop runs. The final fitted\_threshold and errors will be written to another .csv, called final.csv. We have also modified the code to pass a user-specified random seed to the script for setting the initial value for fitted\_threshold. Pseudocode for a Docker container to run that code is given:

# Docker R base image

FROM r-base

COPY . /usr/local/src/myscripts

WORKDIR /usr/local/src/myscripts

CMD ["Rscript", "activation-thresholds.R"]

1. How would we use this code to run our analysis on a grid? Outline what steps we should take.

In suitable directory start instance of Docker

docker-machine start

Write a dockerfile and paste in pseudocode

touch Dockerfile # creates docker file to edit

… after Dockerfile editing, build the Docker container

docker build –t <container\_name> . # builds docker container, gives it ID and name

Tag as “threshold” and upload it to your Docker repository

docker tag <container\_name>

<dockerhub\_username>/<repository>:<threshold> # tags container

docker push <dockerhub\_username>/<repository>:<threshold> # upload

Go to the appropriate directory on Apocrita and pull the docker image using singularity

singularity pull <docker\_username>/<repository>:<threshold>

Write qsub file (threshold\_singularity.sh) that loads singularity and uses it to run the docker image.

touch threshold\_singularity.sh # creates file for job file script

Submit to the cluster

qsub threshold\_singularity.sh

1. Jobfile re-written to achieve the following:

* Move the final outputs to a directory (../output)?
* Change random seed (set with the -p [some integer] argument)?
* Run as an array job, with 42 replicates?

#$ -cwd

#$ -S /bin/bash

#$ -j y

#$ -pe smp 1

#$ -l h\_vmem=2G

#$ -o ../ouput\_directory # move final outputs to a directory

#$ -t 1-42 # number of iterations of script

Echo $SGE\_TASK\_ID # iteration counter

# generate random seed within allowed threshold range

R=$(($RANDOM%141-100))

module load singularity

singularity run activationContainer.img input\_data.csv –p $R