Lab 08

Jennifer Lin

jenniferyjlin@berkeley.edu (mailto:jenniferyjlin@berkeley.edu)

1

Read in the two sequences using the function provided and calculate the number of pairwise differences d.

```
# Read in the two sequences
x <- read.table('lab_08_data.txt',sep = '\t')
```

```
## Warning in read.table("lab_08_data.txt", sep = "\t"): incomplete final line
## found by readTableHeader on 'lab_08_data.txt'
```

```
seq1 <- strsplit(as.vector(x$V1[2]),split = '')[[1]]
seq2 <- strsplit(as.vector(x$V1[3]),split = '')[[1]]
# Calculate the number of pairwise differences d
d_result <- 0
S_result <- length(seq1)
for (i in 1:S_result){
  if(seq1[i] != seq2[i]){
    d_result = d_result + 1
  }
}
print(S_result) # total length of the sequence
```

```
## [1] 684
```

```
print(d_result) # pairwise differences d
```

```
## [1] 86
```

2

Implement a likelihood function based on the Jukes & Cantor model.

The likelihood function is parameterized in terms of λ which is the genetic distance.

λ is proportional to the mutation rate and the amount of time the two species have been diverging from each other.

The type of genetic distance is fundamental in many phylogenetic studies and other studies comparing DNA sequences.

```
# JC_likelihood
# ( 1/4 + 3*(exp(-4*parameter_lambda/3))/4 ) ^ ( S-d )  *  ( 1/4 - 1*(exp(-4*paramete
r_lambda/3))/4 ) ^ ( d )
```

3

Create an R function that calculates the logarithm of the likelihood.

This function requires 3 arguments: S,d, and λ.

```
JC_likelihood_log <- function(S, d, parameter_lambda){
  return(log(  ( 1/4 + 3*(exp(-4*parameter_lambda/3))/4 ) ^ ( S-d )  *  ( 1/4 - 1*(ex
p(-4*parameter_lambda/3))/4 ) ^ ( d )  ))
}
```

4

Use the optim function in R to optimize the log-likelihood function for λ using the value of S and d, from the two sequences.

Does this value match the one observed in the plot?

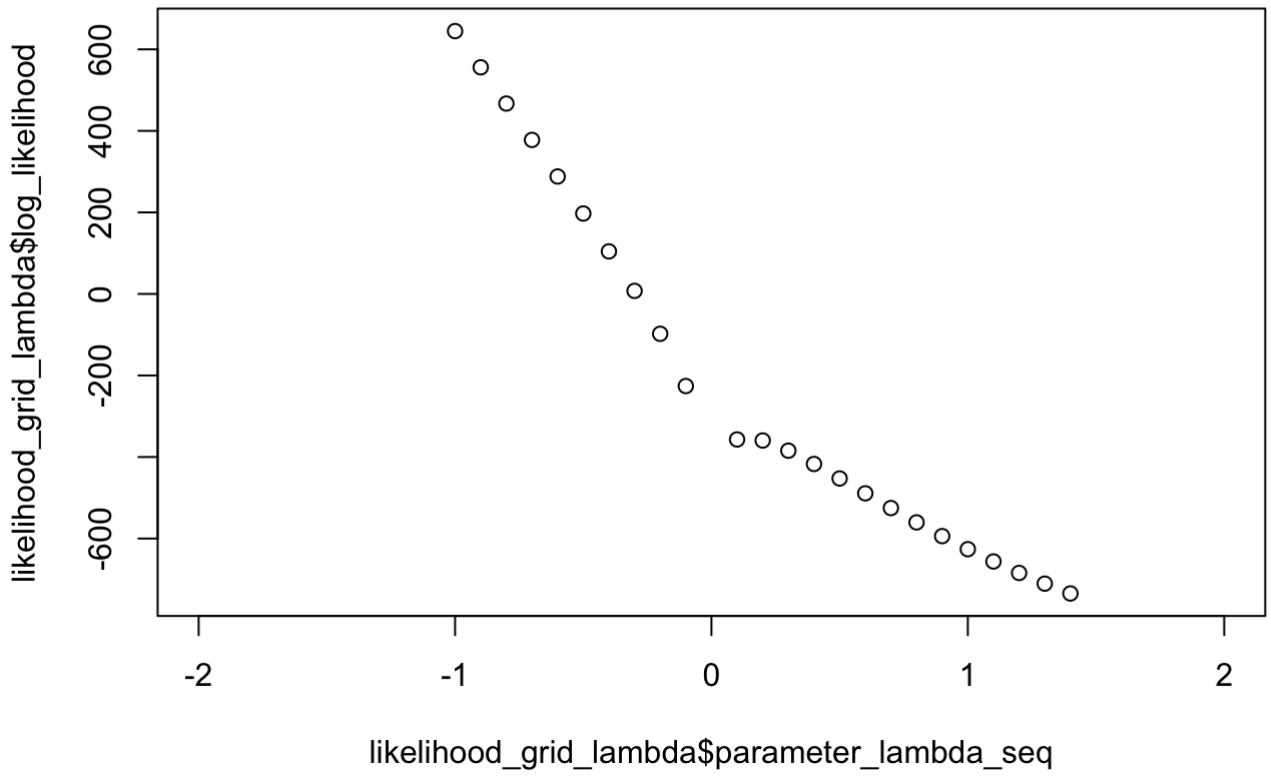
```
# Optim
optim_4 <- optim(par = c(1), fn = JC_likelihood_log, S = S_result, d = d_result)
```

```
## Warning in optim(par = c(1), fn = JC_likelihood_log, S = S_result, d = d_result):
one-dimensional optimization by Nelder-Mead is unreliable:
## use "Brent" or optimize() directly
```

```
print(optim_4$par[1]) # parameter_lambda
```

```
## [1] 1.44375
```

```
# Plot
parameter_lambda_seq <- seq(-2,2,by=0.1)
log_likelihood <- rep(NA,length(parameter_lambda_seq))
likelihood_grid_lambda <- as.data.frame(cbind(parameter_lambda_seq, log_likelihood))
for (i in 1:length(likelihood_grid_lambda$parameter_lambda_seq)){
  likelihood_grid_lambda$log_likelihood[i] <- JC_likelihood_log(S_result, d_result, l
ikelihood_grid_lambda$parameter_lambda_seq[i])
}
plot(y=likelihood_grid_lambda$log_likelihood, x=likelihood_grid_lambda$parameter_lamb
da_seq)
```




```
# Yes, the result value from "optim" matches the one observed in the plot (lambda = ~
1.4).
# (the minimum of log likelihood = the maximum of negative log likelihood)
```

5

Transitions (changes between A and G and between C and T) tends to occur at a higher rate than transversions (all other changes).

Calculate the number of transition ds and the number of transversions dv from the two sequences.

The Kimura two-parameter model, is specified in terms of two parameters α and β.

These two parameters can be interpreted as the rate of transitions and tranversions per unit time, respectively, multiplied by the total amount of time.

Make a function in R that calculates the logarithm of this likelihood. It should take 5 arguments: S, ds, dv, α, and β

```
K_likelihood_log <- function(S, ds, dv, parameters){
  parameter_alpha <- parameters[1]
  parameter_beta <- parameters[2]
  p1 <- ( 1/4 + 1*(exp(-4*parameter_beta))/4 + 1*(exp(-2*(parameter_alpha+parameter_b
eta)))/2 ) ^ ( S-dv-ds )
  p2 <- ( 1/4 + 1*(exp(-4*parameter_beta))/4 - 1*(exp(-2*(parameter_alpha+parameter_b
eta)))/2) ^ ( ds )
  p3 <- ( 1/4 - 1*(exp(-4*parameter_beta))/4 ) ^ ( dv )
  return(log(p1*p2*p3))
}
```

6

Plot the log likelihood function for the values of S, ds, and dv you obtained from the two sequences as a function of α and β in a contour plot.

Approximately where is the maximum likelihood estimate?

For contour plots, please use the plotly package, please refer to this link.

```
# calculate num_transversion and num_transition
num_transition <- 0
num_transversion <- 0
for (i in 1:length(seq1)){if (seq1[i] != seq2[i]){
  if ((seq1[i] == 'A' & seq2[i] == 'G') | (seq1[i] == 'G' & seq2[i] == 'A') | (seq1
[i] == 'T' & seq2[i] == 'C')| (seq1[i] == 'C' & seq2[i] == 'T')){
    num_transition <- num_transition + 1}
  if ((seq1[i] == 'A' & seq2[i] == 'C') | (seq1[i] == 'C' & seq2[i] == 'A') | (seq1
[i] == 'G' & seq2[i] == 'T') | (seq1[i] == 'T' & seq2[i] == 'G')){
    num_transversion <- num_transversion + 1}}}
# create a contour plot
library(plotly)
```

```
## Loading required package: ggplot2
```

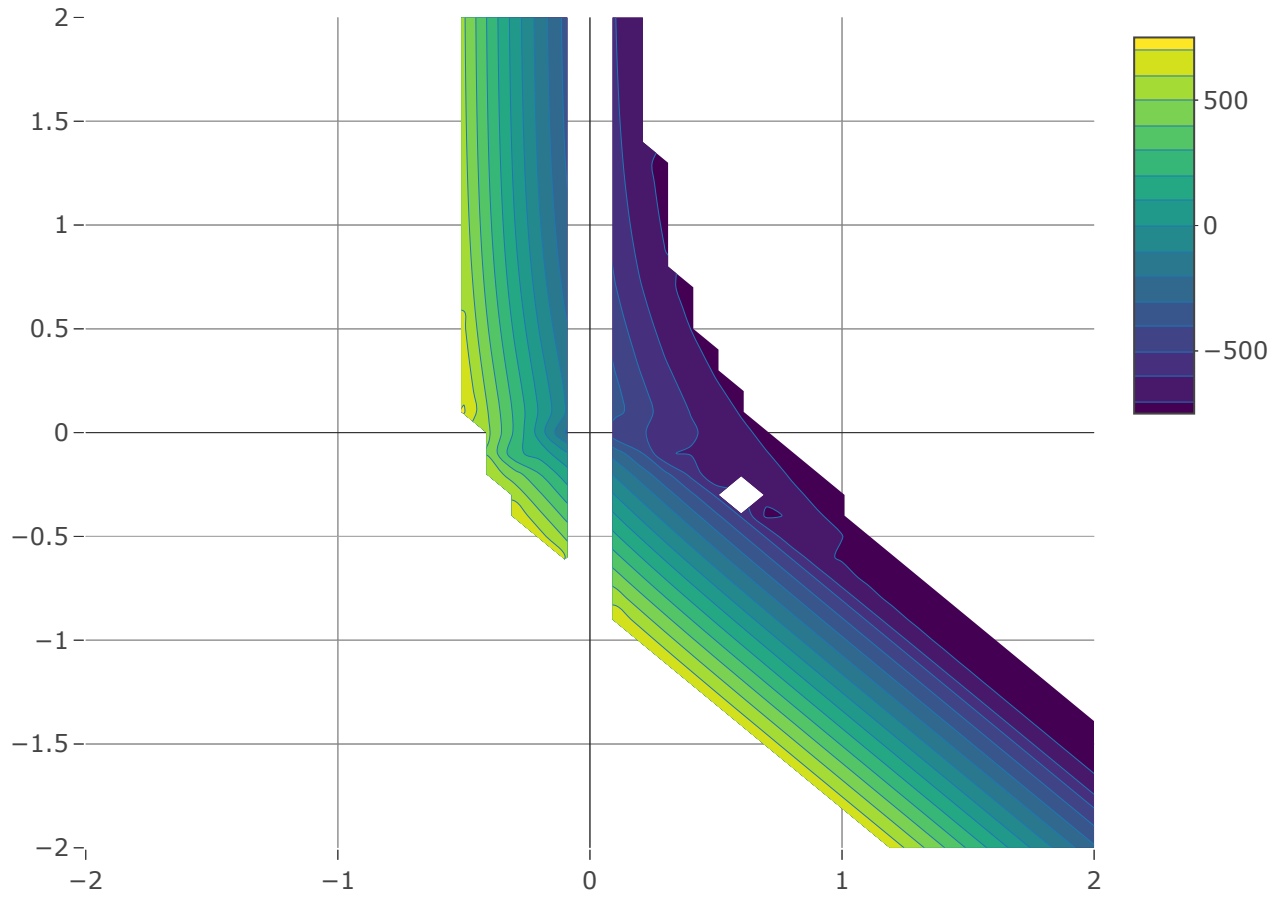
```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
parameter_alpha_seq <- seq(-2,2,by=0.1)
parameter_beta_seq <- seq(-2,2,by=0.1)
likelihood_grid <- matrix(NA, length(parameter_alpha_seq),length(parameter_beta_seq))
for (i in 1:length(parameter_alpha_seq)){
  for (j in 1:length(parameter_beta_seq)){
    likelihood_grid[i,j] <- K_likelihood_log(S=S_result, ds=num_transition, dv=num_tr
ansversion, parameters=c(parameter_alpha_seq[i],parameter_beta_seq[j]))
  }
}
likelihood_grid[which(!is.finite(likelihood_grid))] <- NA # Inf is replaced with NA i
n order to prevent the Error "Wasn't able to determine range of domain"
fig <- plot_ly(z = likelihood_grid, type = 'contour', x = parameter_alpha_seq, y = pa
rameter_beta_seq, colorscale = 'Viridis')
fig
```



7

Use the optim function in R to optimize this log-likelihood function for α and β using the value of S, ds, and dv from the two sequences.

Does this value match the one observed in the plot?

```
optim_7 <- optim(par = c(0,0.5), fn = K_likelihood_log, S = S_result, ds = num_transi
tion, dv = num_transversion)
print(optim_7$par[1]) # parameter_alpha
```

```
## [1] 0.07121582
```

```
print(optim_7$par[2]) # parameter_beta
```

```
## [1] 0.7099365
```

```
# Yes, the result value from "optim" roughly matches the one observed in the plot.
# (the minimum of log likelihood = the maximum of negative log likelihood)
# According to the contour plot, values in the right upper part are all quite low (we
are searching for minimum in this plot).
# Depends on the initial value of the optim function, we may have different results.
# However, these results are all very close to the right upper part of the contour pl
ot.
```