[Re] Local alignment statistics - Regression analysis by Altchul and Gish 1996

Nathan Brouwer

11/18/2019

Introduction

This script replicates the regression analysis discussed in Altschul and Gish (1996). It assumed you have Tabe 1 already loaded into memory OR load it using the code below.

Regression analysis is used to calculate K and lambda from mu (u) and the sequence lengths. The starting point is the following equation:

Plain text version of equation:

mu = (ln(m X n X K))/lambda

Rendered version of equation.

$$\mu = \frac{\ln(m * n * K)}{\lambda}$$

Altschul and Gish (1996) indicate that this equation can be re-arranged into a linear equation of the form y = M X x + b

where

- y is mu (the response or dependent variable)
- M is the slope
- x is ln(m X n) (the predictor or indepndent variable)
- b is the intercept

This gives us y = M X x + b mu = (1 / lambda) X ln(m X n) + ln(K)/lambda

From our simulations we will input sequences of length m and length n, and we will get out a distribution of scores. From the distribution of scores we'll be able to estimate mu. We can then make a scatter plot of mu versus ln(m X n) and use linear regression to estimate a line of best fit of the form y = M X x + b. The parameters for this line, M and b, can then be re-arranged algebracially to estimate lambda and ln(K)

M = 1/lambda

which rearranges to

M*lambda = 1

and then

lambda = 1/M.

So, the inverse of the slope from our regression will give us lambda.

Once we have lambda, we can solve for K

b = ln(K)/lambda

b X lambda = ln(K)

exp(b X lambda) = exp(ln(K))

exp(b X lambda) = K

We just got lambda above, so we plug that value into the equation and get K.

We can see how the original linear equation gets converted be recalling that ln(a X b) = ln(a) + ln(b)

Therefore, ln(m X n X K) can be re-arragned to ln(k) + ln(m X n)

So, starting with mu = (ln(m X n X K))/lambda

We apply the addition rule of logs we just outlined mu = (ln(m X n) + ln(K))/lambda

and we get mu = ln(m X n)/lambda + ln(K)/lambda

which is equivalent to

mu = (1/lambda) X ln(m X n)+ ln(K)/lambda

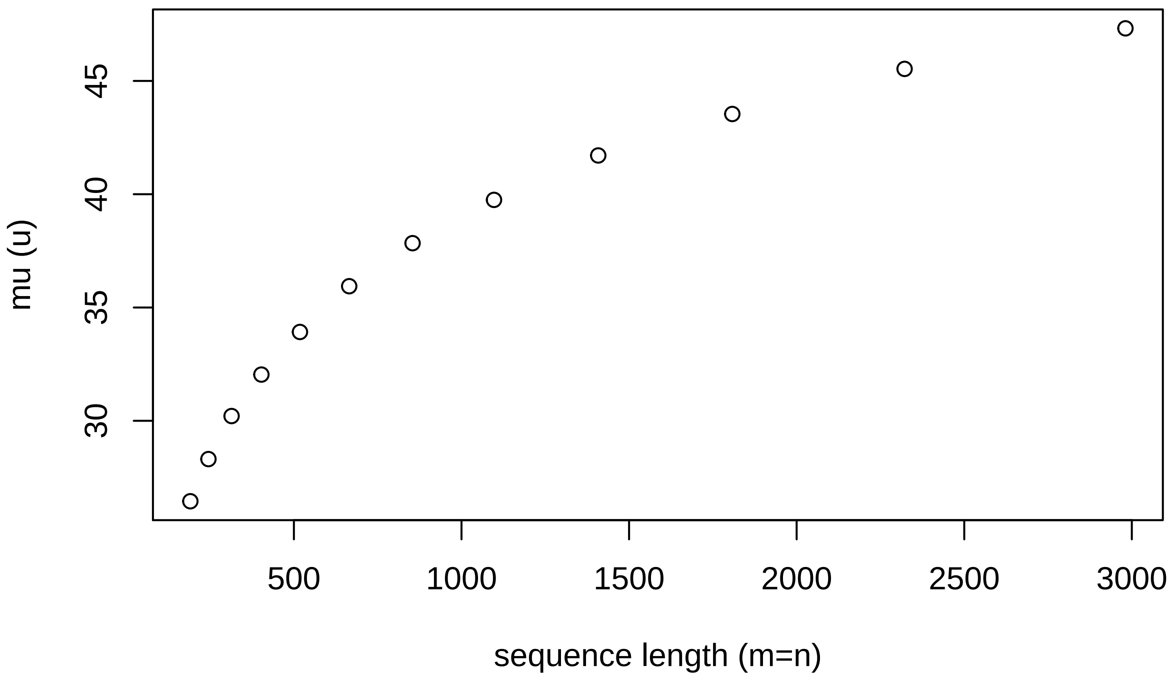
Load Table 1

```r
table1 <- read.csv(file = "table1.csv")
```

Regression analysis

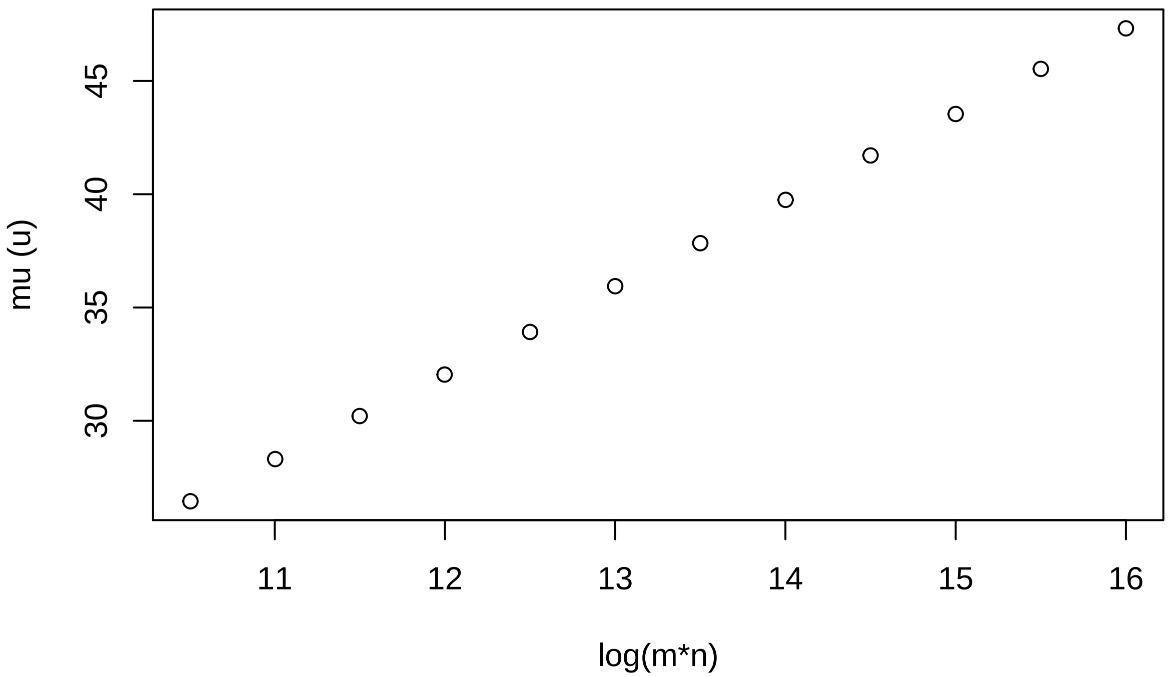
There is a non-linear relationship betwen length of the simulated sequences and the estimate of mu

```r
plot(table1$u ~ table1$mn,
     xlab = "sequence length (m=n)",
     ylab = "mu (u)")
```



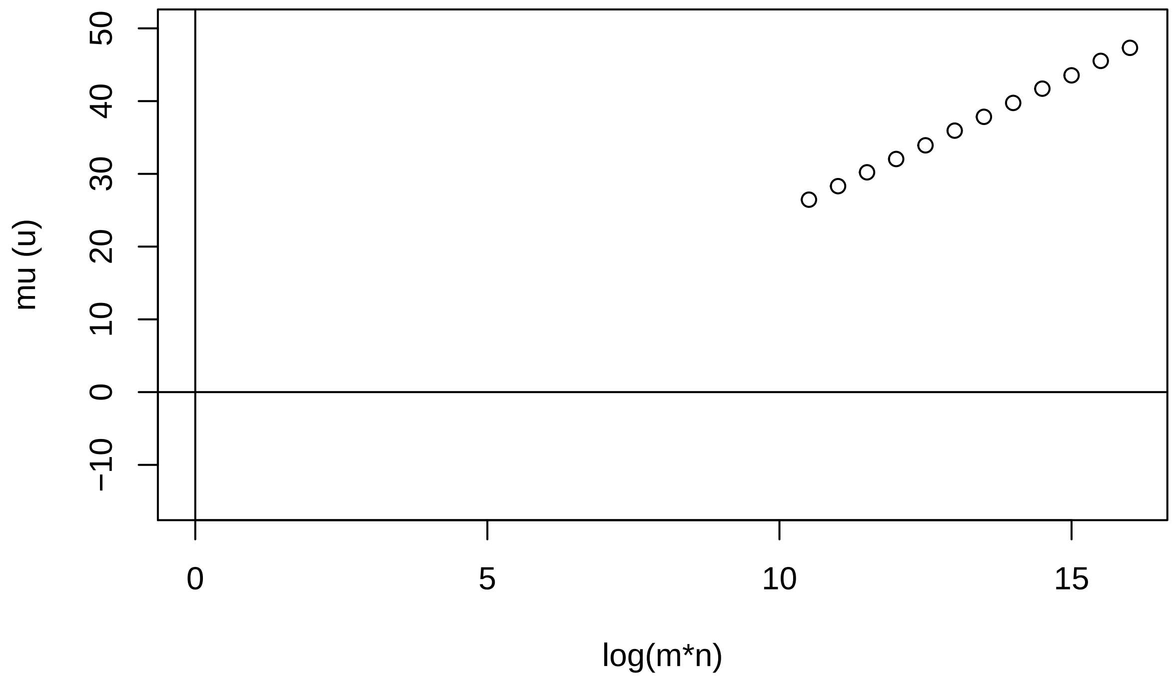
Taking the log of m*n (the search space)

```
plot(table1$u ~ table1$ln.nm,
     xlab = "log(m*n)",
     ylab = "mu (u)")
```



Plot log(m*n) vs u, with the axes adjusted so the origin (x = 0, y = 0) is visible.

```
plot(table1$u ~ table1$ln.nm,
     xlim = c(0,16),
     ylim = c(-15,50),
     xlab = "log(m*n)",
     ylab = "mu (u)")
abline(h = 0)
abline(v = 0)
```



Linear regression using lm()

```
# linear regression
lin.reg <- lm(u ~ ln.nm, data = table1)

# look at output
summary(lin.reg)
```

```
##
## Call:
## lm(formula = u ~ ln.nm, data = table1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.095304 -0.027192  0.008233  0.024556  0.063545
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -13.697825   0.115689  -118.4   <2e-16 ***
## ln.nm         3.817050   0.008658   440.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05174 on 10 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 1.944e+05 on 1 and 10 DF,  p-value: < 2.2e-16
```

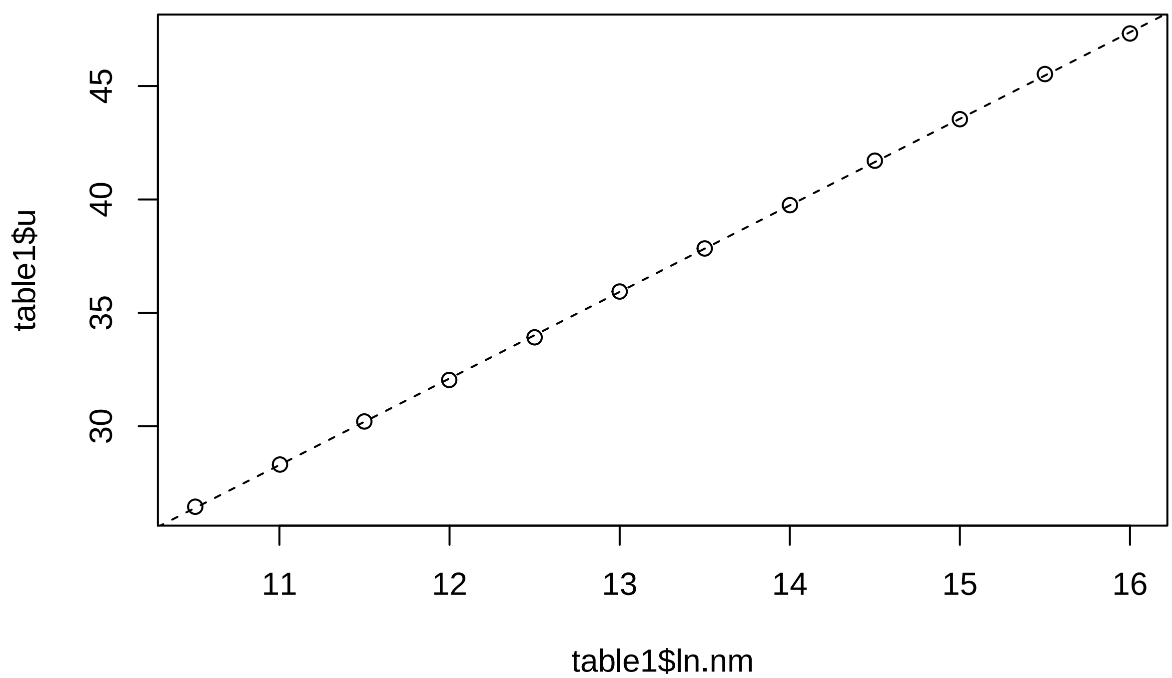
```r
# look at intercept and slope
coef(lin.reg)
```

```
## (Intercept)       ln.nm
##   -13.69782     3.81705
```

Plot regression line on plot scaled normally

```r
# plot data; no adjusted to xlim or ylim
plot(table1$u ~ table1$ln.nm)

# add regression line
abline(lin.reg, lty = 2)
```



Plot regression line on plot scaled with origin visible

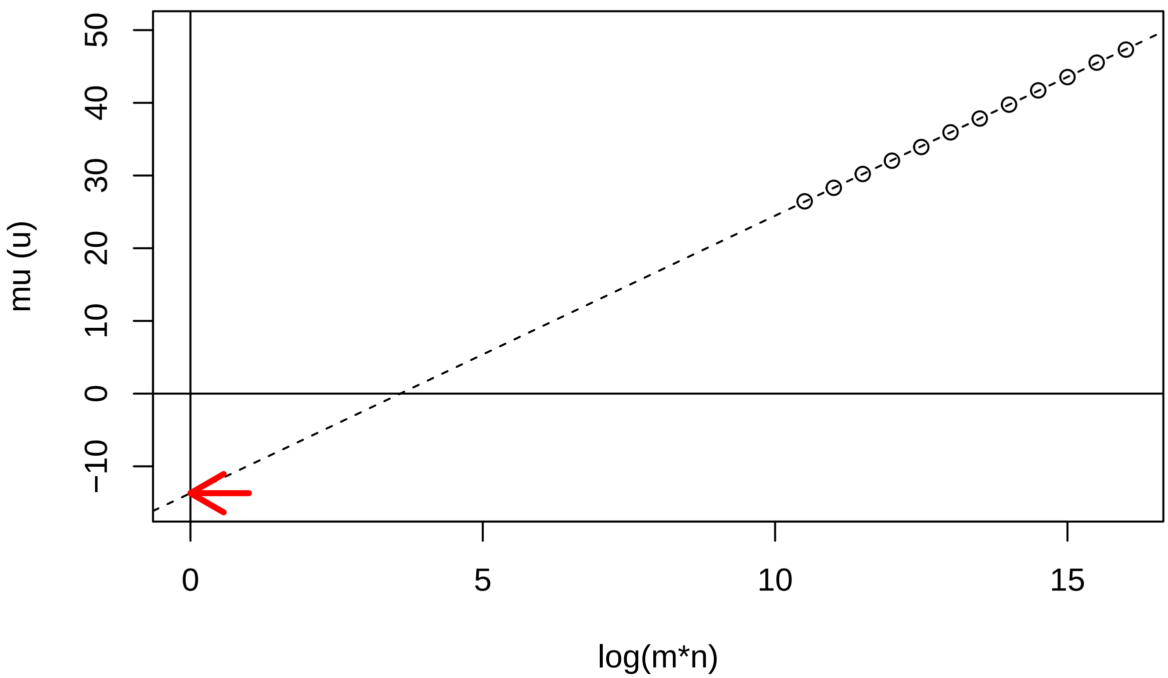
```r
# plot data
plot(table1$u ~ table1$ln.nm,
     xlim = c(0,16),
     ylim = c(-15,50),
     xlab = "log(m*n)",
     ylab = "mu (u)")

#add regression line
abline(lin.reg, lty = 2)

#add reference lines
abline(h = 0)
```

```r
abline(v = 0)

# show where y intercept is
arrows(x1 =0,
       x0 = 1,
       y0 = -13.69782,
       y1 = -13.69782 ,length  = 0.2, lwd = 3, col = 2)
```



Extract parameters from linear regression

```r
#y = M*x + b
#y = M*ln(m*n) + b
# M = slope
# b = intercept
b      <- coef(lin.reg)[1]  #intercept; ln(K)/lambda
M      <- coef(lin.reg)[2]  #slope; 1/lambda



# M = 1 / lambda
# re-arrange to:
# M*lambda = 1
# lambda = 1/M
lambda <- 1/M   # 0.261 reported in original paper
lambda
```

```
##     ln.nm
```

```
## 0.2619824
```

```
# b = ln(K)/lambda
# b*lambda = ln(K)
# exp(b*lambda) = K
K <- exp(b*lambda)  # 0.026 reported in Table 2
K
```

```
## (Intercept)
##   0.0276373
```

Advanced/Optional: Regression analysis on "edged-corrected" length

This is same idea as above, except using the "edge-corrected" distances discussed by Altschul and Gish. This reduces the discrepencies between the results of their experiment and theoretical predictions.

```
lin.reg2 <- lm(u ~ ln.n.m., data = table1)
summary(lin.reg2)
```

```
##
## Call:
## lm(formula = u ~ ln.n.m., data = table1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.125323 -0.025757 -0.000993  0.025927  0.118919
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -11.32176    0.15454  -73.26 5.48e-15 ***
## ln.n.m.       3.67345    0.01167  314.80  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07246 on 10 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 9.91e+04 on 1 and 10 DF,  p-value: < 2.2e-16
```

```
b2     <- coef(lin.reg2)[1]  #intercept; ln(K)/lambda
M2     <- coef(lin.reg2)[2]  #slope; 1/lambda

lambda2 <- 1/M2   # 0.261 reported in original paper
lambda2
```

```
##   ln.n.m.
## 0.2722238
```

```
K2 <- exp(b2*lambda2)  # 0.026 reported in Table 2
K2
```

```
## (Intercept)
##    0.045865
```