Project 1: Genome Scale algorithms

Maria, Mateo and Jacob

2/20/2018

```
# Import Naive
df_Naive = read.csv("Test ExactSubstring_Naive.csv", header = FALSE)

colnames(df_Naive) = c("Function", "Time", "n", "m", "z", "RunName")

df_Naive$RunName = factor(x = df_Naive$RunName)
df_Naive$Function = factor(x = df_Naive$Function)

# Import KMP
df_KMP = read.csv("Test ExactSubstring_KMP.csv", header = FALSE)

colnames(df_KMP) = c("Function", "Time", "n", "m", "z", "RunName")

df_KMP$RunName = factor(x = df_KMP$RunName)
df_KMP$Function = factor(x = df_KMP$Function)
```

Program description

The programs are implemented in Python, we had no unsovled issues implementing the algorithms. Execution of the program on commandline takes the form: python search-naive.py [text-filename] [str-pattern] python search-kmp.py [text-filename] [str-pattern] and the output is directly to the console.

Test of search-naive

python search-naive.py banana.txt ana 2 4

python search-naive.py mississippi.txt ss 3 6

python search-naive.py walrus-and-carpenter.txt Walrus 5 660 915 1059 1773 1951 2326 2617 2855 3050

python search-naive.py ancient-mariner.txt Albatross 2876 2991 3272 3443 5586 6899 14180 19139 23729

Test of search-kmp

python search-kmp.py banana.txt ana 2 4

python search-kmp.py mississippi.txt ss 3 6

python search-kmp.py walrus-and-carpenter.txt Walrus 5 660 915 1059 1773 1951 2326 2617 2855 3050

python search-kmp.py ancient-mariner.txt Albatross 2876 2991 3272 3443 5586 6899 14180 19139 23729

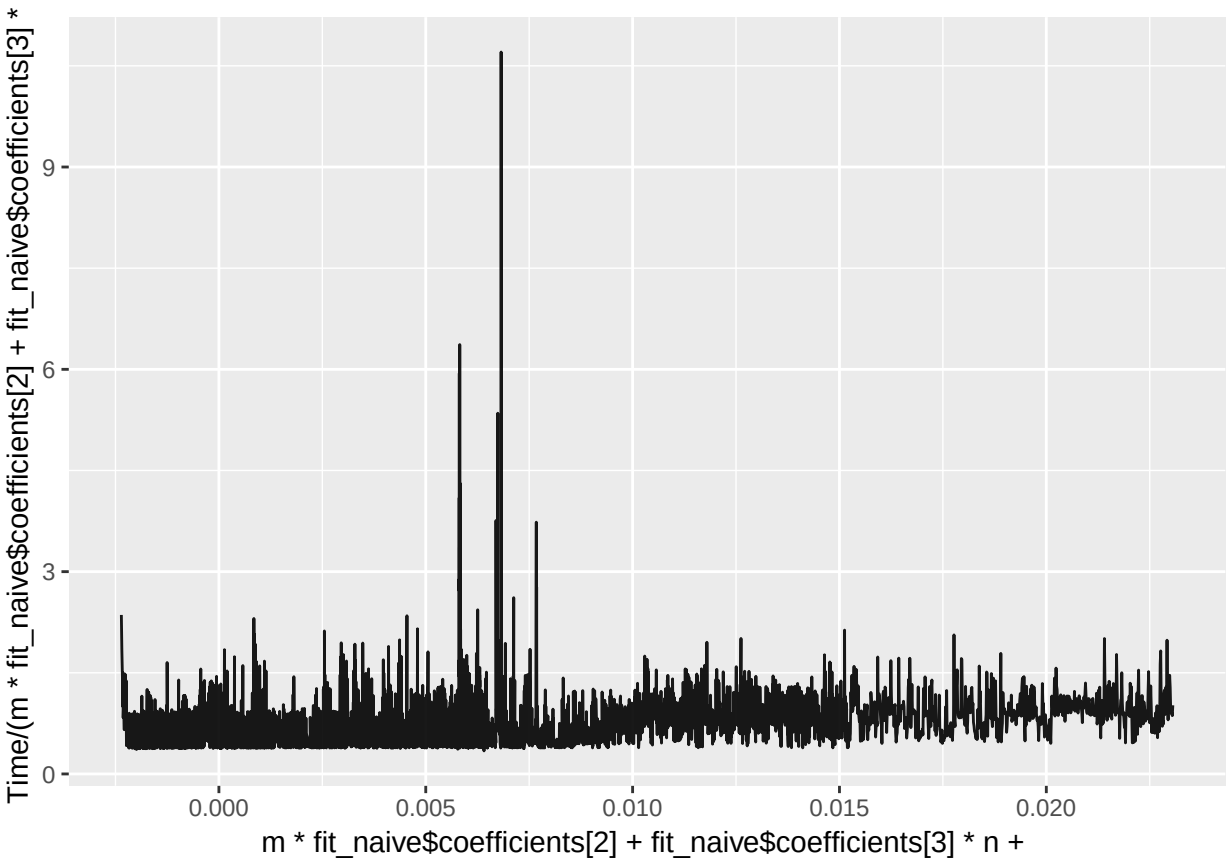
Runtime - Naive

The implementation of the naive algorithm runs in O(n * m) time. With best case input at 0 occurrences of pattern in the text, and worst case is constant matching to keep the inner loop active (m = n^0.5).

The following plot shows the runtime / regression of the Y axis and regression on X axis. Where the regression is: Time ~ n + m. It clearly shows that the runtime is more than linear.

```
fit_naive = lm( data = df_Naive, formula = Time ~ m + n)

ggplot( data = df_Naive ) +
  geom_line(
    aes(
      y = Time / ( m * fit_naive$coefficients[2] + fit_naive$coefficients[3] * n ),
      x = m * fit_naive$coefficients[2] + fit_naive$coefficients[3] * n + fit_naive$coefficients[1]
    ),
    color = "#000000", alpha = 0.9
  )
```



The following plot shows the runtime / regression of the Y axis and regression on X axis. Where the regression is: Time ~ n * m. This plot flatlines around y = 1, with random spikes of longer runtime which can be explained by background resource usage.

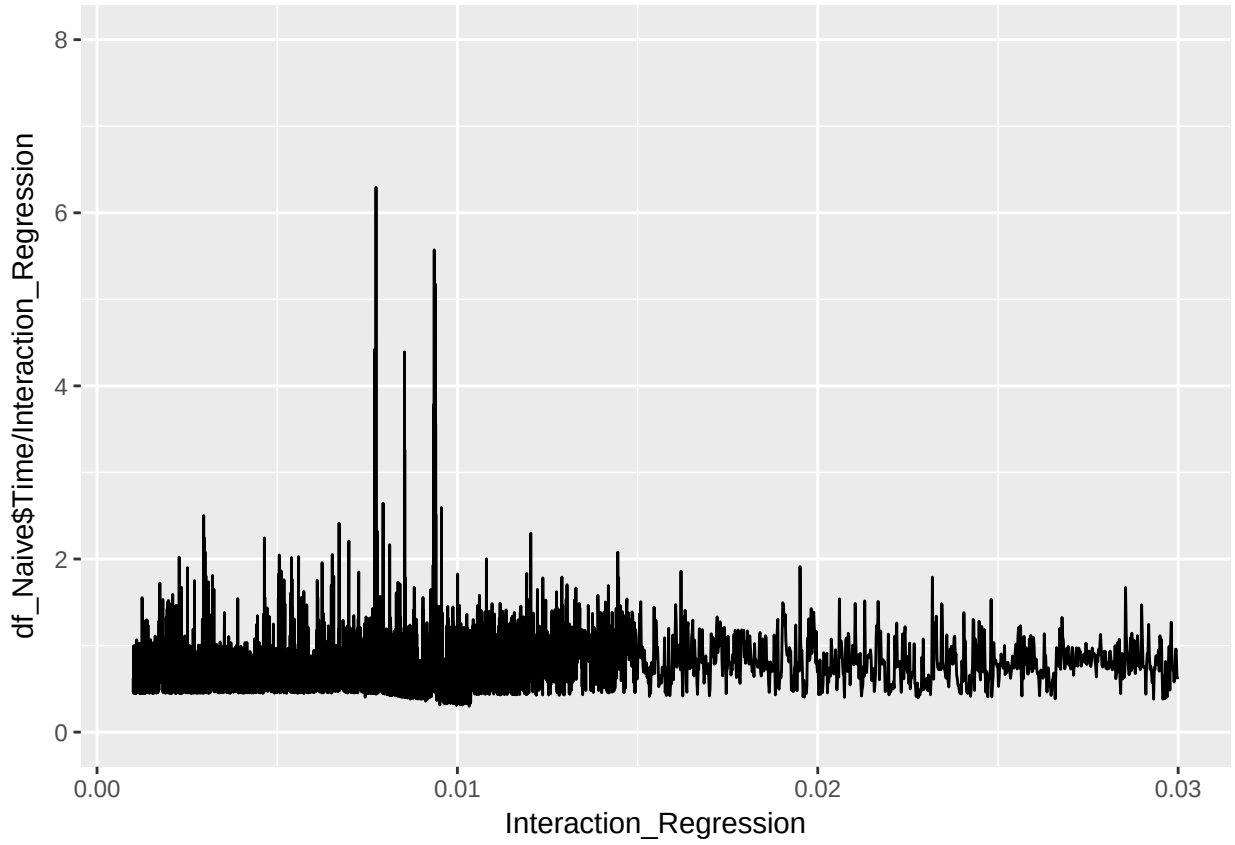
```
fit_naive_poly = glm( data = df_Naive, formula = Time ~ n * m )
fit_naive_poly
```

```
##
## Call:  glm(formula = Time ~ n * m, data = df_Naive)
##
## Coefficients:
## (Intercept)            n            m          n:m
##  -1.003e-03    7.440e-07   -2.348e-06    1.464e-10
##
```

```
## Degrees of Freedom: 5999 Total (i.e. Null);  5996 Residual
## Null Deviance:       0.2614
## Residual Deviance: 0.0761    AIC: -50610
```

```
inc = fit_naive_poly$coefficients[1]
c_n = fit_naive_poly$coefficients[2]
c_m = fit_naive_poly$coefficients[3]
c_nm = fit_naive_poly$coefficients[4]
Interaction_Regression = inc + c_n * df_Naive$n + c_m * df_Naive$m + df_Naive$n * df_Naive$m * c_nm
Interaction_Regression = 0 + c_n * df_Naive$n + 0 * df_Naive$m + df_Naive$n * df_Naive$m * c_nm

ggplot(  ) +
  geom_line(
    aes(
      y = df_Naive$Time / Interaction_Regression,
      x = Interaction_Regression
    )
  ) + xlim(0.001, 0.03) + ylim(0, 8)
```

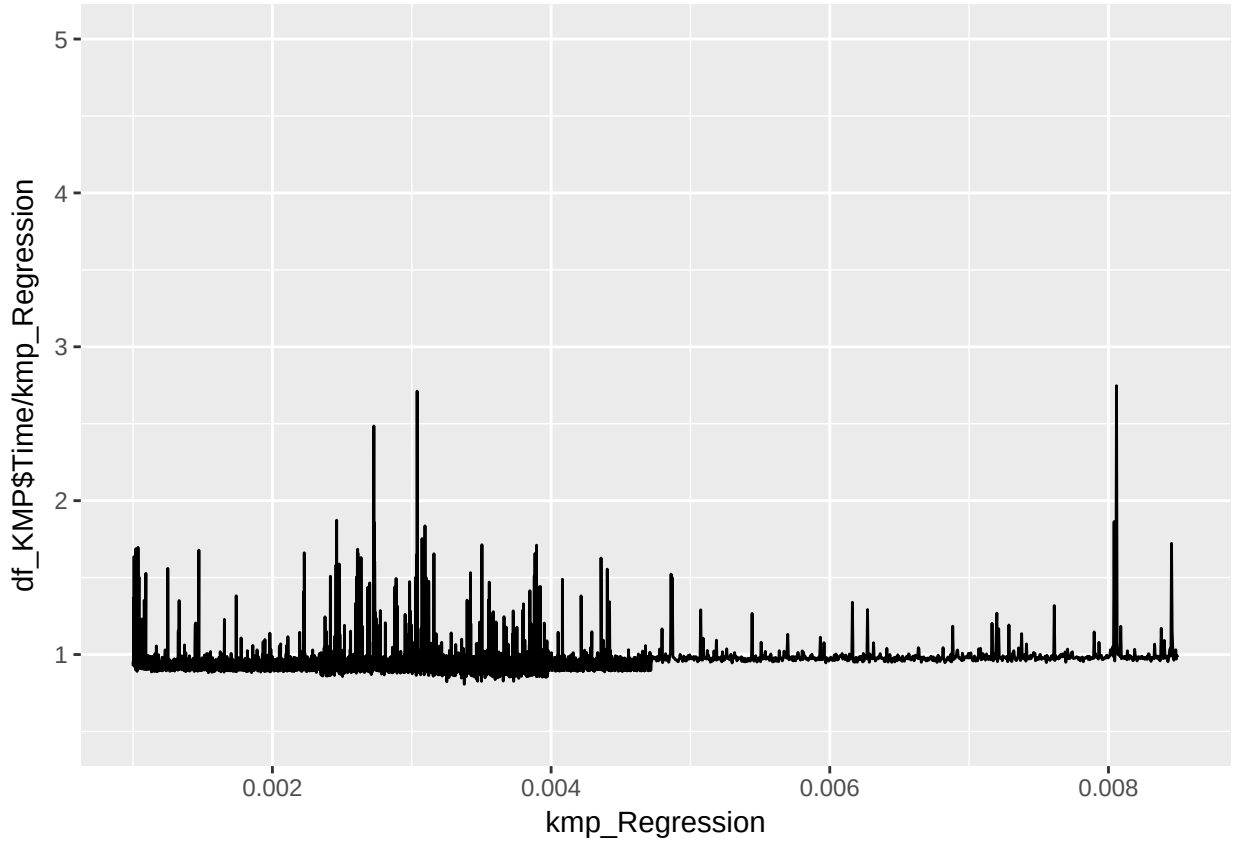


Runtime - KMP

The implementation of the KMP algorithm runs in O(n + m) time. The best case is a pattern with a simple failure function (all pointing to the base case), where the worst case keeps the inner loops busy by jumping with failures. The following plot shows the runtime / regression of the Y axis and regression on X axis. Where the regression is: Time ~ n + m. This plot flatlines around y = 1, with random spikes of longer runtime which can be explained by background resource usage.

```
fit_kmp = glm( data = df_KMP, formula = Time ~ n + m )
inc = fit_kmp$coefficients[1]
c_m = fit_kmp$coefficients[3]
c_n = fit_kmp$coefficients[2]
kmp_Regression = inc + c_n * df_KMP$n + c_m * df_KMP$m
kmp_Regression = 0 + c_n * df_KMP$n + c_m * df_KMP$m

ggplot(  ) +
  geom_line(
    aes(
      y = df_KMP$Time / kmp_Regression,
      x = kmp_Regression
    )
  ) + xlim(0.001, 0.0085) + ylim(0.5, 5)
```



Testdata and Number of occurrences

Tests of runtime was done the same way for both algorithms. First varying m and keeping n the same. Second keepoing m the same and varying n. Third by varying both n and m together.

The number of occurrences of the pattern in the string has no significant impact on the naive implementation. Here is the p-value of an ANOVA of the addition to the regression.

```
fit_naive = lm( data = df_Naive, formula = Time ~ n + m )
fit_naive_z = lm( data = df_Naive, formula = Time ~ n + m + z )
```

```r
an_naive = anova(fit_naive, fit_naive_z)
an_naive$`Pr(>F)`
```

```
## [1]        NA 0.4574749
```

The number of occurrences of the pattern in the string has no significant impact on the KMP implementation. Here is the p-value of an ANOVA of the addition to the regression.

```r
fit_kmp = lm( data = df_KMP, formula = Time ~ n + m )
fit_kmp_z = lm( data = df_KMP, formula = Time ~ n + m + z )

an_kmp = anova(fit_kmp, fit_kmp_z)
an_kmp$`Pr(>F)`
```

```
## [1]        NA 0.8632311
```