Proyecto Final

Rafael Puche

06 de julio de 2016

Buscando Genes en genomas bacterianos

El objetivo principal es extraer ORF (Open Reading Frames) en genomas de Leptospira, y predecir posibles candidatos a genes. Usando una aproximacion de Machine Learning No-supervisado

Notas Teoricas

La secuencia: **ACGTACGTACGTACGT** puede ser agrupada en codones como se observa a continuacion:

1. ACG-TAC-GTA-CGT-ACG-T...
2. A-CGT-ACG-TAC-GTA-CGT...
3. AC-GTA-CGT-ACG-TAC-GT...

La traduccion en aminoacidos y la proteina resultante pueder ser completamente diferente en cada uno de los 3 casos. Un *Reading Frame* son agrupaciones en codones con una posicion especifica de inicio.

```
library (seqinr)
tablecode()
```

Genetic code 1 : standard

T T T	Phe	T C T	Ser	T A T	Tyr	T G T	Cys
T T C	Phe	T C C	Ser	T A C	Tyr	T G C	Cys
T T A	Leu	T C A	Ser	T A A	Stp	T G A	Stp
T T G	Leu	T C G	Ser	T A G	Stp	T G G	Trp
C T T	Leu	C C T	Pro	C A T	His	C G T	Arg
C T C	Leu	C C C	Pro	C A C	His	C G C	Arg
C T A	Leu	C C A	Pro	C A A	Gln	C G A	Arg
C T G	Leu	C C G	Pro	C A G	Gln	C G G	Arg
A T T	Ile	A C T	Thr	A A T	Asn	A G T	Ser
A T C	Ile	A C C	Thr	A A C	Asn	A G C	Ser
A T A	Ile	A C A	Thr	A A A	Lys	A G A	Arg
A T G	Met	A C G	Thr	A A G	Lys	A G G	Arg
G T T	Val	G C T	Ala	G A T	Asp	G G T	Gly
G T C	Val	G C C	Ala	G A C	Asp	G G C	Gly
G T A	Val	G C A	Ala	G A A	Glu	G G A	Gly
G T G	Val	G C G	Ala	G A G	Glu	G G G	Gly

Extraccion de ORF de un genoma en archivo FASTA

En este punto se emplearon tres aproximaciones para extraer los ORF:

1. Una funcion en R, llamada `findORFsinSeq` que incluye el uso de la libreria *seqinr* y *Biostrings*
2. El uso de la funcion *long-orfs* que esta incluida en el programa **Glimmer** *(Gene Locator and Interpolated Markov ModelER)* el cual esta diseñado especificamente para la prediccion de genes en bacterias con una precision del 97%, este modelo utiliza cadenas de Markov.

3. El programa **ORFfinder** del NCBI el cual en su version de linea de comandos puede buscar ORF en secuencias de ADN mayores a 50 K

```
#### Cargar genoma FASTA
#setwd("/home/rpuchequin/MEGA_Maestria/Machine_Learning_IVIC/Proyecto_final_data/Prediccion_ORF_Lepto/"
#Lepto_1 <- read.fasta(file = "Lepto_1_assembly.fasta")`
### Sacar el contig de mayor tamaño y guardarlo en otro vector
#Lepto_1_big <- Lepto_1[[11]]
### Convertir los caracteres individuales en una sola cadena de nucleotidos
#Lpt_big <- c2s(Lepto_1_big)
### Guardar el contig si es necesario
# write.fasta(sequences = L, names = names(L), nbchar = 80, file.out = "ContigGrande.fasta")
```

Resultado del *long-orf*

Linea de comando para correr el programa: `./long-orfs -A -f -z 11 Lepto_Bratislava_strain_PigK151.fasta glimmer_ORF_Lepto_B`

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:seqinr':
##
##     count, query

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

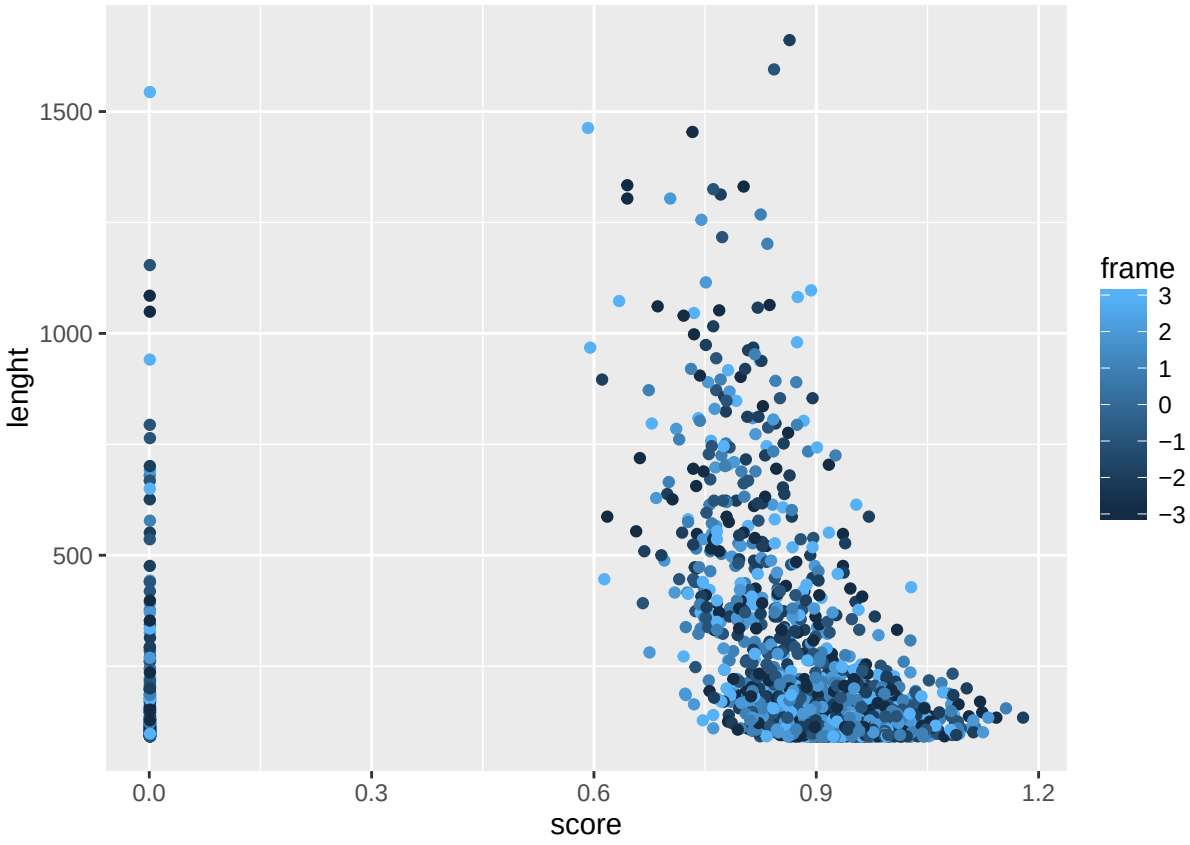
```
library(ggplot2)
```

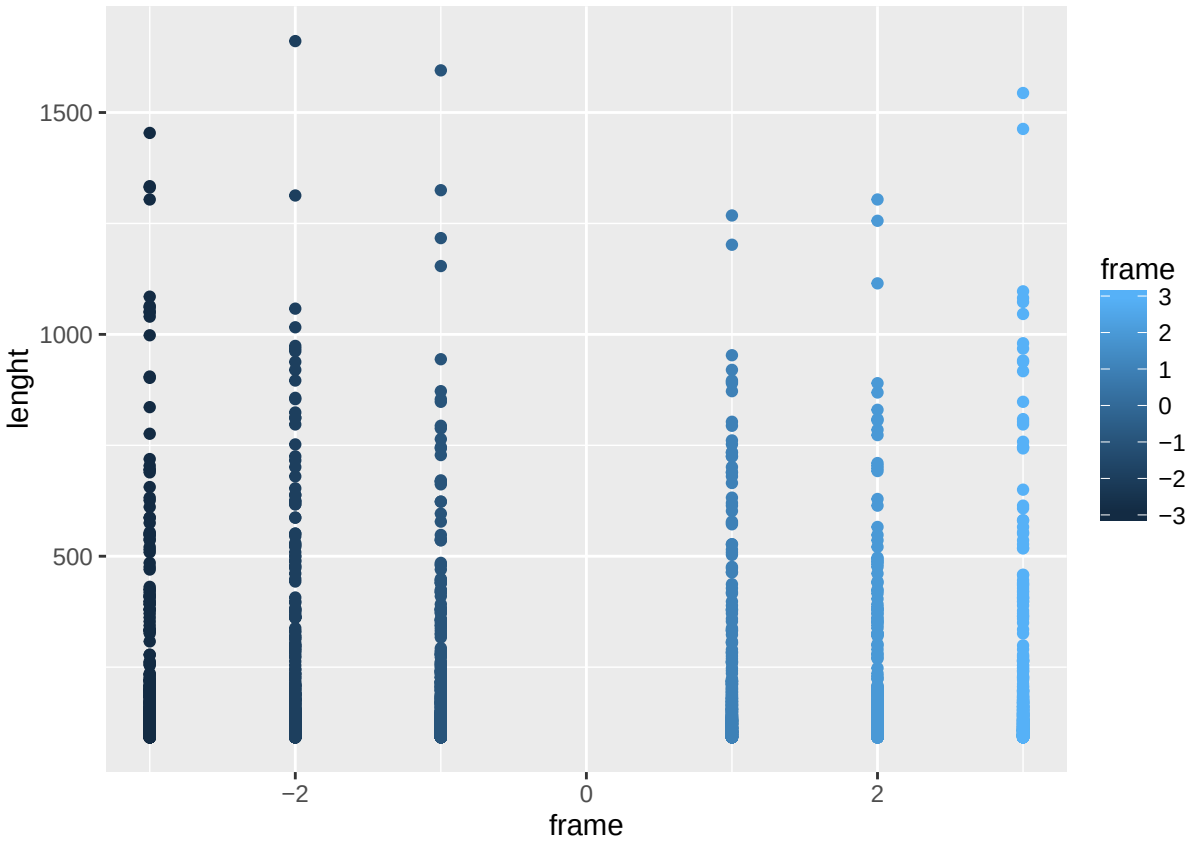
```
Data_Glimmer <- read.csv("glimmer_ORF_Lepto_B.csv")
Data_Glimmer <- mutate(.data = Data_Glimmer, score=score/1000)
Data_Glimmer <- mutate(.data = Data_Glimmer, lenght=abs(lenght))
summary(Data_Glimmer)
```

```
##  Genes_Putativos    star_orf          stop_orf           frame
##  Min.   :   1    Min.   :    623   Min.   :    724   Min.   :-3.000000
##  1st Qu.: 568    1st Qu.:1072966   1st Qu.:1073091   1st Qu.:-2.000000
##  Median :1135    Median :2232045   Median :2231953   Median : 1.000000
##  Mean   :1135    Mean   :2227467   Mean   :2227461   Mean   :-0.005289
##  3rd Qu.:1702    3rd Qu.:3419526   3rd Qu.:3419657   3rd Qu.: 2.000000
##  Max.   :2269    Max.   :4365984   Max.   :4366076   Max.   : 3.000000
##      score            lenght
```

```
##  Min.   :0.00064   Min.   :  92.0
##  1st Qu.:0.84500   1st Qu.: 104.0
##  Median :0.91900   Median : 128.0
##  Mean   :0.82890   Mean   : 206.1
##  3rd Qu.:0.96600   3rd Qu.: 191.0
##  Max.   :1.17900   Max.   :1661.0
```

Revisando la data





Funcion para normalizar columnas

```
normalize <- function(x) {
     return ((x - min(x)) / (max(x) - min(x)))
}
```

Aplicar normalizacion

```
Data_Glimmer <- mutate(.data = Data_Glimmer, lenght=normalize(lenght))
summary(Data_Glimmer$lenght)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.000000 0.007648 0.022940 0.072740 0.063100 1.000000
```

```
Data_Glimmer <- mutate(.data = Data_Glimmer, score=normalize(score))
summary(Data_Glimmer$score)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.7166  0.7794  0.7029  0.8192  1.0000
```

Evaluando nuestro modelo

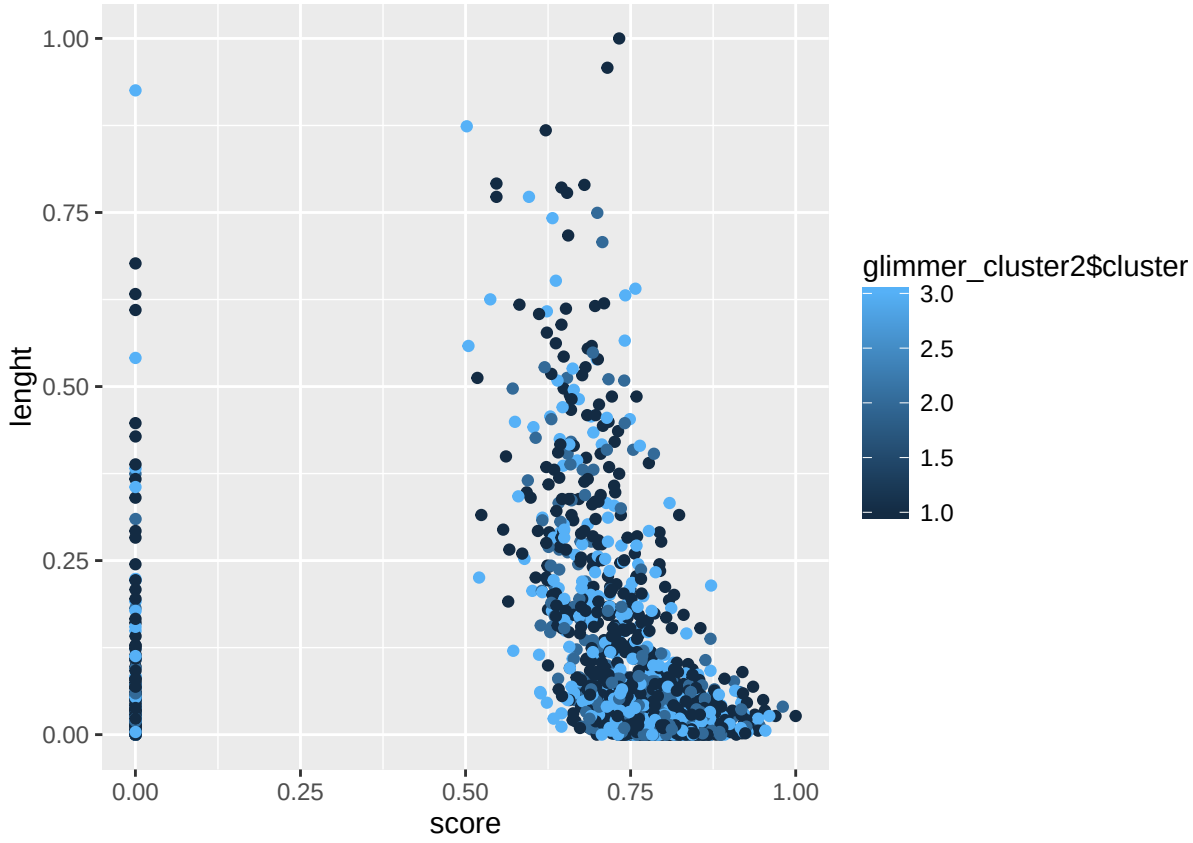
```
glimmer_cluster2 <- kmeans(Data_Glimmer[, 4:6], 3, nstart = 50)
table(glimmer_cluster2$cluster)
```

```
##
##    1    2    3
## 1124  394  751
```

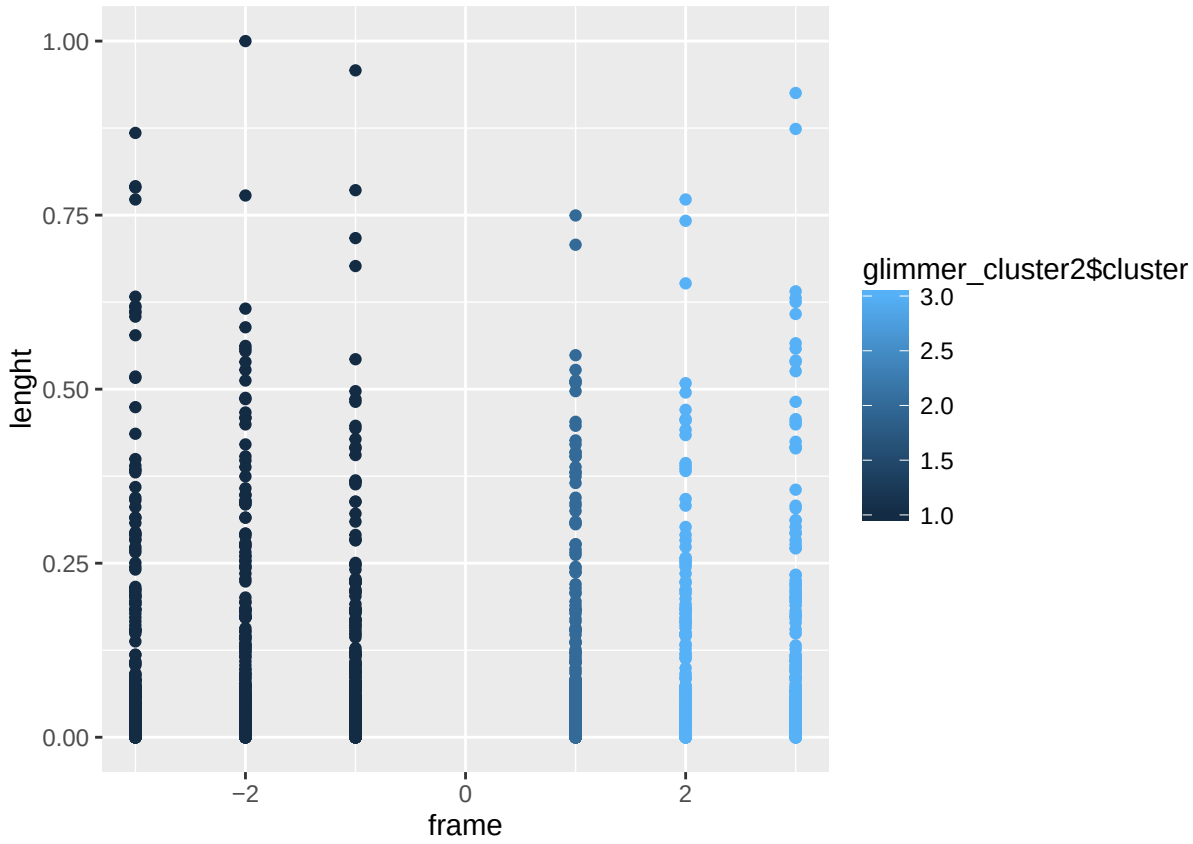
```
glimmer_cluster2$centers
```

```
##       frame     score     lenght
## 1 -2.031139 0.6982213 0.07804856
## 2  1.000000 0.7121552 0.06831924
## 3  2.499334 0.7050288 0.06711256
```

```
ggplot(Data_Glimmer, aes(score, lenght, color=glimmer_cluster2$cluster)) + geom_point()
```




```
ggplot(Data_Glimmer, aes(frame, lenght, color=glimmer_cluster2$cluster)) + geom_point()
```



Mas Grupos

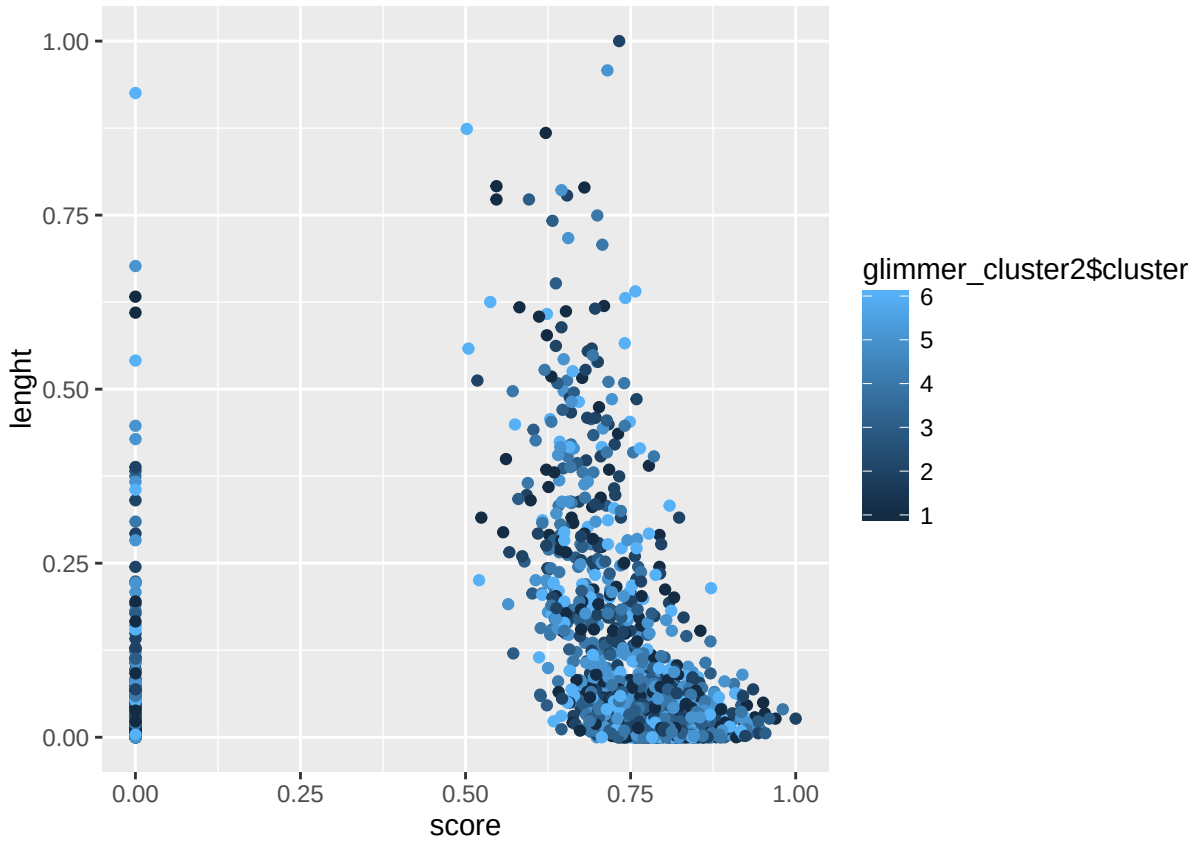
```
glimmer_cluster2 <- kmeans(Data_Glimmer[, 4:6], 6, nstart = 50)
table(glimmer_cluster2$cluster)
```

```
##
##   1   2   3   4   5   6
## 386 387 376 394 351 375
```

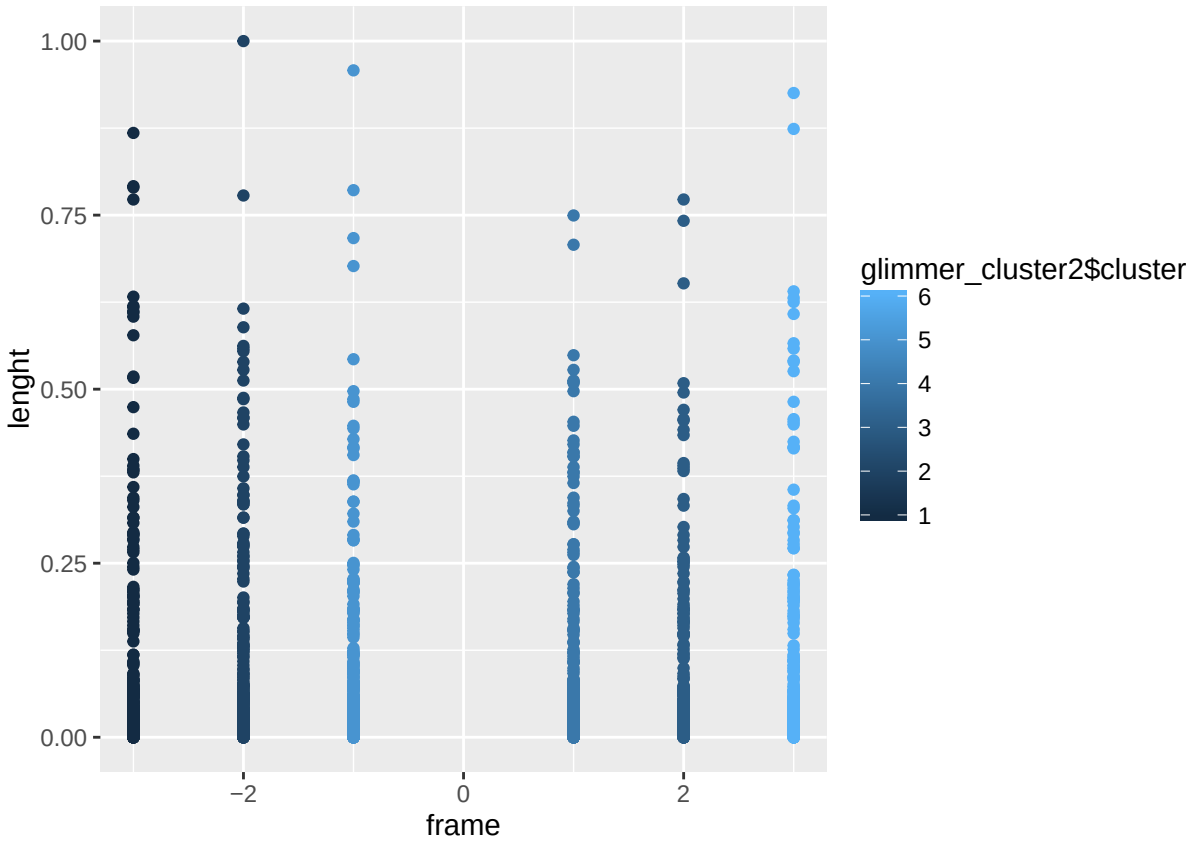
```
glimmer_cluster2$centers
```

```
##   frame      score     lenght
## 1    -3 0.6943871 0.07647688
## 2    -2 0.7004815 0.08118043
## 3     2 0.6970368 0.06107868
## 4     1 0.7121552 0.06831924
## 5    -1 0.6999460 0.07632386
## 6     3 0.7130420 0.07316252
```

```
ggplot(Data_Glimmer, aes(score, lenght, color=glimmer_cluster2$cluster)) + geom_point()
```




```
ggplot(Data_Glimmer, aes(frame, lenght, color=glimmer_cluster2$cluster)) + geom_point()
```


1.00
0.75
0.50
0.25
0.00
lenght
−2
0
2
frame
glimmer_cluster2$cluster
6
5
4
3
2
1