Flexible Logo plots of alphanumeric strings and symbols using *Logolas*

Kushal K Dey

Stephens Lab, The University of Chicago

*`Correspondending Email: kkdey@uchicago.edu`

November 5, 2016

Abstract

Logo plots are popular in genomic studies for sequence alignment and motif detection. However, logos have been pretty restrictive in its scope because of limited library of symbols it uses and the lack of flexibility in extending it to other applications. In this package, we provide an easy and more flexible interface for the user to plot logos and more importantly, we extend the library of logos from A, C, T, G (as in seqLogo) and English alphabets (RWebLogo) to include numbers and alpha-numeric strings with provision for punctuations and arrows. It also provides the user with a simple interface to create her own logo and add to her personal library. In this vignette, we discuss a number of applications beyond sequence alignment where such flexible logo plots can be used.

Logolas version: 0.1.0 [1]

[1] This document used the vignette from *Bioconductor* package *CountClust, DESeq2* as *knitr* template

Contents

1 Introduction

Logo plots are a popular tool in bioinformatics and regulatory genomics studies for representing sequence alignment patterns and also sequence and protein motif detection. One of the first logo plotting tools is *seqLogo* by Oliver Bembom [**?**], specifically targeted at DNA sequence alignment. However, it has a library of only 4 symbols - A, C, G and T- corresponding to the 4 nucleotides. The package *RWebLogo* is an extension of WebLogo python package that plots custom sequence logos by extending it to all alphabets. Another package *motifStack* works with both DNA/RNA sequence motif and amino acid sequence motif and customizes font size and colors.

Logolas adds more flexibility by customizing logos and the graphical design of the logo plots. Also it extends the library of symbols beyond English alphabets to numbers, symbols (arrows, punctuations) and to alphanumeric strings. It allows the user to choose a range of information criteria to determine logo sizes and more importantly, a simple user interface to create new logos and add them to the library of logos already present and to strings. We show several applications in genomics, ecology and document mining, where such logo plots can be applied.

2 Logolas Installation

Logolas requires the following CRAN-R package : *grid*, *gridExtra*, *RColorBrewer*, *devtools*.

```
source("http://bioconductor.org/biocLite.R")
biocLite("Logolas")
```

For the developmental version on Github, one can use

```
install_github('kkdey/Logolas')
```

Then load the package with:

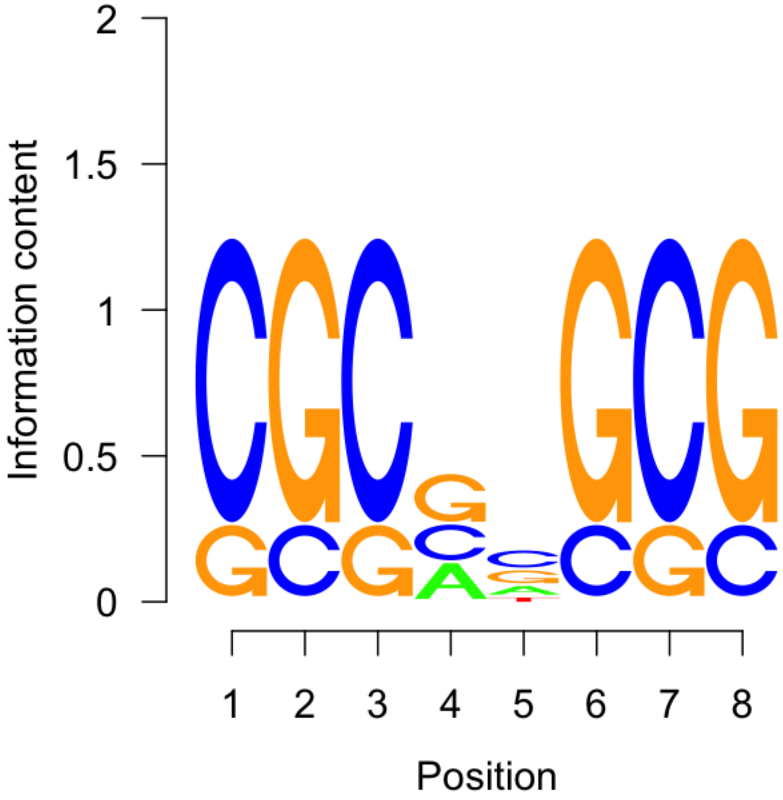
```
library(Logolas)
```

3 Application

We start with the most basic application of logo plots - for alignment of DNA sequence, comprising of logos A, C, T and G, corresponding to the four nucleotide. This is the typical application of *seqLogo*. We start with a demo example and apply *seqLogo* to it.

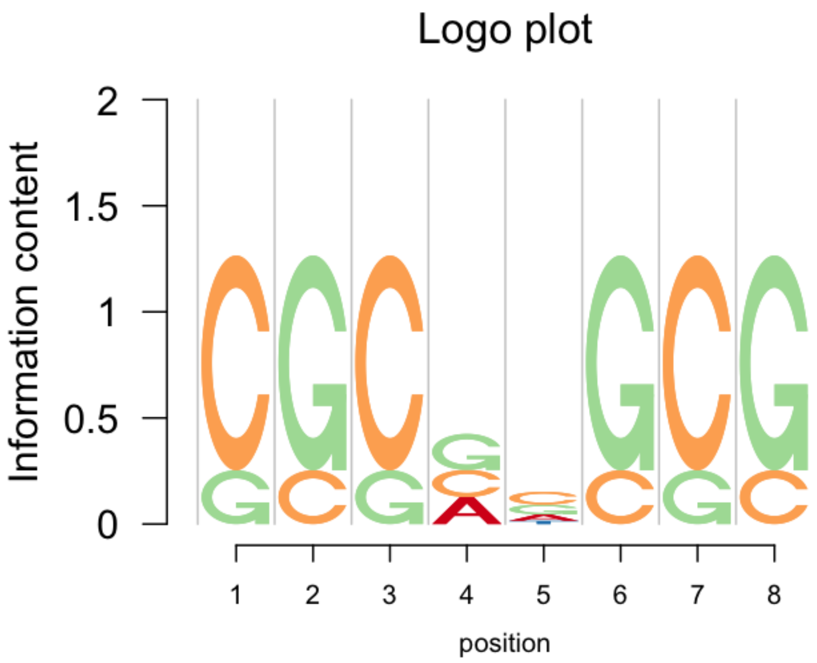
The user first needs to make a position weight matrix from the matrix using the `makePWM()` function. Then it uses the `seqLogo()` function on the output to plot the logo plots.

```
library(seqLogo)
mFile <- system.file("Exfiles/pwm1", package="seqLogo")
m <- read.table(mFile)
p <- makePWM(m)
seqLogo(p)
```



Now we apply *Logolas* to build similar plot. We start with the same matrix as in *seqLogo*, and assign row names and column names that will be used in the plot as symbols and block labels respectively.

```
rownames(m) <- c("A", "C", "G", "T")
colnames(m) <- 1:8
logomaker(m,
          cols= RColorBrewer::brewer.pal(dim(m)[1],name ="Spectral"),
          frame_width = 1,
          ic.scale = TRUE,
          yscale_change=FALSE,
          xlab="position",
          col_line_split = "grey80")
```

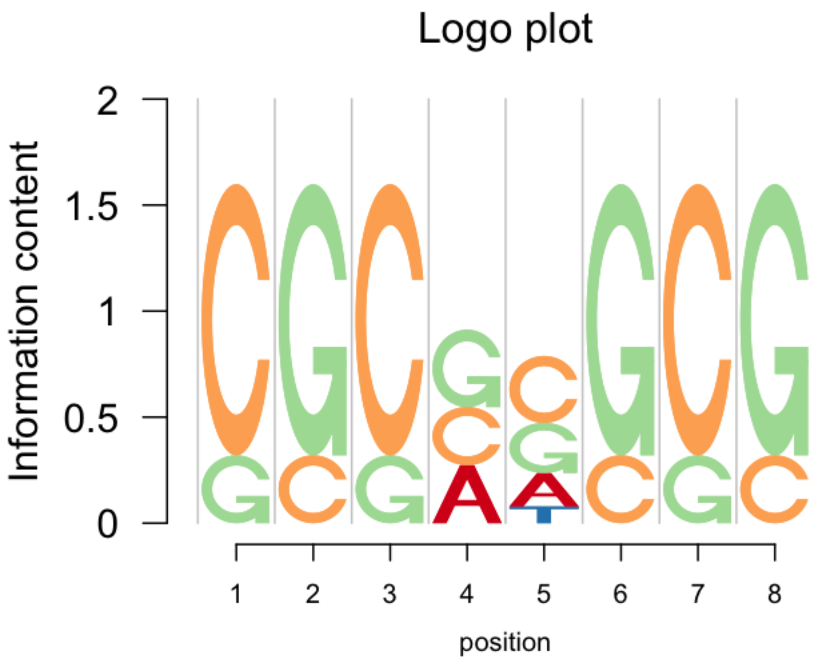


As default, if `ic.scale=` TRUE , the heights of the bars at each position are determined by the Shannon entropy. The size of logo in each stack is proportional to the relative abundance of that logo in that stack. To change to other orders of Renyi entropy, one can tune the input parameter `alpha`.

A higher value of `alpha` makes the logos more prominent, besides maintaining relative structure.

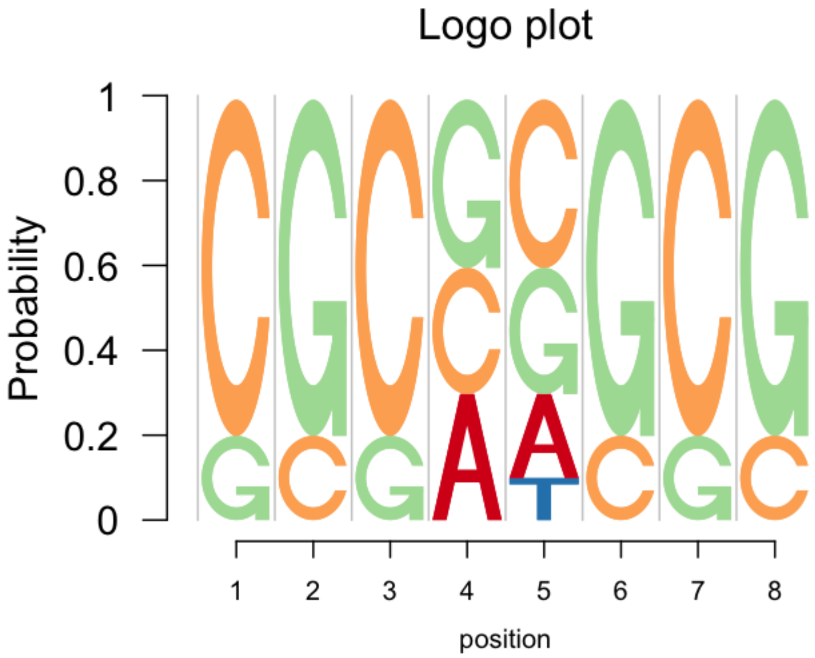
Also, the Y-axis can be adjusted by taking `yscale_change=TRUE`

```
rownames(m) <- c("A", "C", "G", "T")
colnames(m) <- 1:8
logomaker(m,
          cols= RColorBrewer::brewer.pal(dim(m)[1],name ="Spectral"),
          frame_width = 1,
          ic.scale = TRUE,
          alpha = 2,
          yscale_change=FALSE,
          xlab="position",
          col_line_split = "grey80")
```



One can also use normalized heights of the stacks of logos for each column by choosing `ic.scale=` FALSE .

```r
rownames(m) <- c("A", "C", "G", "T")
colnames(m) <- 1:8
logomaker(m,
          cols= RColorBrewer::brewer.pal(dim(m)[1],name ="Spectral"),
          frame_width = 1,
          ic.scale = FALSE,
          alpha = 2,
          xlab="position",
          col_line_split = "grey80")
```


Logo plot
Probability
position