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:

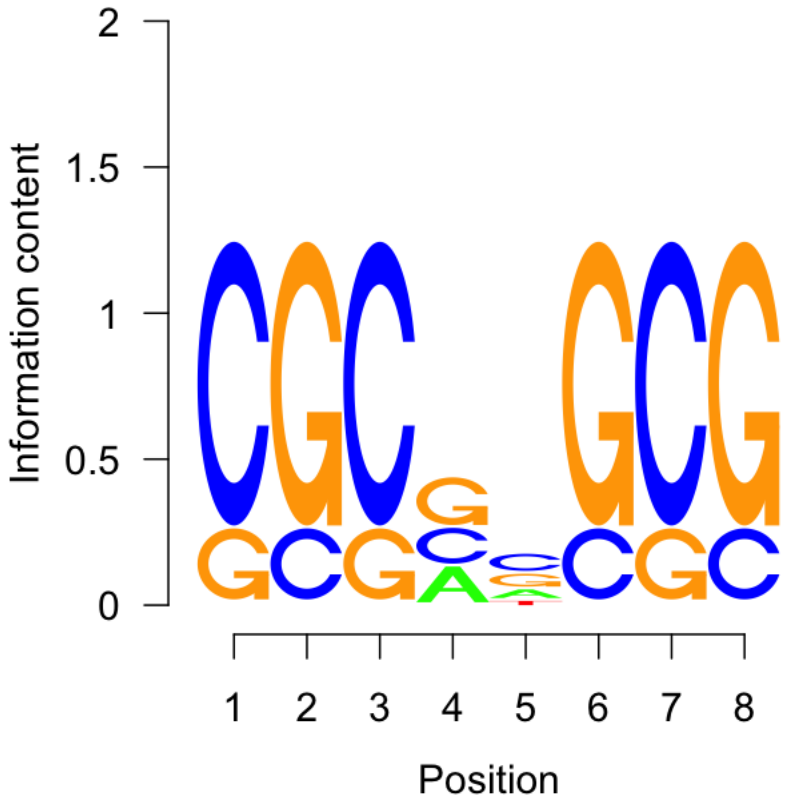
```
library(Logolas)
```

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.

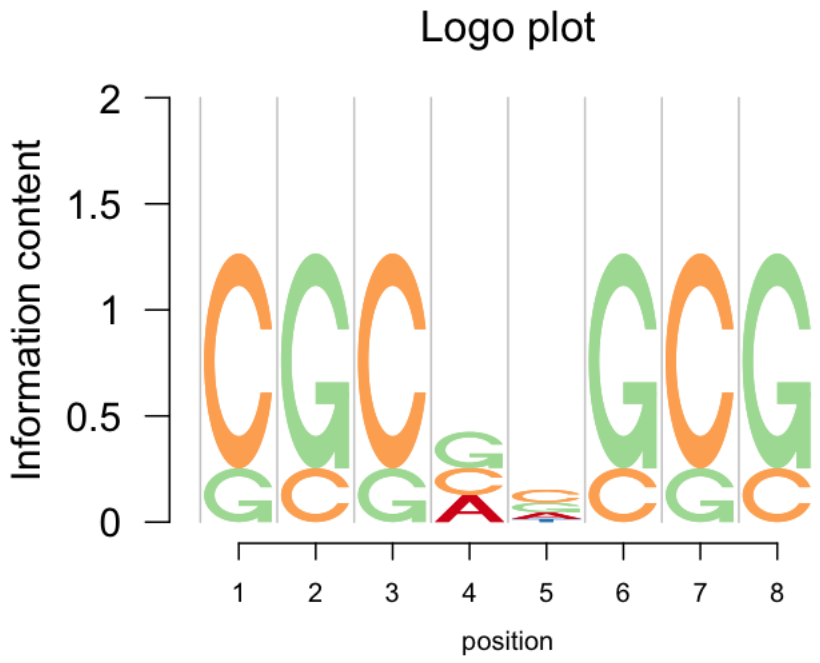
The user first needs to make a position weight matrix from the matrix using the `makePWM()` function. Then it uses the `seqLogo()` function on the output to plot the logo plots.

```
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.

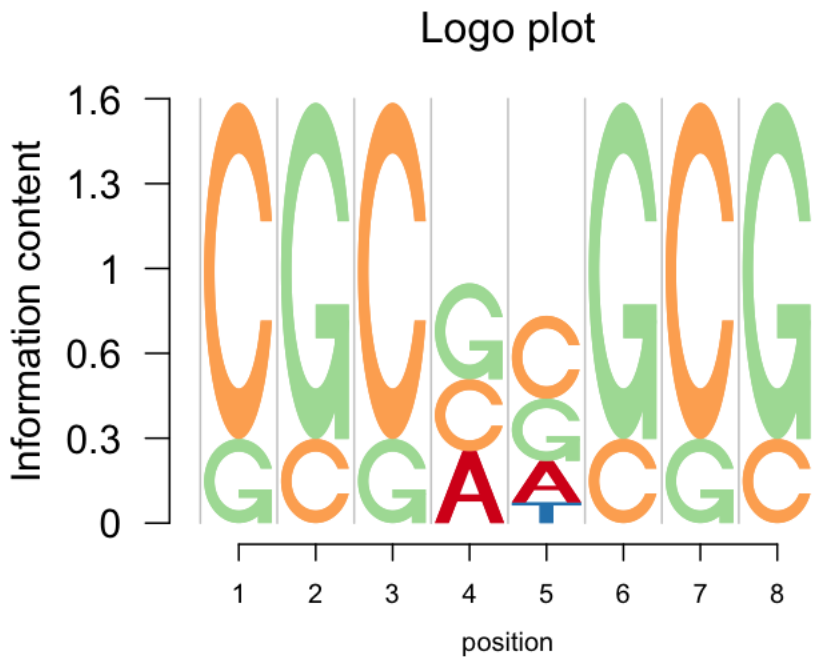
To change to other orders of Renyi entropy, one can tune the input parameter `alpha`. A higher value of `alpha` makes the logos more prominent, besides maintaining relative structure. One can set different infromation criteria using the `ic_computer` function.

```
ic_computer(m, alpha=3)
```

```
## [1] 1.673 1.673 1.673 0.931 0.849 1.673 1.673 1.673
```

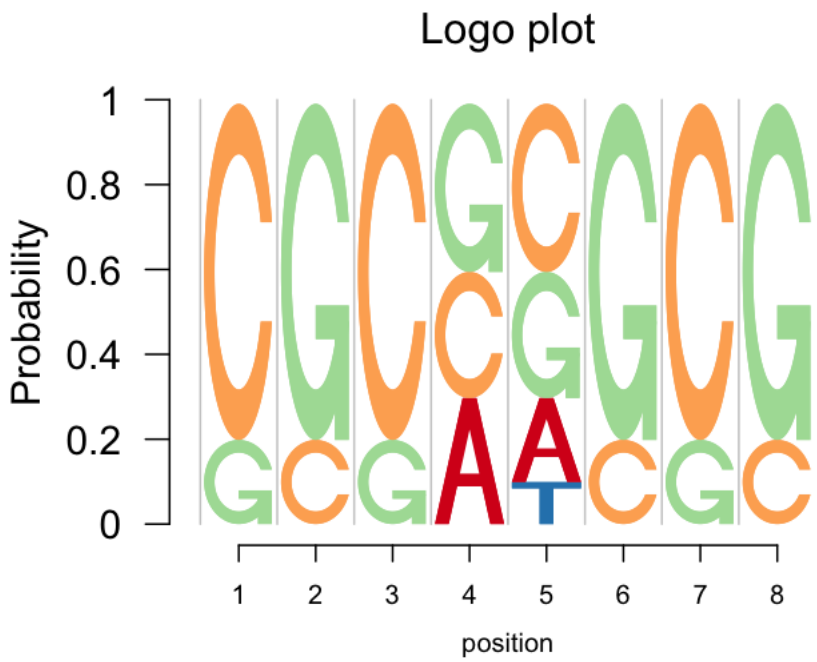
Also, the Y-axis can be adjusted by taking `yscale_change=TRUE`

```
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=TRUE,
          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 .

```
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",
          ylab = "Probability")
```



Logolas lets you customize the colors of the logos, has option for user-defined information function under option `ic`, beyond the different Renyi criteria that can be set. User can set titles, X-labels, Y-labels, axis ticks and also the relative width of each column of the logo stack, making it much more flexible than the standard packages for logo plotting.

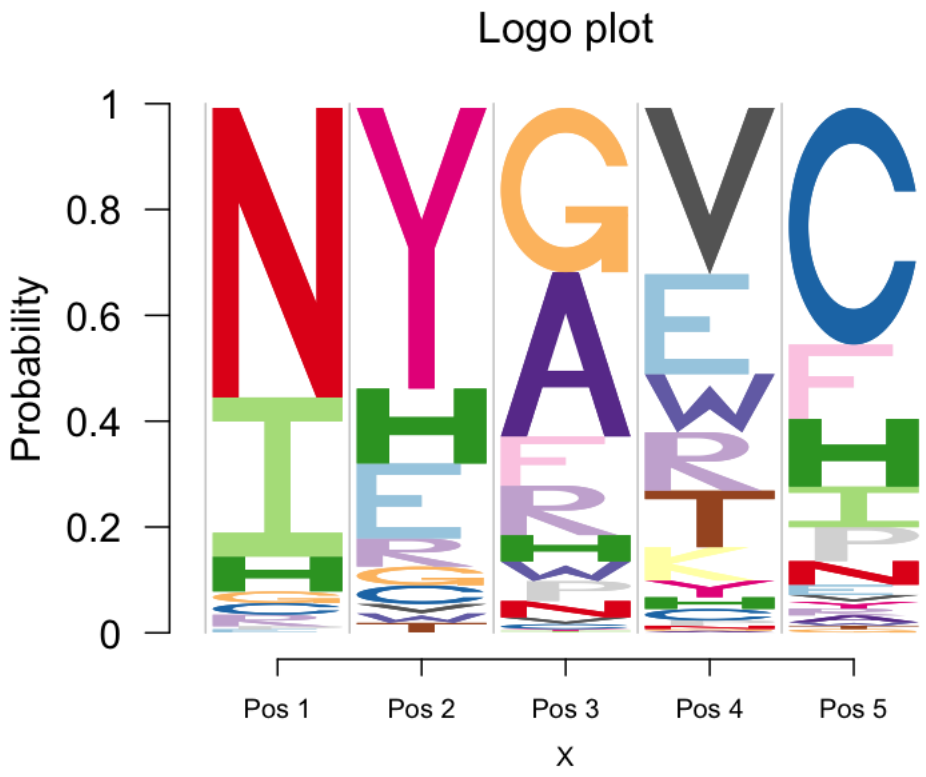
3.1 Amino acid sequence motif

One can use *Logolas* for amino acid sequence motif detection as well, as the logo library of the software includes all the English alphabets and the 20 amino acids have a 1-letter representation on English alphabets.

```
counts_mat <- rbind(c(0, 0, 100, 1, 2), c(4, 3, 30, 35, 2),
                    c(100, 0, 10, 2, 7),rep(0,5),
                    c(4, 2, 3, 7, 70), c(1, 8, 0, 60, 3),
                    rep(0, 5), c(4, 2, 100, 1, 1),
                    c(12, 8, 16, 7, 20), c(55, 0, 1, 0, 12),
                    rep(0,5), c(rep(0,3), 20, 0),
                    rep(0,5), c(0, 0, 30, 0, 22),
                    c(1, 0, 12, 3, 10), rep(0,5),
                    c(0, 1, 0, 34, 1), c(0, 1, 12, 35, 1),
                    c(0, 30, 1, 10, 2), c(0, 1, 4, 100, 2))
```

Note that all we needed to do to get the logo plots is to say the row names and column names as per the the logos and the stack labels and then fix the colors for the logos.

```
rownames(counts_mat) <- c("A", "R", "N", "D","C", "E", "Q", "G",
                          "H", "I", "L", "K", "M", "F", "P", "S",
                          "T", "W", "Y", "V")

colnames(counts_mat) <- c("Pos 1", "Pos 2", "Pos 3", "Pos 4", "Pos 5")

cols1 <- c(rev(RColorBrewer::brewer.pal(12, "Paired"))[c(3,4,7,8,11,12,5,6,9,10)],
           RColorBrewer::brewer.pal(12, "Set3")[c(1,2,5,8,9)],
           RColorBrewer::brewer.pal(9, "Set1")[c(9,7)],
           RColorBrewer::brewer.pal(8, "Dark2")[c(3,4,8)])

logomaker(counts_mat,
          cols= cols1,
          frame_width = 1,
          ic.scale  = FALSE,
          yscale_change = FALSE)
```



3.2 String Logos: Mutation profiling

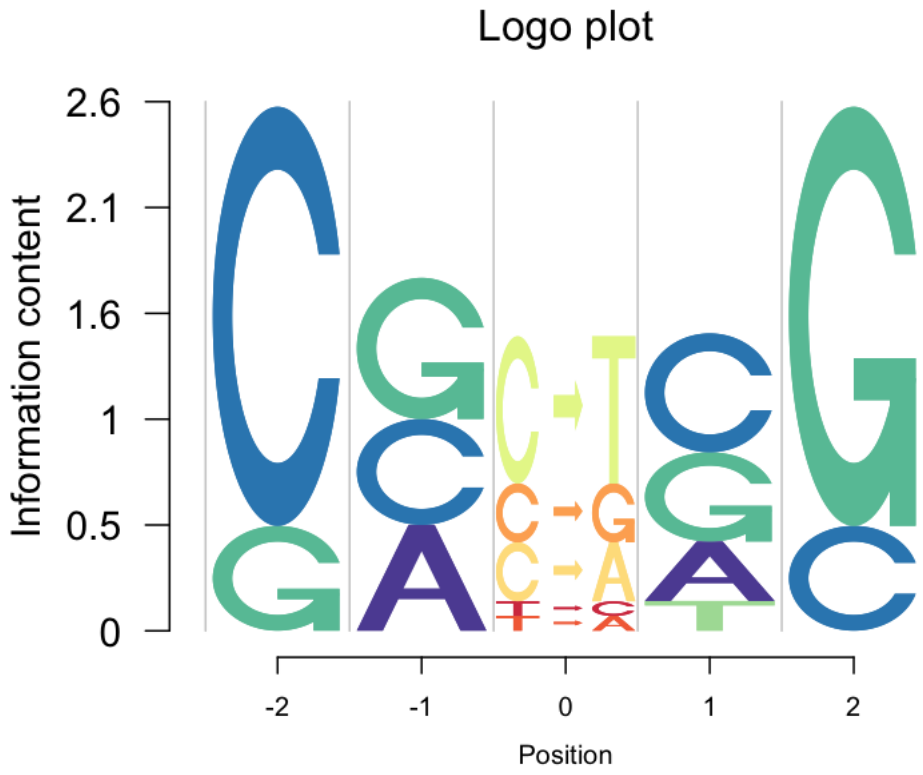
We now step beyond just alphabets as logos, and present the first example of how a string as logos. Suppose for a set of cell lines, we are provided data on the number of mutations (nucleotide substitions) and the corresponding nucleotides along the flanking bases of the nucleotide substitution. This is the kind of problem addressed by Shiraishi et al 2015 [**?**]. Shiraishi et al have a package `pmsignature` for plotting the substitution and flanking bases profile, but here we use logos to profile them. We apply it on a demo example.

```
library(seqLogo)
mFile <- system.file("Exfiles/pwm1", package="seqLogo")
m <- read.table(mFile)
p <- makePWM(m)
mat1 <- cbind(p@pwm[,c(3,4)], rep(0,4), p@pwm[,c(5,6)]);
colnames(mat1) <- c("-2", "-1", "0", "1", "2")
mat2 <- cbind(rep(0,6), rep(0,6),
              c(0.5, 0.2, 0.2, 0.05, 0.05, 0),
              rep(0,6), rep(0,6))
rownames(mat2) <- c("C>T", "C>A", "C>G",
                    "T>A", "T>C", "T>G")

table <- rbind(mat1, mat2)
```

Note that we use the symbols `X>Y` to depict $X \rightarrow Y$ substitutions. The data contains proportion of logos in each position - -2 left flanking, -1 left flanking, mutation, 1 right flanking and 2 right flanking. Note that `X>Y` type symbols occur only in the middle stack as that is the mutation stack, while the nucleotides A, C, T and G occur only in the left two and right two flanking bases stacks.

Then we apply `logomaker` on that matrix.

```
logomaker(table,
          cols= rev(RColorBrewer::brewer.pal(dim(table)[1],
          name = "Spectral")),
          frame_width = 1,
          ic.scale = TRUE,
          yscale_change=TRUE,
          xlab = "Position",
          ylab = "Information content")
```



3.3 String Logos: Histone marks patterns

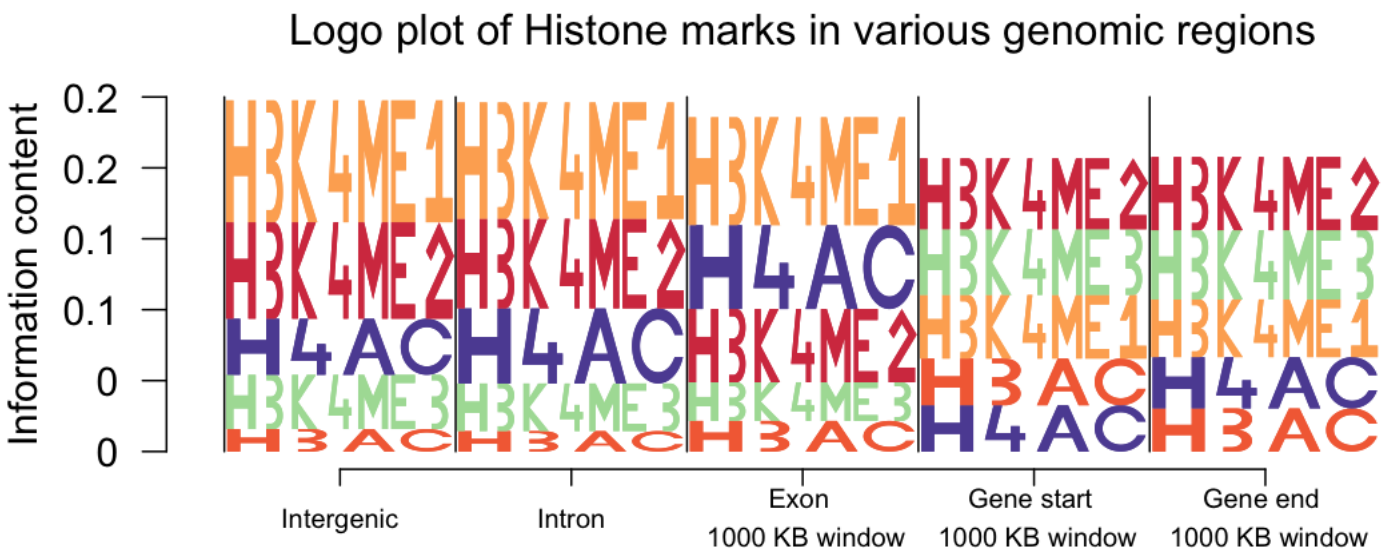
In studies relted to histone marks, one might be interested to see if certain histone marks are prominent than others in some cell lines or tissues or in some genomic regions. In this case, we apply Logolas on an example data from Koch et al (2007) [Supp Table 2 of that paper]. The authors recorded number of histone modification sites identified by the HMM which overlap with an intergenic sequence, intron, exon, gene start and gene end for the lymphoblastoid cell line, GM06990, in the ChIP-CHIP data. Logolas provides a handy visualization to see how the patterns of histone modification sites changes across genomic region types for that cell line.

First we input the data from Supp Table 2 due to Koch et al (2007).

```
mat <- rbind(c(326, 296, 81, 245, 71),
             c(258, 228, 55, 273, 90),
             c(145, 121, 29, 253, 85),
             c(60, 52, 23, 180, 53),
             c(150, 191, 63, 178, 63))

rownames(mat) <- c("H3K4ME1", "H3K4ME2", "H3K4ME3", "H3AC", "H4AC")
colnames(mat) <- c("Intergenic","Intron","Exon \n 1000 KB window",
                   "Gene start \n 1000 KB window","Gene end \n 1000 KB window")
```

Note here that the histone mark symbols are alphanumeric, for example $H3K4ME1$. We now apply *Logolas* on this data.

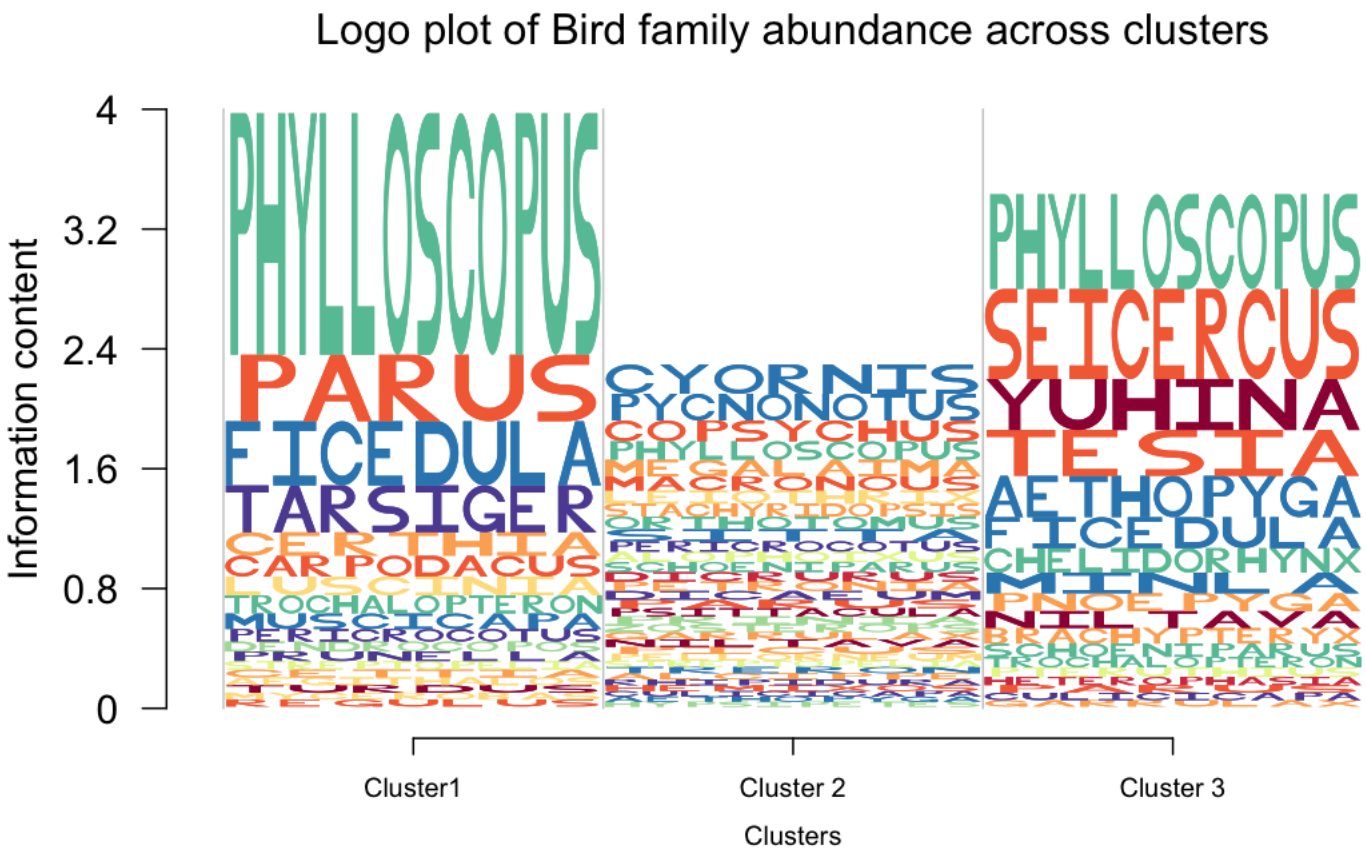
```
logomaker(mat,
          cols= sample(RColorBrewer::brewer.pal(10,name = "Spectral"), dim(mat)[1]),
          frame_width = 1,
          ic.scale = TRUE,
          pop_name = "Histone marks in various genomic regions",
          xlab = "",
          ylab = "Information content",
          yscale_change = TRUE,
          col_line_split = "black")
```



3.4 String Logos: Ecological Data

String logos can be used to represent how families or genus vary between sites for ecological or metagenomic data. In this case, we present an example of abundance patterns of different families of birds in three clusters of regions. The bird family names then act as logos and the clusters of regions represent the stacks in the logo plot.

```
set.seed(100)
data("himalayan_fauna_3_clusters")
logomaker(himalayan_fauna_3_clusters,
          cols= sample(RColorBrewer::brewer.pal(10,name = "Spectral"),
                       dim(himalayan_fauna_3_clusters)[1], replace=TRUE),
          frame_width = 1,
          ic.scale = TRUE,
          pop_name = "Bird family abundance across clusters",
          xlab = "Clusters",
          ylab = "Information content")
```



3.5 String Logos: Document mining

So far we mainly focused on biology examples where *Logolas* can be applied. But, it has application beyond the field of biology. In fact, *Logolas* can be used as an alternative to divided bar charts and pie charts. One example application of *Logolas* is in document mining and presenting keywords or tags as logos.

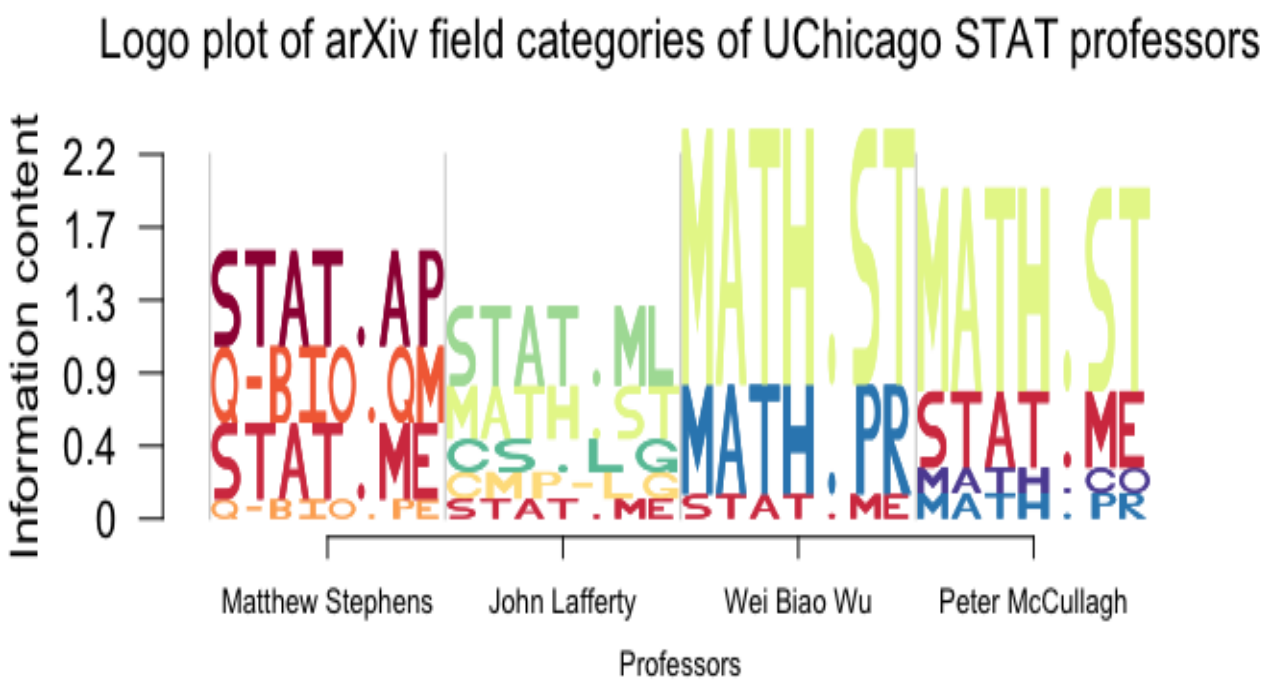
Here we build a logo plot of the field categories of manuscipts submitted by 4 professors from Dept. of Statistics, University of Chicago, on aRxiv.Note here the field categories are a combination of numbers and alphabets and dots and dashes.

We first generate the data

```
library(aRxiv)
rec1 <- arxiv_search('au:"Matthew Stephens"', limit=50)
rec2 <- arxiv_search('au:"John Lafferty"', limit=50)
rec3 <- arxiv_search('au:"Wei Biao Wu"', limit=50)
rec4 <- arxiv_search('au:"Peter Mccullagh"', limit=50)

primary_categories_1 <- toupper(rec1$primary_category)
primary_categories_2 <- toupper(rec2$primary_category)
primary_categories_3 <- toupper(rec3$primary_category)
primary_categories_4 <- toupper(rec4$primary_category)

factor_levels <- unique(c(unique(primary_categories_1),
                   unique(primary_categories_2),
                   unique(primary_categories_3),
                   unique(primary_categories_4)))

primary_categories_1 <- factor(primary_categories_1, levels=factor_levels)
primary_categories_2 <- factor(primary_categories_2, levels=factor_levels)
primary_categories_3 <- factor(primary_categories_3, levels=factor_levels)
primary_categories_4 <- factor(primary_categories_4, levels=factor_levels)


tab_data <- cbind(table(primary_categories_1),
                  table(primary_categories_2),
                  table(primary_categories_3),
                  table(primary_categories_4))

colnames(tab_data) <- c("Matthew Stephens",
                        "John Lafferty",
                        "Wei Biao Wu",
                        "Peter McCullagh")

tab_data <- as.matrix(tab_data)
```

We apply *Logolas* on the data

```
logomaker(tab_data,
          cols= RColorBrewer::brewer.pal(dim(tab_data)[1],
          name = "Spectral"),
          frame_width = 1,
          ic.scale = TRUE,
          pop_name = "arXiv field categories of UChicago STAT professors",
          xlab = "Professors",
          ylab = "Information content")
```



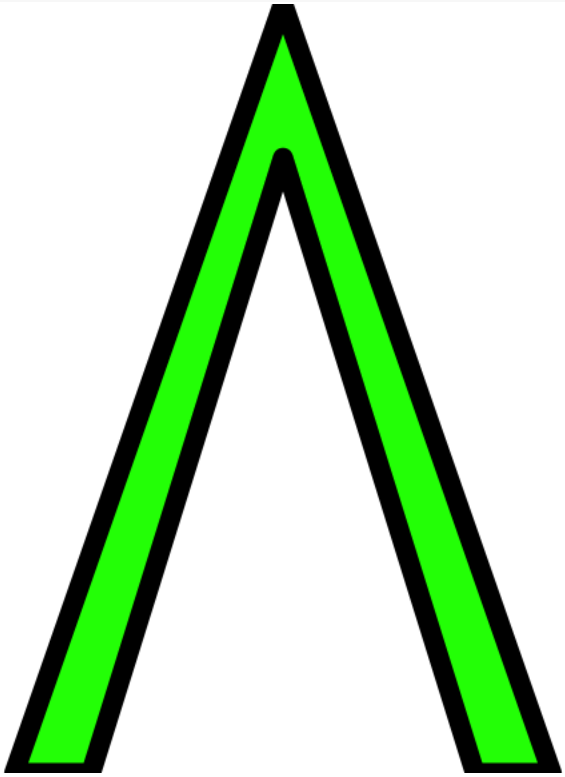
4 Creating logos and adding to Library

An user can create her own logo and add to her personalized library and *Logolas* provides a very simple interface for doing so.

For example, if one wants to have the symbol Lambda as part of his logo, he can create it as follows

```
LAMBDAletter <- function(colfill="green"){

  x <- c(0.15, 0.5, 0.85, 0.75, 0.5, 0.25)
  y <- c(0, 1, 0, 0, 0.8, 0)
```

```
lambda <- LAMBDAletter()
grid::grid.newpage()
grid::pushViewport(grid::viewport(x=0.5,y=0.5,width=1, height=1,
                                  clip=TRUE))
grid::grid.polygon(lambda$x, lambda$y,
                   default.unit="native",
                   id=lambda$id,
                   gp=grid::gpar(fill=lambda$fill,
                                 lwd=10))
```



```
  fill <- colfill
  id <- rep(1, length(x))

  ll <- list("x"= x,
             "y"= y,
             "id" = id,
             "fill" = fill)
  return(ll)
}
```

Do make sure, your function name is of the form ‘*letter”‘ where you can be creative with the ‘”*”‘ part. Also make sure the name you put must be in uppercase letters. You can check if the symbol plot looks like a lambda or not.

Once you are happy with the shape of the symbol, you can add it to the mix of all other symbols using

the following command.

Note that we put 'lambda' inside "/.../" to make sure that the function reads it as a new symbol and not general English alphabets or numbers.

This confirms that the symbol has been read into the mix, and now it can be used for stacking logo symbols under the 'logomaker()' functionality.

```
counts_mat <- rbind(c(0, 10, 100, 60, 20),
                    c(40, 30, 30, 35, 20),
                    c(100, 0, 15, 25, 75),
                    c(10, 30, 20, 50, 70)
)

colnames(counts_mat) <- c("Pos 1", "Pos 2", "Pos 3", "Pos 4", "Pos 5")
rownames(counts_mat) <- c("R/LAMBDA/Q", "A", "X", "Y")
```

LAMBDA symbol is added under 'addlogos' and 'addlogos$_text$'$options for the$'$logomaker$'$$mix of symbols$.

```
logomaker(counts_mat,
          cols= RColorBrewer::brewer.pal(dim(counts_mat)[1],
          name = "Spectral"),
          frame_width = 1,
          addlogos="LAMBDA",
          addlogos_text="LAMBDA")
```