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

Kushal K Dey

Stephens Lab, Dept. of Statistics, The University of Chicago

*Correspondending Email: kkdey@uchicago.edu

April 24, 2017

Abstract

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

Logolas version: 1.0.0 [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 for sequence and protein motif detection. One of the first and widely used logo plotting tools is *seqLogo* by Oliver Bembom [1], 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 [2]. Another package *motifStack* works with both DNA/RNA sequence motif and amino acid sequence motif and customizes font size and colors [3].

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. It provides a simple user interface to create new logos and add them to their personalized library of logos and even apply them in 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

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

Then load the package with:

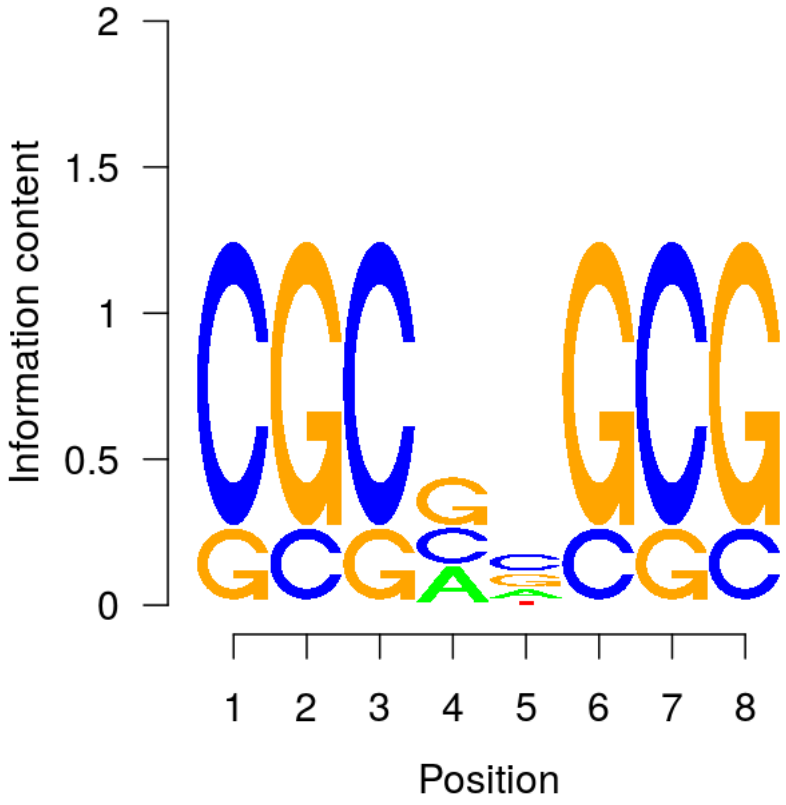
```
library(Logolas)
```

3 Applications

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 the same demo example provided in the *seqLogo* vignette.

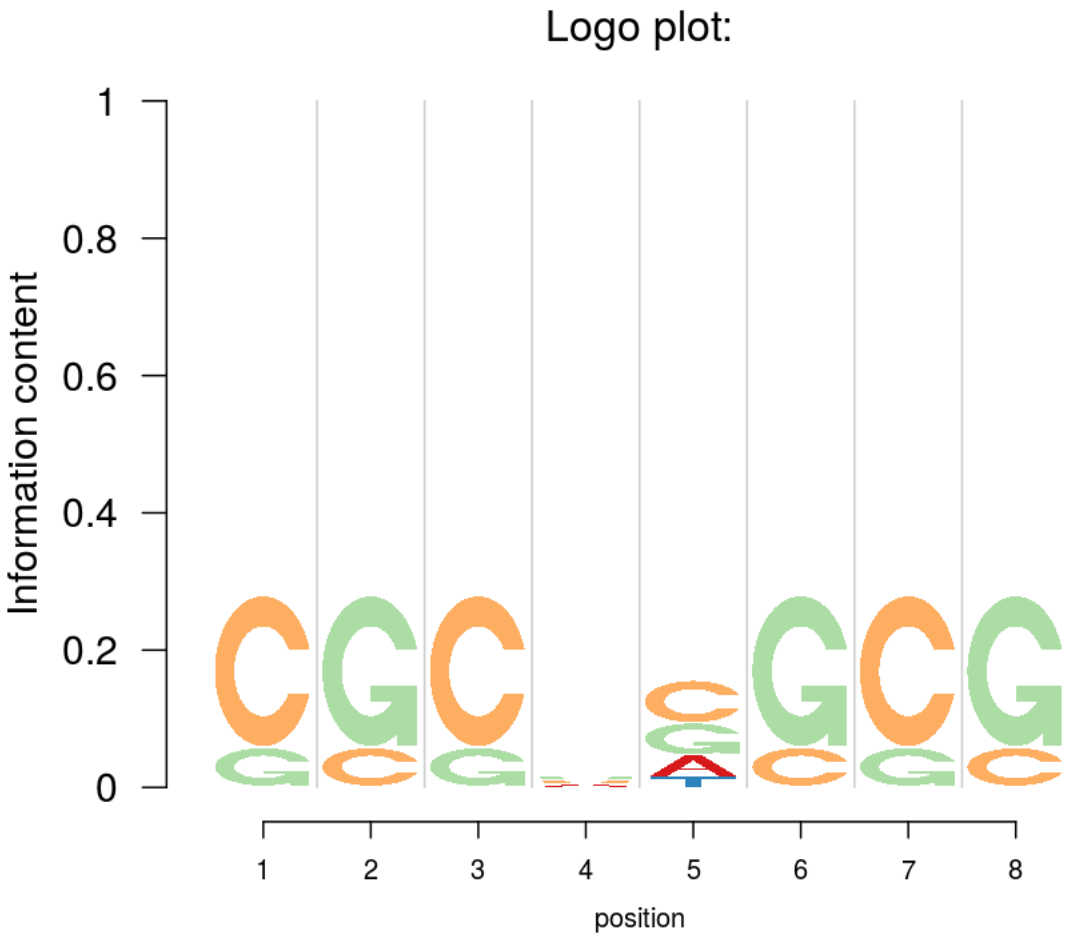
The user first needs to make a position weight matrix from the matrix using the `makePWM()` function. Then it uses the `seqLogo()` function on the output to plot the logo plots.

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



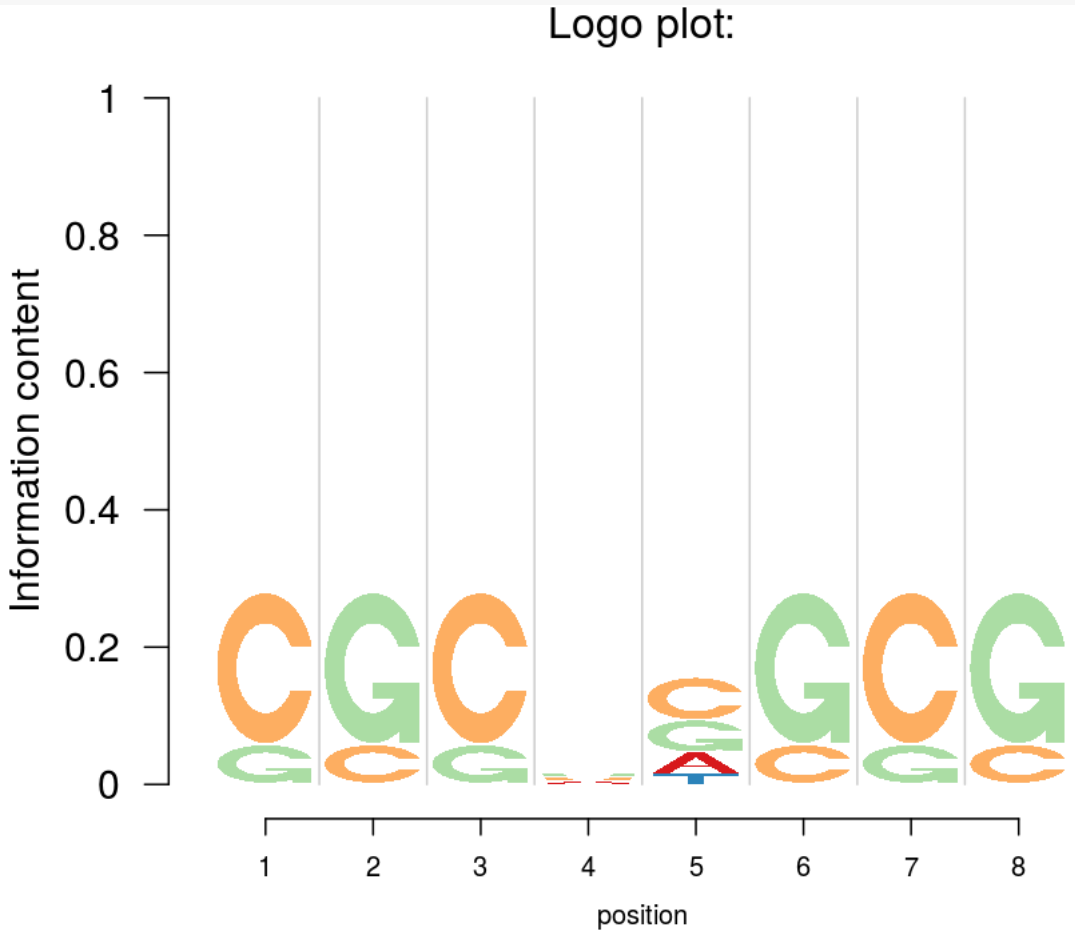
Now we apply *Logolas* to build a similar plot.The user can directly use the 'p@pwm' object generated by `seqLogo()` as shown below

```
color_profile <- list("type" = "per_row",
                      "col" = RColorBrewer::brewer.pal(dim(p@pwm)[1],name ="Spectral"))
logomaker(p@pwm,
          color_profile = color_profile,
          frame_width = 1,
          ic.scale = TRUE,
          yscale_change=FALSE,
          xlab="position",
          col_line_split = "grey80")
```



Besides using the 'makePWM' format of `seqLogo()` package, the user has the flexibility to directly input a matrix with row names and column names as well.

```
rownames(m) <- c("A", "C", "G", "T")
colnames(m) <- 1:8
logomaker(m,
          color_profile = color_profile,
          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 as in *seqLogo*. The size of logo in each stack is proportional to the relative abundance of that logo in that stack.

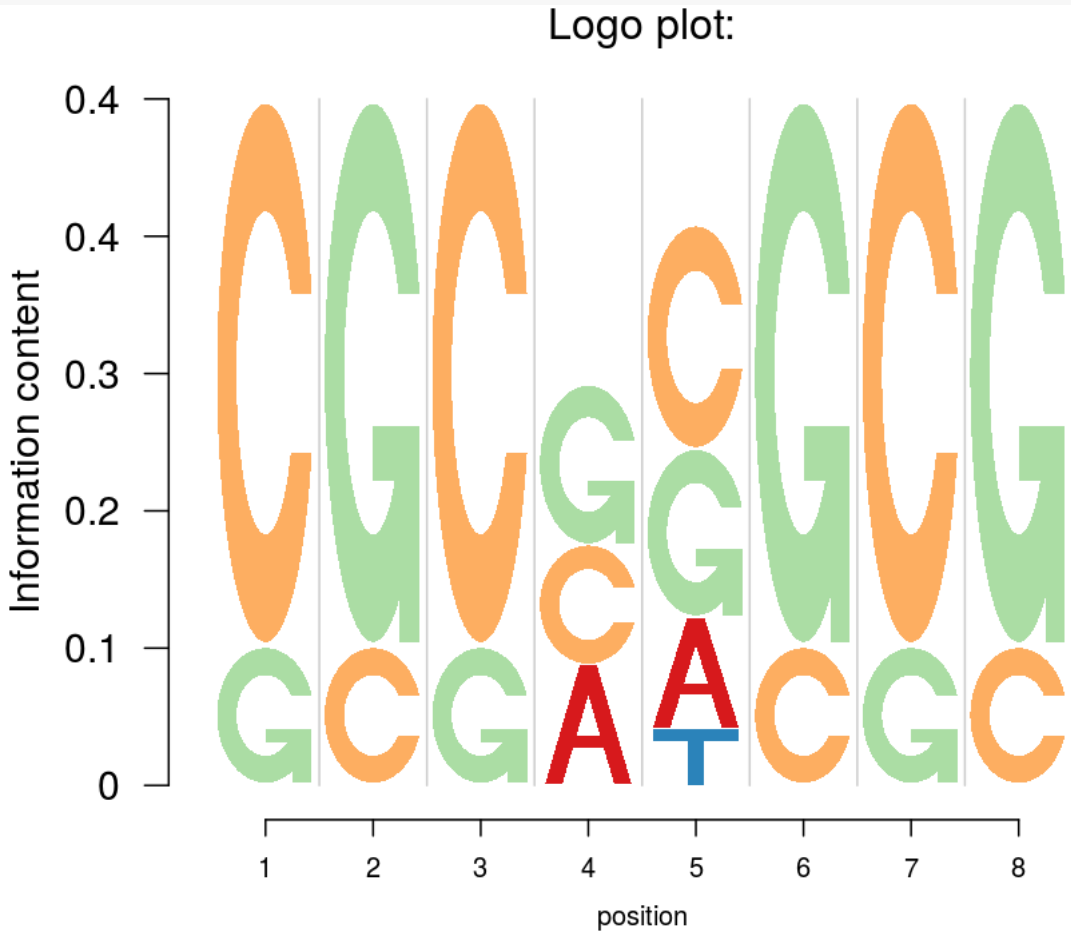
To change to other orders of Renyi entropy, one can tune the input parameter `alpha`. A higher value of `alpha` makes the logos more prominent, besides maintaining relative structure. One can set different infromation criteria using the `ic_computer` function.

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

```
## [1] 0.5283 0.5283 0.5283 0.0434 0.3390 0.5283 0.5283 0.5283
```

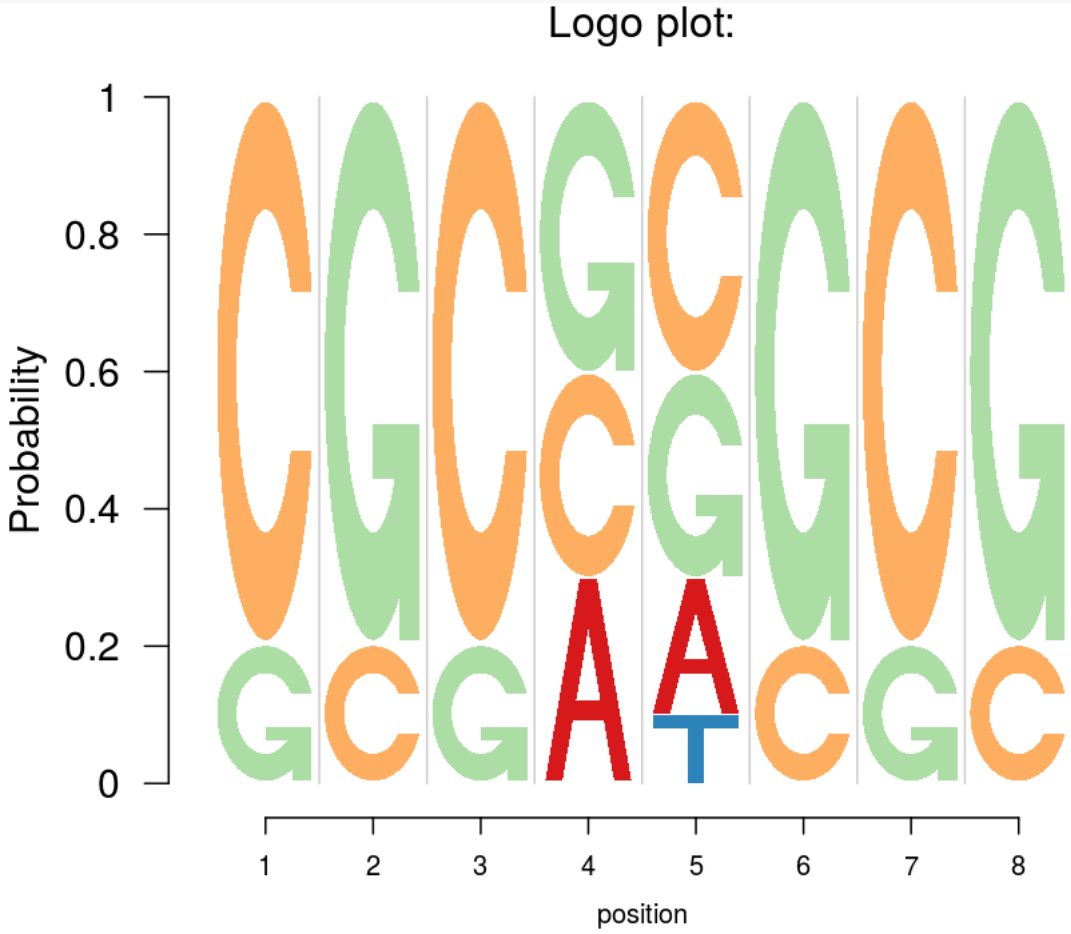
Also, the Y-axis can be adjusted by taking `yscale_change=TRUE`

```
rownames(m) <- c("A", "C", "G", "T")
colnames(m) <- 1:8
logomaker(m,
          color_profile = color_profile,
          frame_width = 1,
          ic.scale = TRUE,
          alpha = 2,
          yscale_change=TRUE,
          xlab="position",
          col_line_split = "grey80")
```



One can also normalize the 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,
          color_profile = color_profile,
          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 using `color_profile`, 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.

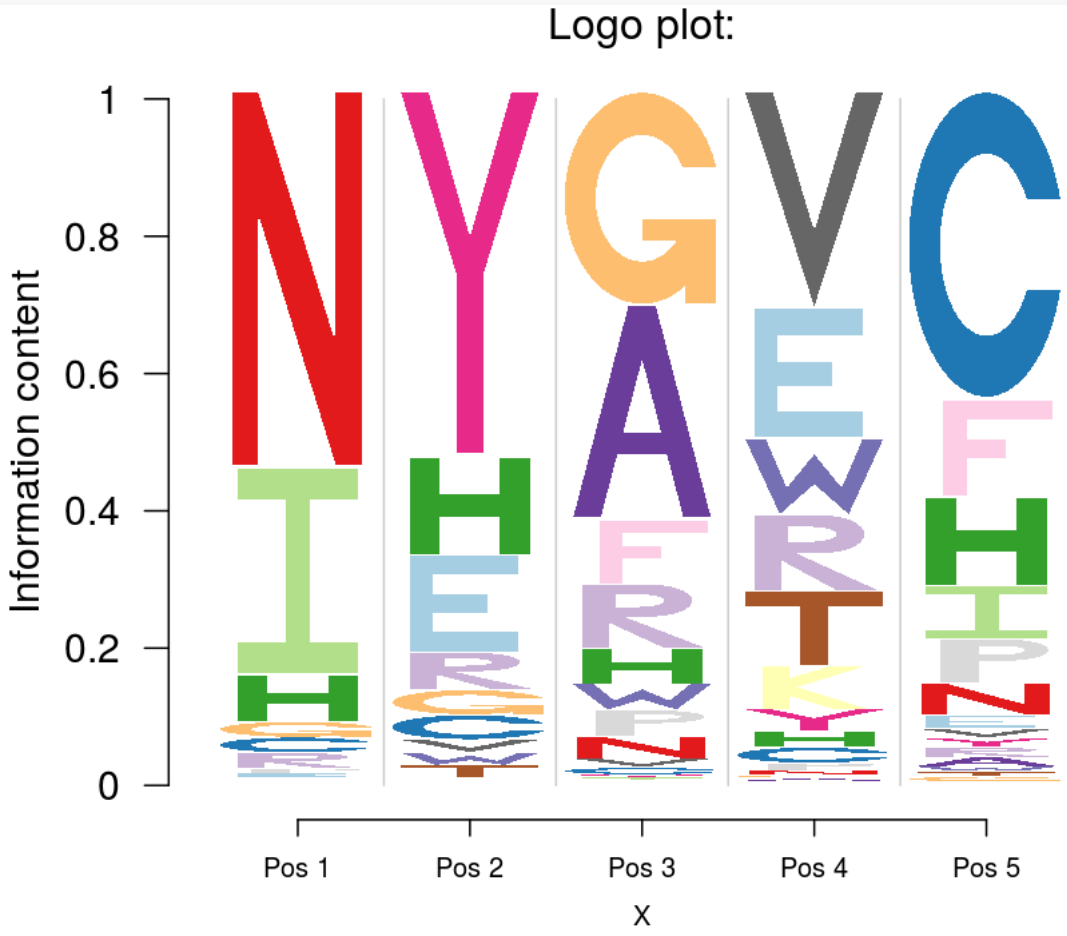
3.1 Amino acid sequence motif

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 using 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 one needs to do to build the logo plots is to specify 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)])

color_profile <- list("type" = "per_row",
                      "col" = cols1)

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



3.2 String Logos: Mutation profiling

We now step beyond alphabet logos and present the first example of how a string can be used as a logo. Suppose for a set of cell lines, we are provided data on the number of mutations (nucleotide substitions) and nucleotides at the flanking bases of the nucleotide substitution. This is the kind of problem addressed by Shiraishi et al 2015 [4]. They developed a package `pmsignature` for plotting the substitution and flanking bases profile, but here we use logos to do the same. We apply it here on a demo example.

```
mFile <- system.file("Exfiles/pwm1", package="seqLogo")
m <- read.table(mFile)
p <- seqLogo::makePWM(m)
pwm_mat <- slot(p,name = "pwm")
mat1 <- cbind(pwm_mat[,c(3,4)], rep(0,4), pwm_mat[,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 denote the $X \to Y$ 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 (column) 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 (columns).

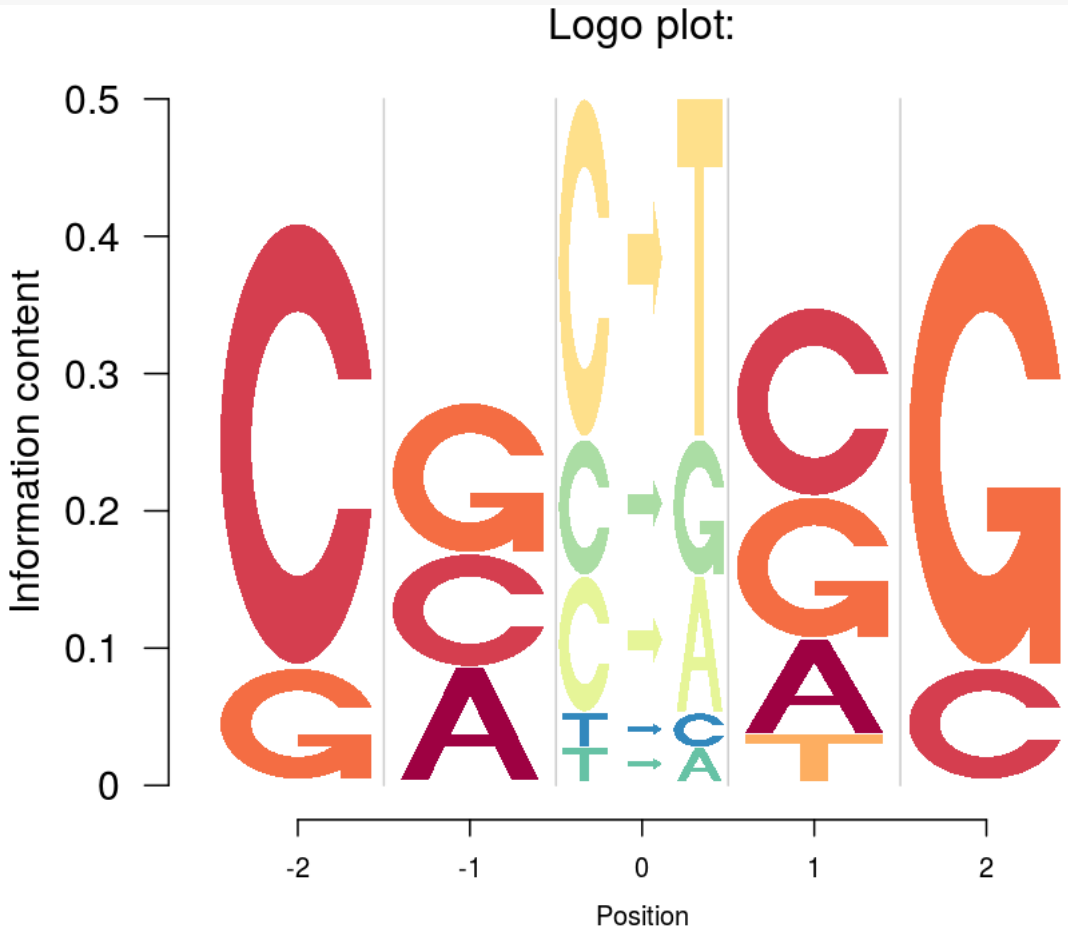
Then we apply `logomaker` on that matrix.

One issue with this plot is that the user may want to have the C in `C>T` to be of the same color, but here the symbol C and `C>T` are treated as separate entities. However Logolas coloring profile provides the user the flexibility to color each symbol instead of a string. We use the color type `per_symbol` instead of the `per_row` profile we have been using so far.

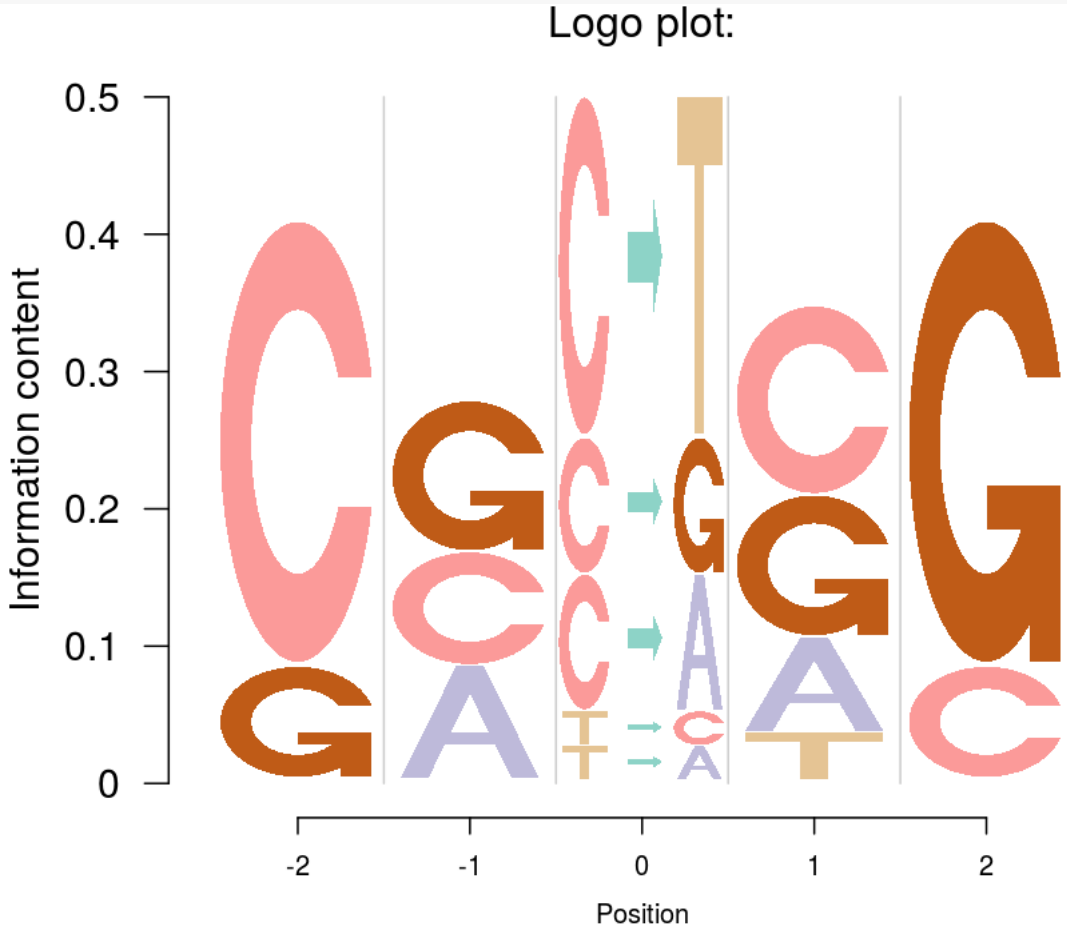
We consider a list of all symbols `total_chars` which is set as default to the list chosen above (so you can skip the `total_chars` argument below). However, if the user adds a symbol to the library (the process of doing that we show in the end), then the library is expected to grow and the user then might want to update the `total_chars` list by adding new symbols.

Another coloring option is `per_column`, in which we have a specific color for a specific column. This sort of coloring might be useful when the user wants to highlight the difference between columns. We provide an example of this next.

```
color_profile <- list("type" = "per_row",
                      "col" = RColorBrewer::brewer.pal(dim(table)[1],name ="Spectral"))

logomaker(table,
          color_profile = color_profile,
          frame_width = 1,
          ic.scale = TRUE,
          yscale_change=TRUE,
          xlab = "Position",
          ylab = "Information content")
```



```
cols = RColorBrewer::brewer.pal.info[RColorBrewer::brewer.pal.info$category == 'qual',]
col_vector = unlist(mapply(RColorBrewer::brewer.pal, cols$maxcolors, rownames(cols)))

total_chars = c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O",
            "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z", "zero", "one", "two",
            "three", "four", "five", "six", "seven", "eight", "nine", "dot", "comma",
            "dash", "colon", "semicolon", "leftarrow", "rightarrow")

set.seed(20)
color_profile <- list("type" = "per_symbol",
                      "col" = sample(col_vector, length(total_chars), replace=FALSE))

logomaker(table,
          color_profile = color_profile,
          total_chars = total_chars,
          frame_width = 1,
          ic.scale = TRUE,
          yscale_change=TRUE,
          xlab = "Position",
          ylab = "Information content")
```



3.3 String Logos: Ecological Data

String logos can be used to represent how families or genus of species vary across 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 act as logos and the clusters of regions represent the stacks in the logo plot. Here the coloring pattern used is `per_column` which distinguished between the different clusters of bird species.

3.4 String Logos: Histone marks patterns

In studies related 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] [5]. The authors recorded number of histone modification sites identified by their algorithm 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.

3.5 String Logos: Document mining

So far we mainly focused on biology examples where *Logolas* can be applied. But, it has applications beyond the field of biology. One example application of *Logolas* is in document mining and representing keywords or tags as logos.

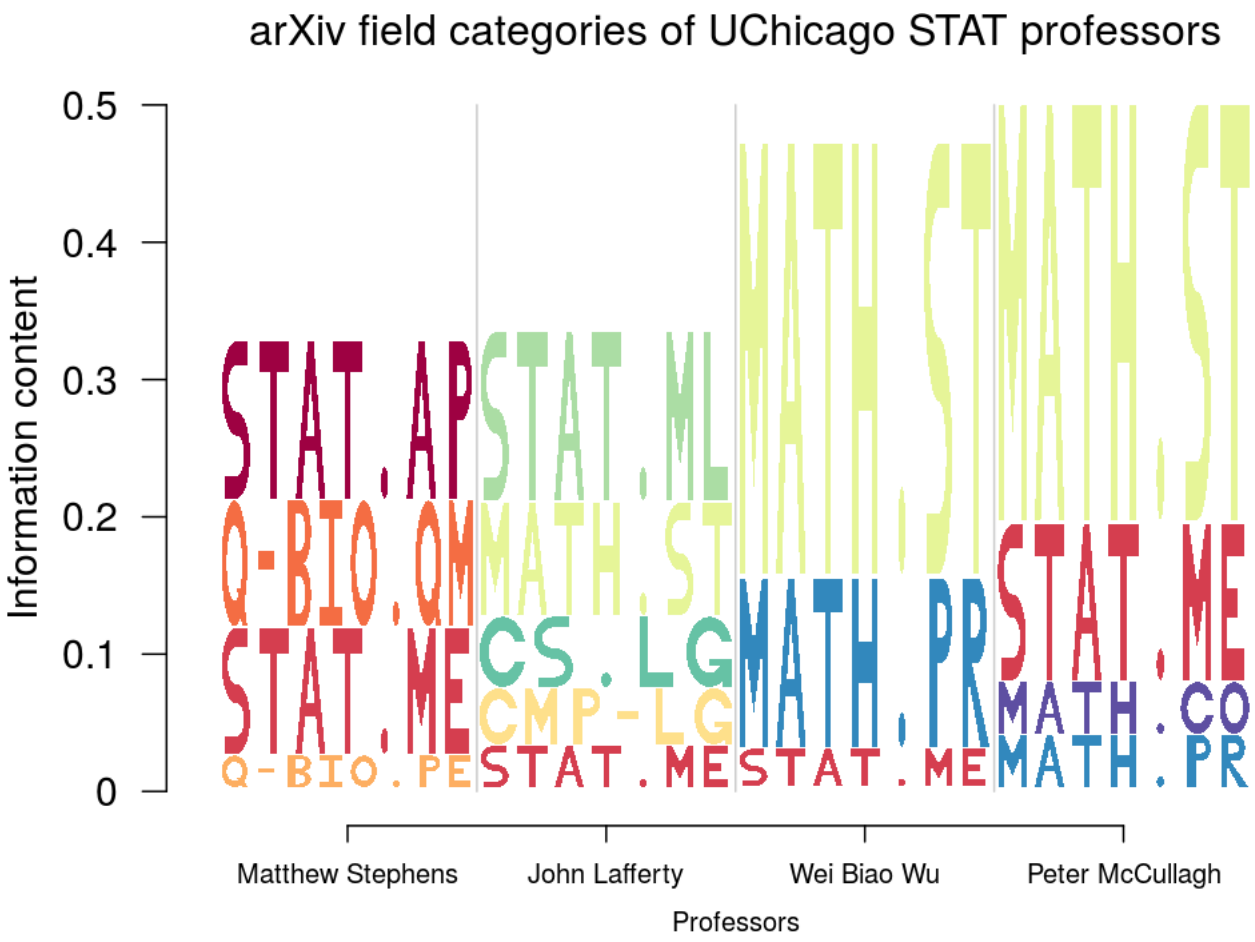
Here we build a logo plot of the field categories of manuscipts submitted on aRxiv by 4 professors from Dept. of Statistics, University of Chicago. Note that the field categories here are a combination of numbers, alphabets, dots and dashes.

We first generate the data

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

Next, we apply *Logolas* on the data

```
color_profile <- list("type" = "per_row",
                      "col" = RColorBrewer::brewer.pal(dim(tab_data)[1],
          name = "Spectral"))

logomaker(tab_data,
          color_profile = color_profile,
          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 her logo, she 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)

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

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

The function name has to be of the form ‘*letter"‘ where the user can be creative with the ‘"*"‘ part. Also the name must be in uppercase letters. The user can then check if the symbol plot looks like a lambda or not.

The user can then add this symbol to the library which contains alphabets, numbers, punctuations etc already.

To use `lambda` as part of a string, the user has to put `lambda` inside "/.../" to make sure that the function reads it as a new symbol and not general English alphabets or numbers. We provide an example below.

```
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/LMBD/Q", "A", "X", "Y")
```

LAMBDA symbol is added under `addlogos` and `addlogos_text` options for the `logomaker` mix of symbols.

5 Conclusion

Logolas allows an user to use alphabets, numbers, punctuations, arrows and alpha-numeric strings as logos, which expands the horizon of applicability of logo plots. Logo plots can be used as a substitute for divided bar charts and pie charts, replacing colors by actual symbols of the category represented and thereby deprecating the need for legends. It also can be used to see patterns of variation of categories (logos) observed across time points, genomic positions or any sequential variable (stacks or columns). Finally, the flexibility of creating new logos, which can be new shapes or symbols, expands the scope of logo plots even further.

6 Acknowledgements

We would like to acknowledge Oliver Bembom, the author of 'seqLogo' for acting as an inspiration and providing the foundation on which this package is created. We would also like to thank Kevin Luo, Hussein al Asadi, John Blischak and Alex White for helpful discussions.

7 Session Info

```
sessionInfo()
```

```
## R version 3.4.0 (2017-04-21)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.2 LTS
##
## Matrix products: default
## BLAS: /home/biocbuild/bbs-3.5-bioc/R/lib/libRblas.so
## LAPACK: /home/biocbuild/bbs-3.5-bioc/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] Logolas_1.0.0 knitr_1.15.1
```

```
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.10       XML_3.98-1.6       digest_0.6.12      rprojroot_1.2
##  [5] R6_2.2.0           grid_3.4.0         backports_1.0.5    stats4_3.4.0
##  [9] magrittr_1.5       evaluate_0.10      httr_1.2.1         highr_0.6
## [13] stringi_1.1.5      curl_2.5           aRxiv_0.5.15       rmarkdown_1.4
## [17] BiocStyle_2.4.0    RColorBrewer_1.1-2 tools_3.4.0        stringr_1.2.0
## [21] yaml_2.1.14        compiler_3.4.0     seqLogo_1.42.0     htmltools_0.3.5
```

References

[1] Bembom O (2016). seqLogo: Sequence logos for DNA sequence alignments. R package version 1.40.0.

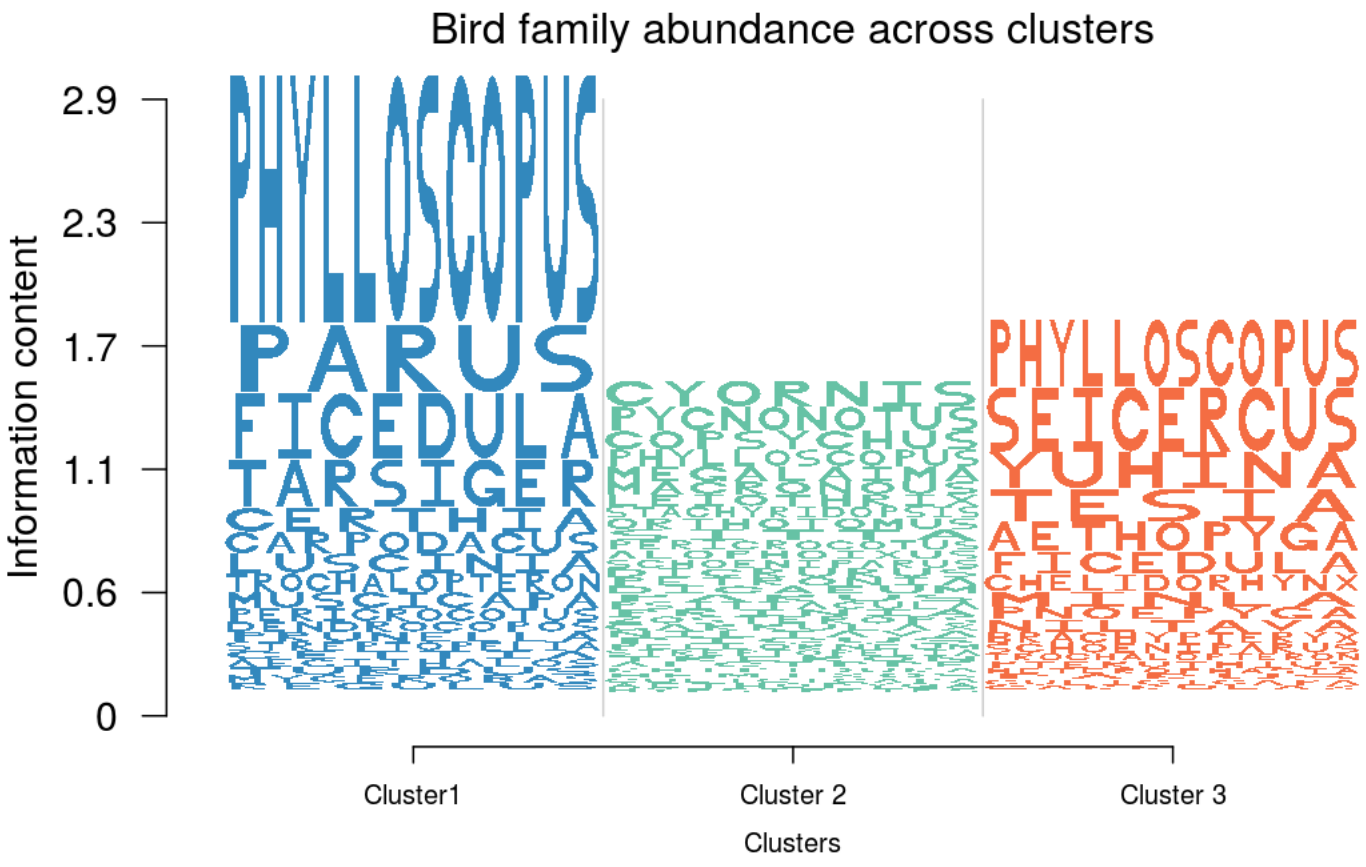
[2] Omar Wagih (2014). RWebLogo: plotting custom sequence logos. R package version 1.0.3. https://CRAN.R-project.org/package=RWebLogo

[3] Jianhong Ou and Lihua Julie Zhu (2015). motifStack: Plot stacked logos for single or multiple DNA, RNA and amino acid sequence. R package version 1.14.0.

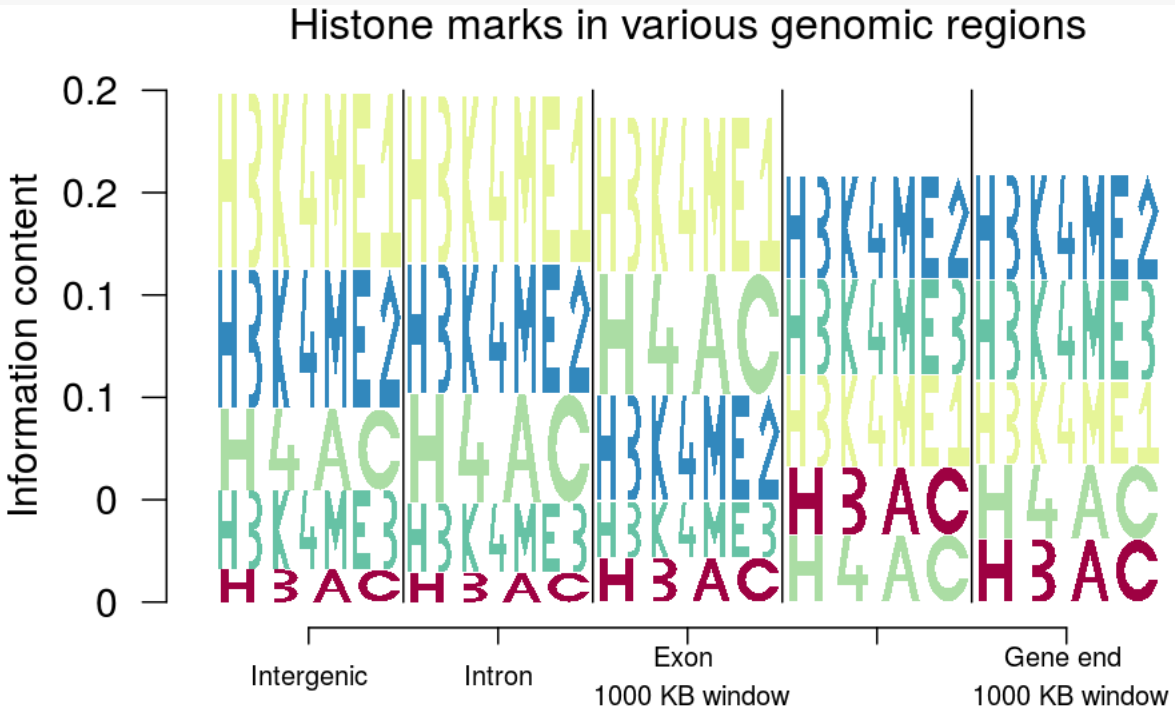
[4] Shiraishi Y, Tremmel G, Miyano S, Stephens M (2015) A Simple Model-Based Approach to Inferring and Visualizing Cancer Mutation Signatures. PLoS Genet 11(12): e1005657. doi: 10.1371/journal.pgen.1005657

[5] Koch CM, Andrews RM, Flicek P, et al (2007). The landscape of histone modifications across 1% of the human genome in five human cell lines. Genome Research. 2007;17(6):691-707. doi:10.1101/gr.5704207.

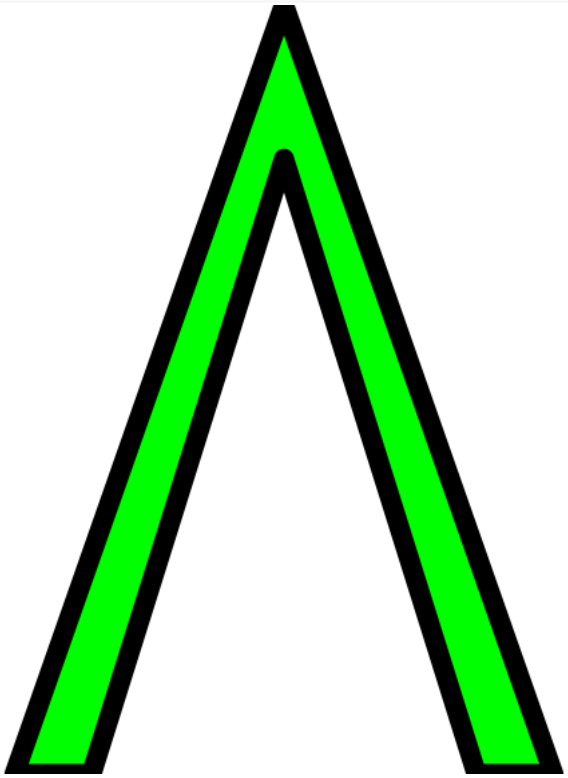
```
set.seed(20)
data("himalayan_fauna_3_clusters")

color_profile <- list("type" = "per_column",
                      "col" = sample(RColorBrewer::brewer.pal(10,name = "Spectral"),
                       dim(himalayan_fauna_3_clusters)[2], replace=TRUE))
logomaker(himalayan_fauna_3_clusters,
          color_profile = color_profile,
          frame_width = 1,
          ic.scale = TRUE,
          alpha=200,
          pop_name = "Bird family abundance across clusters",
          xlab = "Clusters",
          ylab = "Information content")
```



```
color_profile <- list("type" = "per_row",
                      "col" = sample(RColorBrewer::brewer.pal(10,name = "Spectral"),
                          dim(mat)[1]))


logomaker(mat,
          color_profile = color_profile,
          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")
```



```
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))
```



```
color_profile <- list("type" = "per_row",
                      "col" = RColorBrewer::brewer.pal(dim(counts_mat)[1], name = "Spectral

logomaker(counts_mat,
          color_profile = color_profile,
          frame_width = 1,
          addlogos="LMBD",
          addlogos_text="LAMBDA")
```