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

Kushal K Dey, Dongyue Xie, Matthew Stephens

Dept. of Statistics, The University of Chicago

*Correspondending Email: kkdey@uchicago.edu

October 30, 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: 1.2.1 [1]

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

Contents

1 Introduction

Ever since its introduction due to Schneider and Stephens (1990), sequence logos have proved to be an effective tool in informative visualization of patterns in aligned sets of sequences. There are several existing packages for plotting sequence logos, like *seqLogo*, *seq2Logo*, *motifStack* etc.

We introduce another Logo visualization package called *Logolas* which addresses some limitations of the above packages and makes logo visualization a more generic tool with potential applications in a much wider scope of problems, in comparison to the existing softwares. Some features of *Logolas* are as follows.

- **EDLogo Representation** : General logo plotting softwares highlight only enrichment of certain symbols, but Logolas allows the user to highlight both enrichment and depletion of symbols at any position, leading to more parsimonious and visually appealing representation.

- **Strings as symbols** : General logo building softwares have limited library of symbols usually restricted to English alphabets. Logolas allows the user to plot symbols for any alphanumeric string, comprising

of English alphabets, numbers, punctuation marks, arrows etc. It also provides an easy interface for the user to create her own logo and add to the library of symbols that can be plotted (check the Vignette).

- **Dirichlet Adaptive Shrinkage** : Most existing logo plotting tools take position weight matrix (PWM) as input for the logo plot and do not adapt well to the scale of the underlying positional frequencies from which the PWM is created. In Logolas, we provide a Dirichlet Adaptive Shrinkage method (dash) approach for adaptively scaling the heights of the logos based on the frequency scale. This approach is similar in lines to the **ashr** software due to M. Stephens (2016).

- **Better stylizations** : Logolas also allows for new and flexible stylizations of the logos, involving color palettes, new fill and border styles, several options for determining heights of the logos etc. We also show how to plot multi-panel Logolas plots and combine Logolas plots with ggplot2 graphics.

2 Logolas Installation

Logolas loads as dependencies the following CRAN-R package : *grid*, *gridExtra*, *graphics*, *SQUAREM*, *LaplacesDemon*, *LaplacesDemon*, *Matrix*, *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 Standard Logos

One of the first applications of logo plots have been in visualization of DNA sequence motifs. We start with the example PWM from **seqLogo**.

3.1 First example with Logolas

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

We next define the color pattern for the logo plot.

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

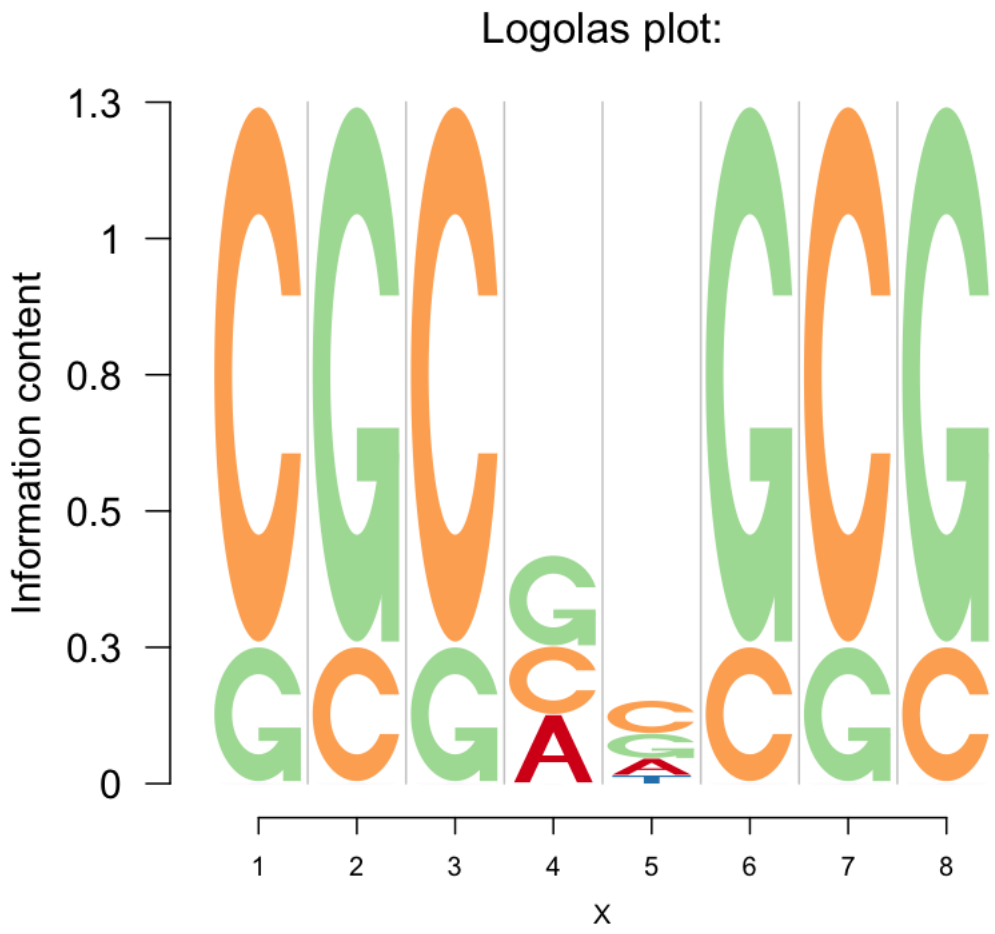
We now apply the main plotting function *logomaker* in **Logolas** for plotting the standard logo plot.

3.2 Different configurations of standard logo

Logolas also allows the user to choose different colors, change the scale of the Y axis, normalize the height of the logos and also play around with the margins of the plot in the plotting region. We present a demonstration of these feature of **Logolas** in the example below. We also show in this example, how multi-panel Logolas plots can be created.

We first set up the multiple panels and then present the standard logo plots under different customizations.

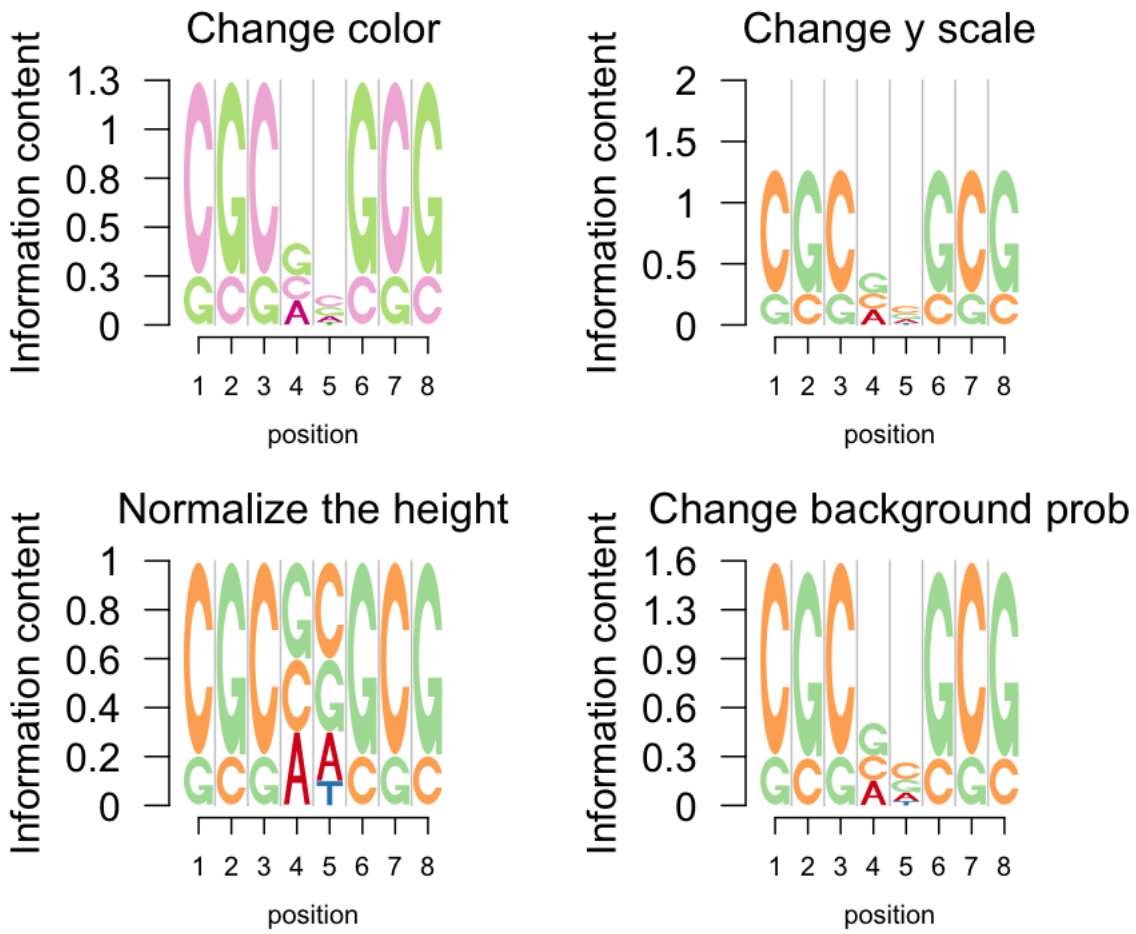
```
logomaker(attr(p, "pwm"),color_profile = color_profile, frame_width = 1,ic.scale = TRUE)
```




```
#######################      Setting up multiple panels       ########################

library(grid)
library(gridBase)
library(gridExtra)

grid.newpage()
layout.rows <- 2
layout.cols <- 2
top.vp <- viewport(layout=grid.layout(layout.rows, layout.cols,
                                      widths=unit(rep(5,layout.cols), rep("null", 2)),
                                      heights=unit(rep(5,layout.rows), rep("null", 1))))
plot_reg <- vpList()
l <- 1
for(i in 1:layout.rows){
  for(j in 1:layout.cols){
    plot_reg[[l]] <- viewport(layout.pos.col = j, layout.pos.row = i, name = paste0("plotl
```

```
    l <- l+1
  }
}


plot_tree <- vpTree(top.vp, plot_reg)

pushViewport(plot_tree)


############################   Creating the multi-panel logos   ############################


#change the color of letters
#change the diverging palettes
color_profile1=list("type" = "per_row",
"col" = RColorBrewer::brewer.pal(dim(attr(p, "pwm"))[1],name ="PiYG"))
seekViewport(paste0("plotlogo", 1))
logomaker(attr(p, "pwm"),xlab = 'position',color_profile = color_profile1,
          frame_width = 1,
          newpage = FALSE,
          pop_name = 'Change color',
          control = list(viewport.margin.left = 5))

#change the y scale:
#if yscale_change = FALSE, then the height of y axis would be 2.
seekViewport(paste0("plotlogo", 2))
logomaker(attr(p, "pwm"),xlab = 'position',color_profile = color_profile,
          frame_width = 1,
          newpage = FALSE,
          pop_name = 'Change y scale',
          yscale_change = FALSE,
          control = list(viewport.margin.left = 5))
#Normalize the height of bars to 1
seekViewport(paste0("plotlogo", 3))
logomaker(attr(p, "pwm"),xlab = 'position',color_profile = color_profile,
          frame_width = 1,
          newpage = FALSE,
          ic.scale = FALSE,
          pop_name = 'Normalize the height',
          control = list(viewport.margin.left = 5))
#change the background probability
#And modify the title and the axis label
seekViewport(paste0("plotlogo", 4))
```

```
logomaker(attr(p, "pwm"),xlab = 'position',color_profile = color_profile,
          frame_width = 1,
          bg=c(0.32, 0.18, 0.2, 0.3),
          newpage = FALSE,
          pop_name = 'Change background prob',
          control = list(viewport.margin.left = 5))
```

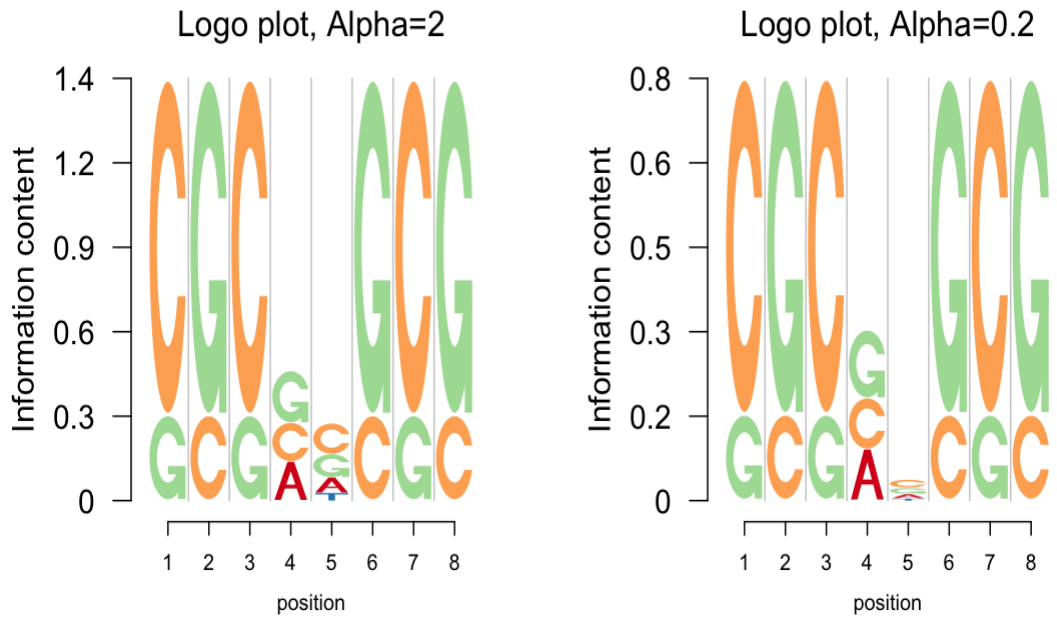


3.3 Other entropy measures

Besides the default Shannon entropy, *logomaker* function in textbfLogolas can also determine the Information Criterion using Renyi entrop. For this entropy measure, the Information content at position i is $IC_{i,\alpha} = \frac{1}{1-\alpha}\Sigma_b \log_2(q_{b,i}^{\alpha} - 0.25^{1-\alpha})$. When $\alpha \to 1$, the limiting value of Renyi entropy is the Shannon entropy. In 'logomaker', 'alpha' is a control parameter that can be tuned to get plots for different entropy measures. The figure below shows the logo plots with different values of α.

```
grid.newpage()
layout.rows <- 1
layout.cols <- 2
```

```
top.vp <- viewport(layout=grid.layout(layout.rows, layout.cols,
                                      widths=unit(rep(6,layout.cols), rep("null", 2)),
                                      heights=unit(c(20,20), rep("lines", 2))))

plot_reg <- vpList()
l <- 1
for(i in 1:layout.rows){
  for(j in 1:layout.cols){
    plot_reg[[l]] <- viewport(layout.pos.col = j, layout.pos.row = i, name = paste0("plotlo
    l <- l+1
  }
}


plot_tree <- vpTree(top.vp, plot_reg)

pushViewport(plot_tree)
seekViewport(paste0("plotlogo", 1))
logomaker(attr(p, "pwm"),xlab = 'position',color_profile = color_profile,
          frame_width = 1,
          newpage = FALSE,
          pop_name = 'Logo plot, Alpha=2',
          control = list(viewport.margin.left = 5,alpha=2))


seekViewport(paste0("plotlogo", 2))
logomaker(attr(p, "pwm"),xlab = 'position',color_profile = color_profile,
          frame_width = 1,
          newpage = FALSE,
          pop_name = 'Logo plot, Alpha=0.2',
          control = list(viewport.margin.left = 5,alpha=0.2))
```

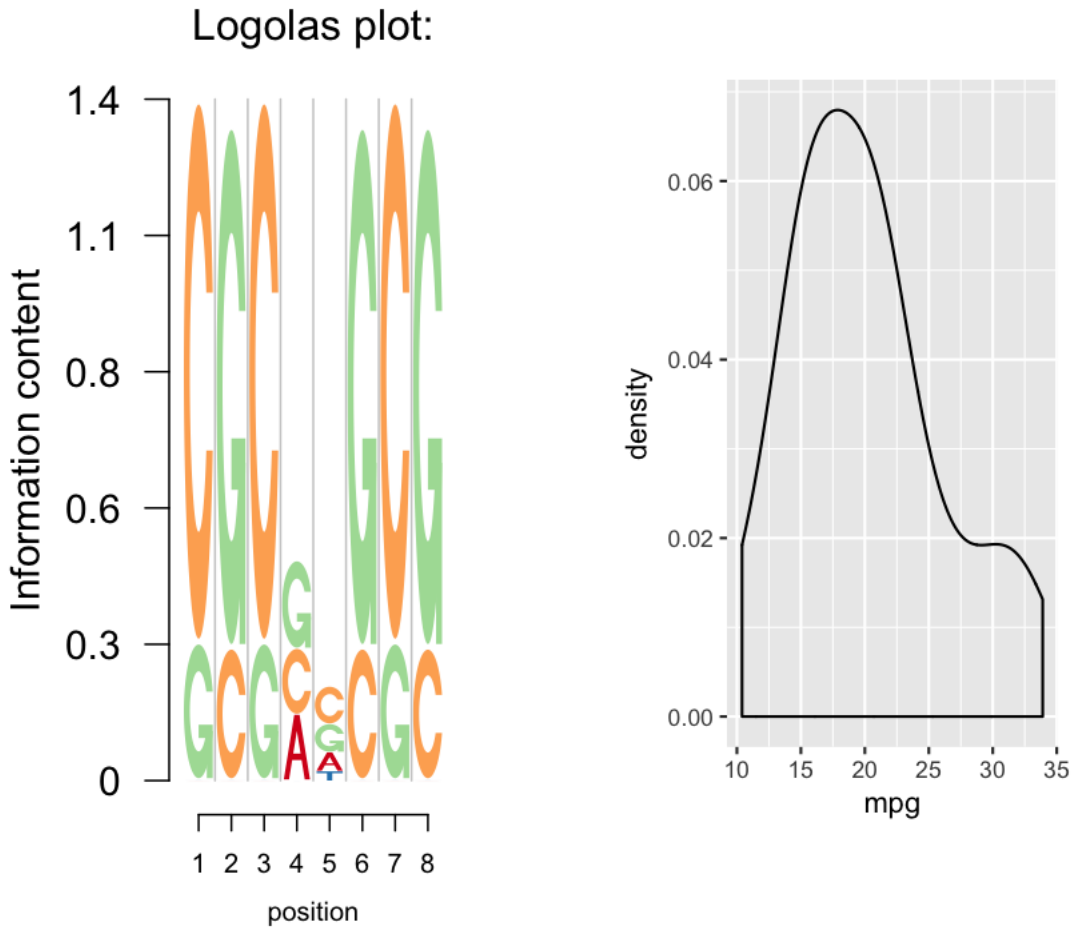


3.4 Combine Logolas with ggplot2

The user can also embed the Logolas plots with ggplot2 graphics. We present a simple illustration below.

```
library(ggplot2)
library(grid)
library(gridBase)


grid.newpage()
layout.rows <- 1
layout.cols <- 2
top.vp <- viewport(layout=grid.layout(layout.rows, layout.cols,
                                      widths=unit(rep(6,layout.cols), rep("null", layout.co
                                      heights=unit(rep(20,20), rep("null",1))))

plot_reg <- vpList()
```

```
l <- 1
for(i in 1:layout.rows){
  for(j in 1:layout.cols){
    plot_reg[[l]] <- viewport(layout.pos.col = j, layout.pos.row = i, name = paste0("plotlo
    l <- l+1
  }
}


plot_tree <- vpTree(top.vp, plot_reg)

pushViewport(plot_tree)
seekViewport(paste0("plotlogo", 1))
logomaker(attr(p, "pwm"),xlab = 'position',color_profile = color_profile,
          bg = c(0.28, 0.22, 0.24, 0.26),
          frame_width = 1,
          newpage = FALSE,
          control = list(viewport.margin.left = 5))


seekViewport(paste0("plotlogo", 2))
vp3 = viewport(width=0.8, height=0.8, x = 0.5, y = 0.5)
p <- qplot(mpg, data=mtcars, geom="density")
print(p, vp = vp3)
```



4 EDLogo Representation

Standard sequence logos tend to highlight the enrichment of symbols. In this esection, we introduce *EDLogo*, an alternative visualization to the standard sequence logos that highlights both enrichment and depletion.

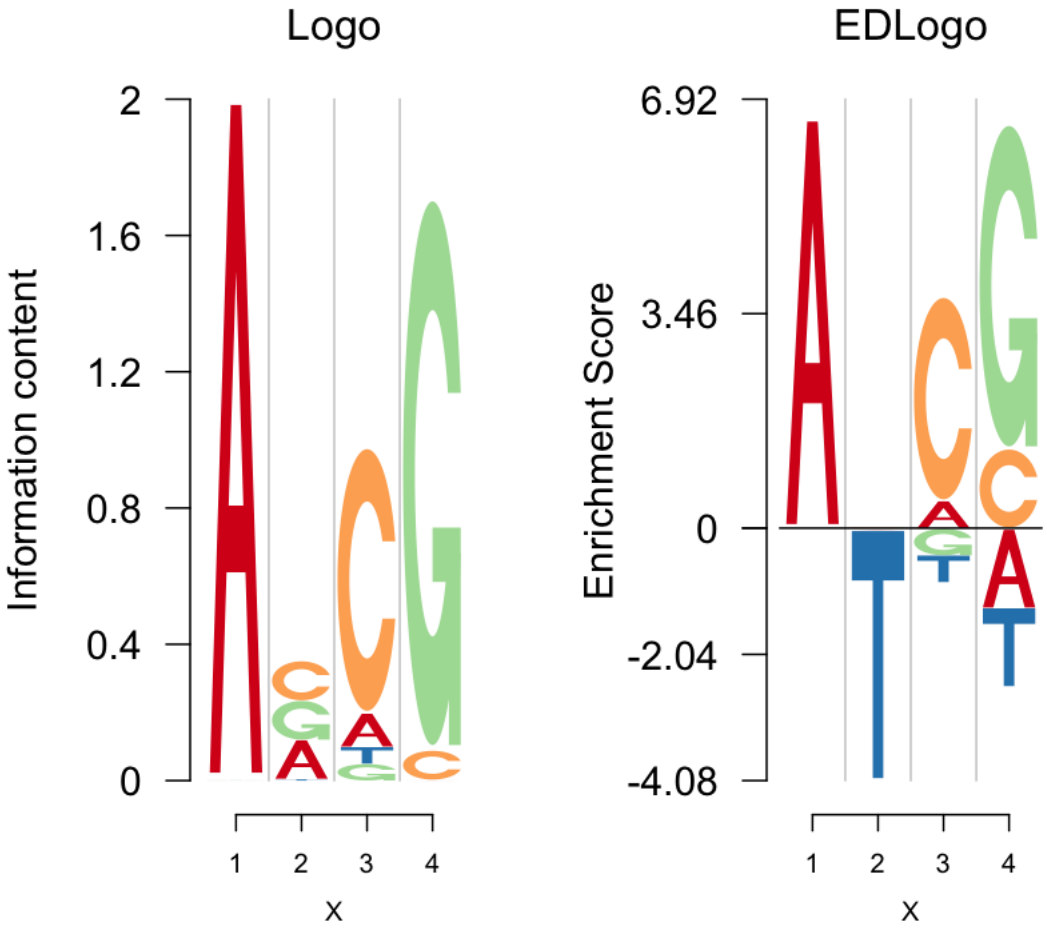
For example, suppose for a particular position, the base composition of (A, C, G, T) is as follows

$$(A, C, G, T) := (0.33, 0.33, 0.01, 0.33)$$

Sequence logos will show a mild enrichment of A, C, G in this position and the stack height for this position will be relativelys small compared to the ones with enrichment. This generally makes it hard to see the depletion. *EDLogo* is a solution to this problem. We illustrate that with an example compariosn of *EDLogo* and standard sequence logo below.

4.1 EDLogo - First example

```
library(ggplot2)
library(grid)
library(gridBase)


grid.newpage()
layout.rows <- 1
layout.cols <- 2
top.vp <- viewport(layout=grid.layout(layout.rows, layout.cols,
                                      widths=unit(rep(6,layout.cols), rep("null", layout.co
                                      heights=unit(rep(20,20), rep("null",1))))

plot_reg <- vpList()
l <- 1
for(i in 1:layout.rows){
  for(j in 1:layout.cols){
    plot_reg[[l]] <- viewport(layout.pos.col = j, layout.pos.row = i, name = paste0("plotlo
    l <- l+1
  }
}

plot_tree <- vpTree(top.vp, plot_reg)

pushViewport(plot_tree)

pwm=cbind(c(1,0,0,0),c(0.328,0.332,0.33,0.01),
          c(0.1,0.8,0.05,0.05),c(0,0.05,0.95,0))
rownames(pwm)=c('A','C','G','T')
colnames(pwm)=1:ncol(pwm)

seekViewport(paste0("plotlogo", 1))
logomaker(pwm,color_profile=color_profile,frame_width = 1,pop_name = 'Logo',control = list(

seekViewport(paste0("plotlogo", 2))
nlogomaker(pwm,logoheight = 'log',color_profile = color_profile,pop_name = 'EDLogo',control
```

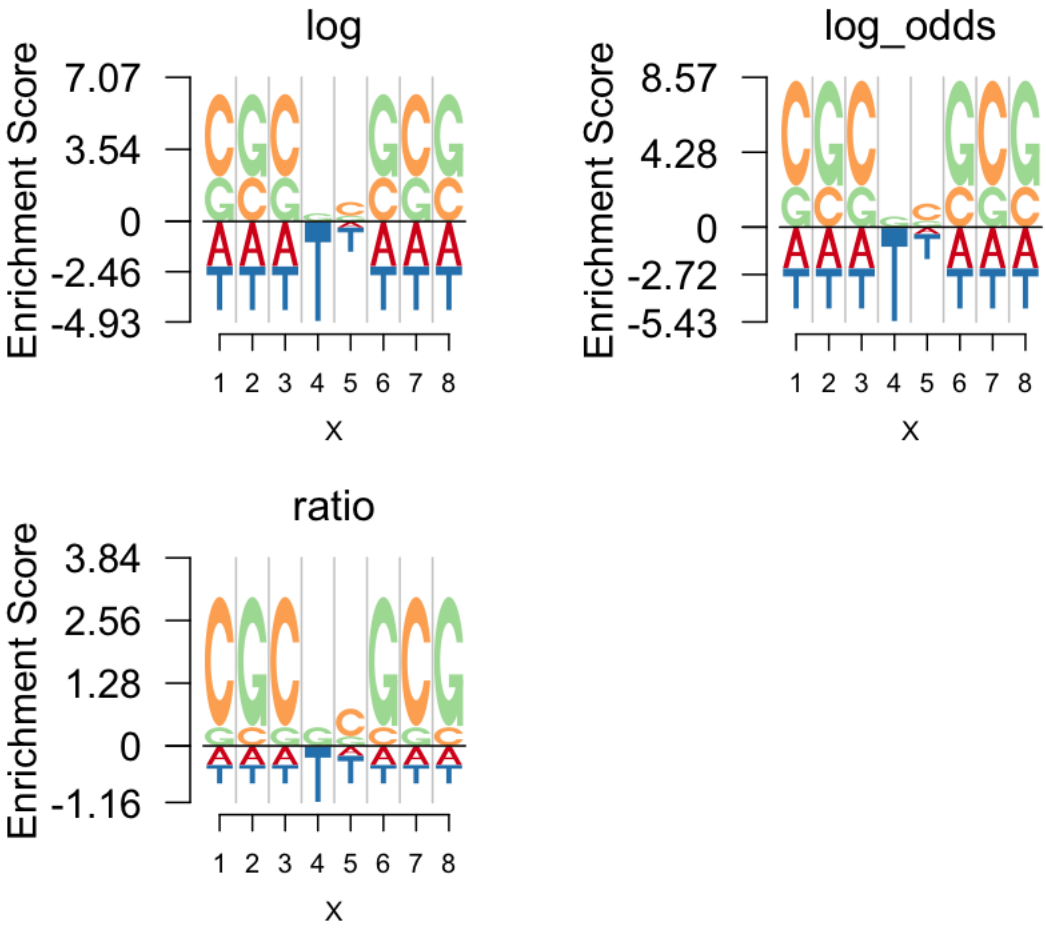


4.2 Different options of EDLogo

The *EDLogo* plot is created using the *nlogomaker* function. Note that there is a *logoheight* option in this plot which was set to *log* for the above example. Actually there are total 8 options for *logoheight* - the details can be found in Documentation. A comparative analysis of some of these options on the seqLogo example is provided below.

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

grid.newpage()
layout.rows <- 2
layout.cols <- 2
top.vp <- viewport(layout=grid.layout(layout.rows, layout.cols,
                                      widths=unit(rep(5,layout.cols), rep("null", 2)),
                                      heights=unit(rep(5,layout.rows), rep("null", 1))))
plot_reg <- vpList()
```

```
l <- 1
for(i in 1:layout.rows){
  for(j in 1:layout.cols){
    plot_reg[[l]] <- viewport(layout.pos.col = j, layout.pos.row = i, name = paste0("plotlo
    l <- l+1
  }
}


plot_tree <- vpTree(top.vp, plot_reg)

pushViewport(plot_tree)
seekViewport(paste0("plotlogo", 1))
nlogomaker(attr(p, "pwm"),logoheight = 'log',color_profile = color_profile,frame_width = 1


seekViewport(paste0("plotlogo", 2))
nlogomaker(attr(p, "pwm"),logoheight = 'log_odds',color_profile = color_profile,frame_widt

seekViewport(paste0("plotlogo", 3))
nlogomaker(attr(p, "pwm"),logoheight = 'ratio',color_profile = color_profile,frame_width =
```



5 Applications

We present some examples beyond the DNA, RNA sequence motif examples that we dealt with thus far.

5.1 Amino acid sequence motif

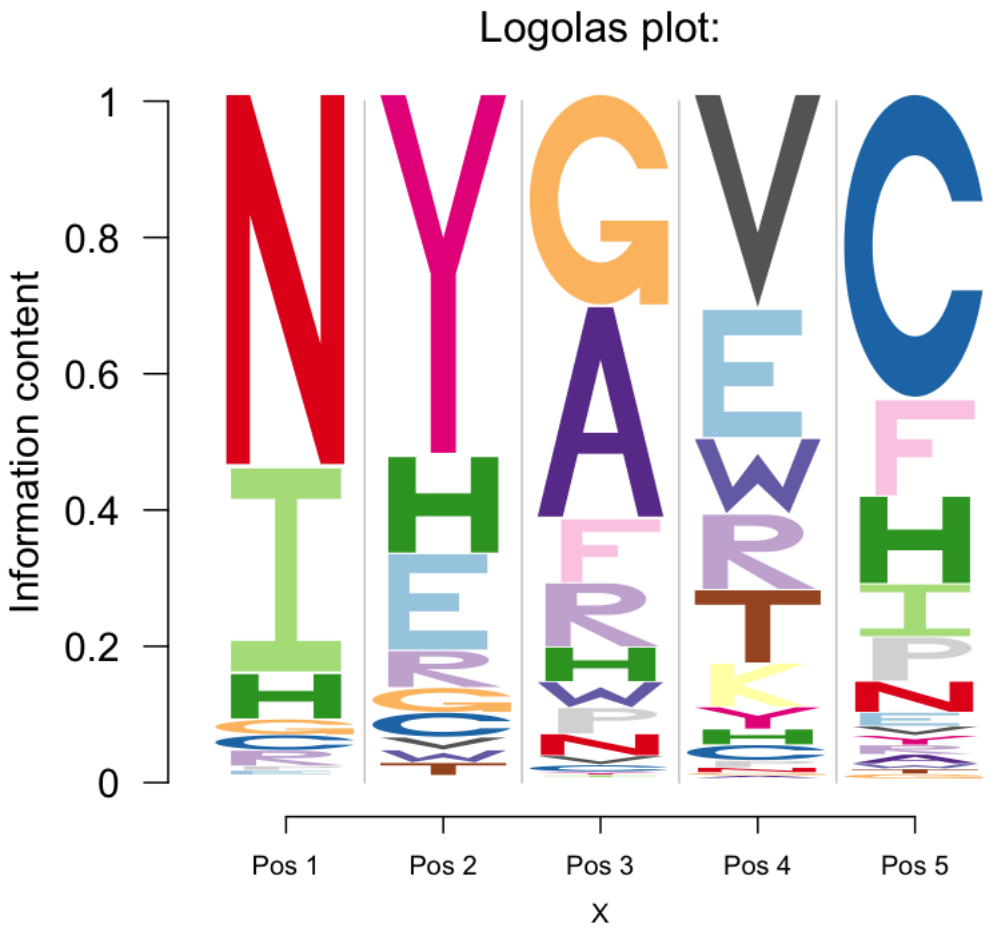
One can use *Logolas* for amino acid sequence motif detection as well, as the logo library of the software includes all the English alphabets and the 20 amino acids have a 1-letter representation using English alphabets.

```
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)
```



5.2 String Logos: Mutational Signature profiling

We now step beyond alphabet logos and present the first example of how a string can be used as a logo. Suppose we are provided with the mutational signature data for a particular tissue or cell type. A mutational signature is usually represented by the mutation type flanked by the bases adjacent to it. We provide a demo of how a mutation signature can be represented by **Logolas**.

```
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)
```

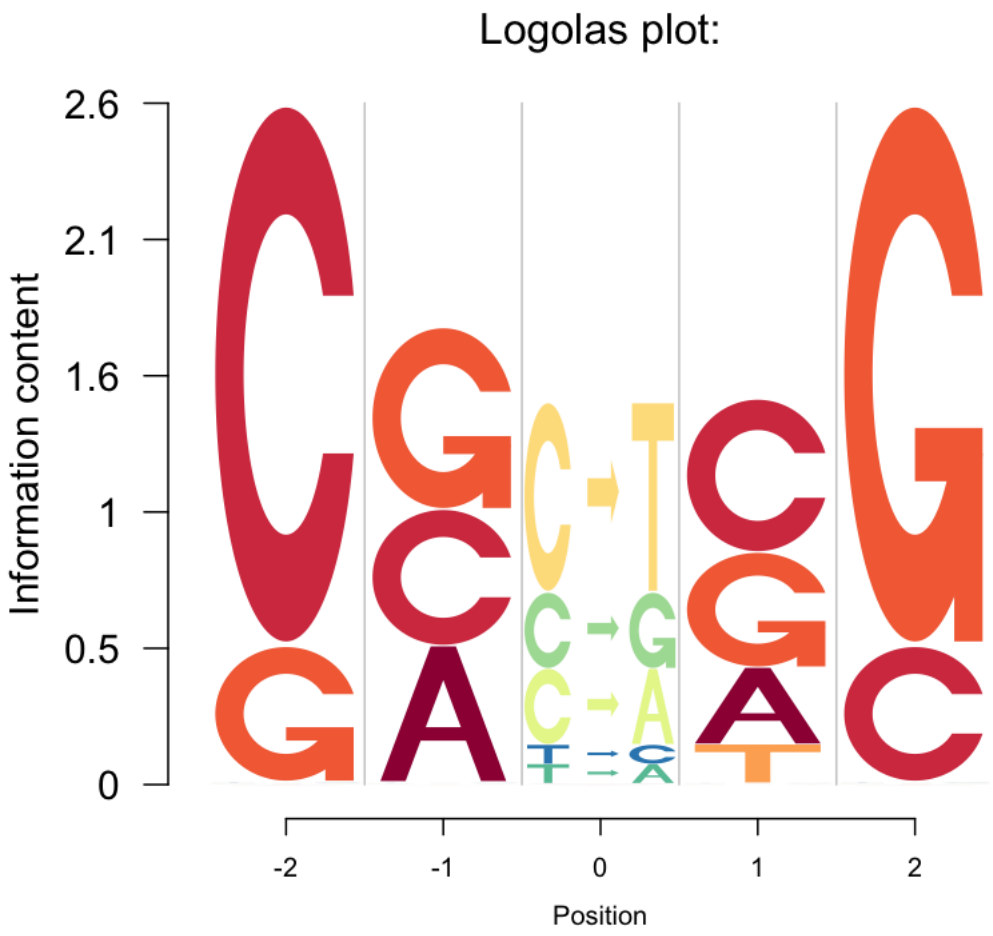
Note that we use the symbols `X>Y` to denote the $X \to Y$ substitutions. The data contains proportion of logos in each position - -2 left flanking, -1 left flanking, mutation, 1 right flanking and 2 right flanking. Note that `X>Y` type symbols occur only in the middle stack (column) as that is the mutation 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.

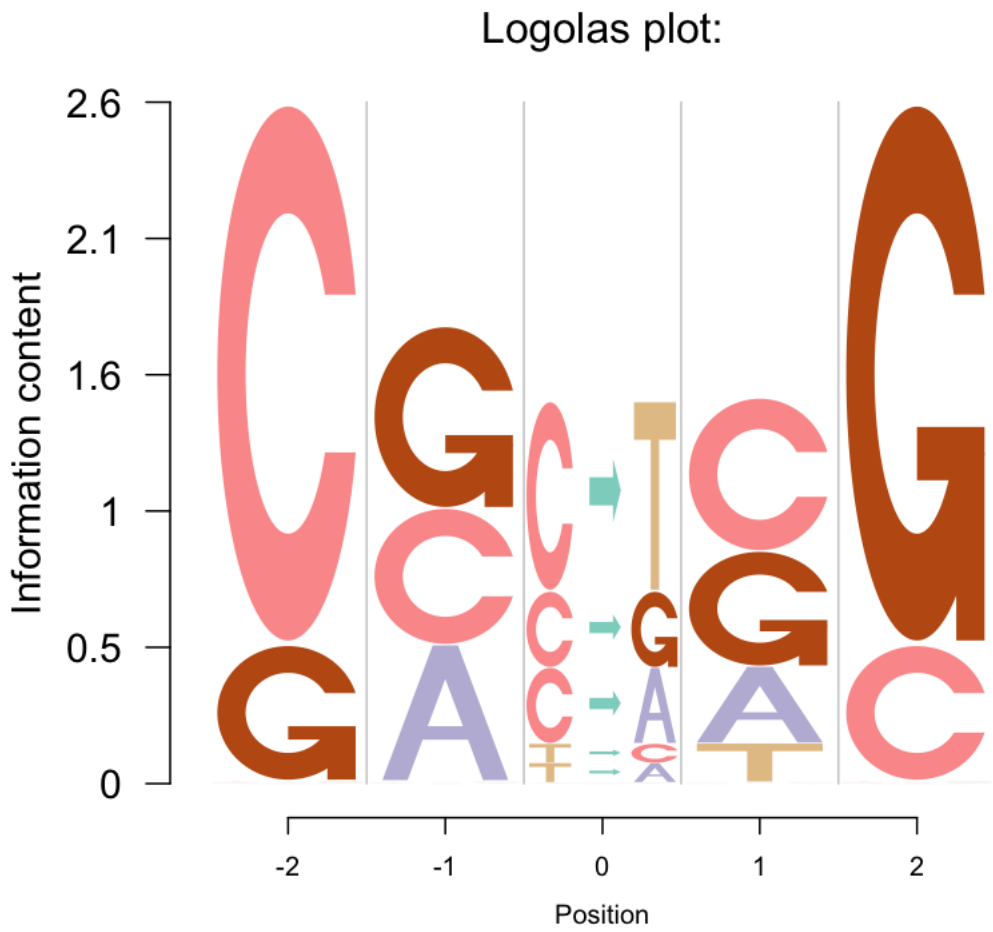
One issue with this plot is that the user may want to have the C in `C>T` to be of the same color, but here the symbol C and `C>T` are treated as separate entities. However Logolas coloring profile provides the user the flexibility to color each symbol instead of a string. We use the color type `per_symbol`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

Another coloring option is `per_column`, in which we have a specific color for a specific column.

```
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")
```



5.3 String Logos: Histone marks patterns

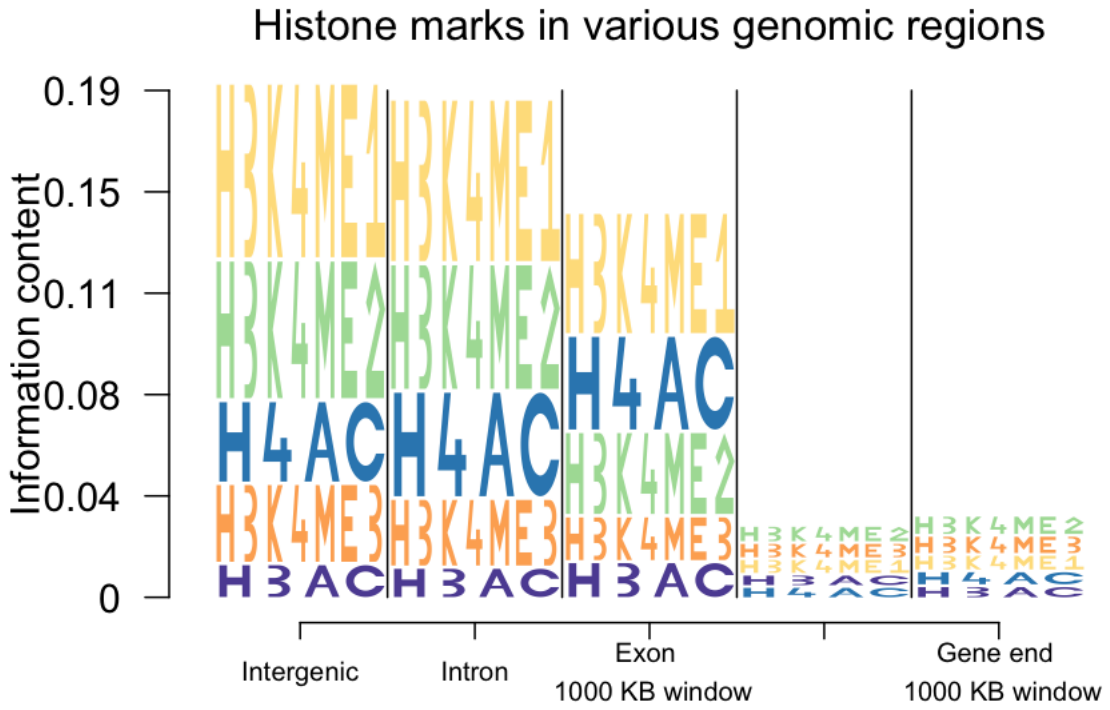
In studies related to histone marks, one might be interested to see if certain histone marks are prominent than others in some cell lines or tissues or in some genomic regions. In this case, we apply *Logolas* on an example data from Koch et al (2007) [Supp Table 2 of that paper]. The authors recorded 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")
```

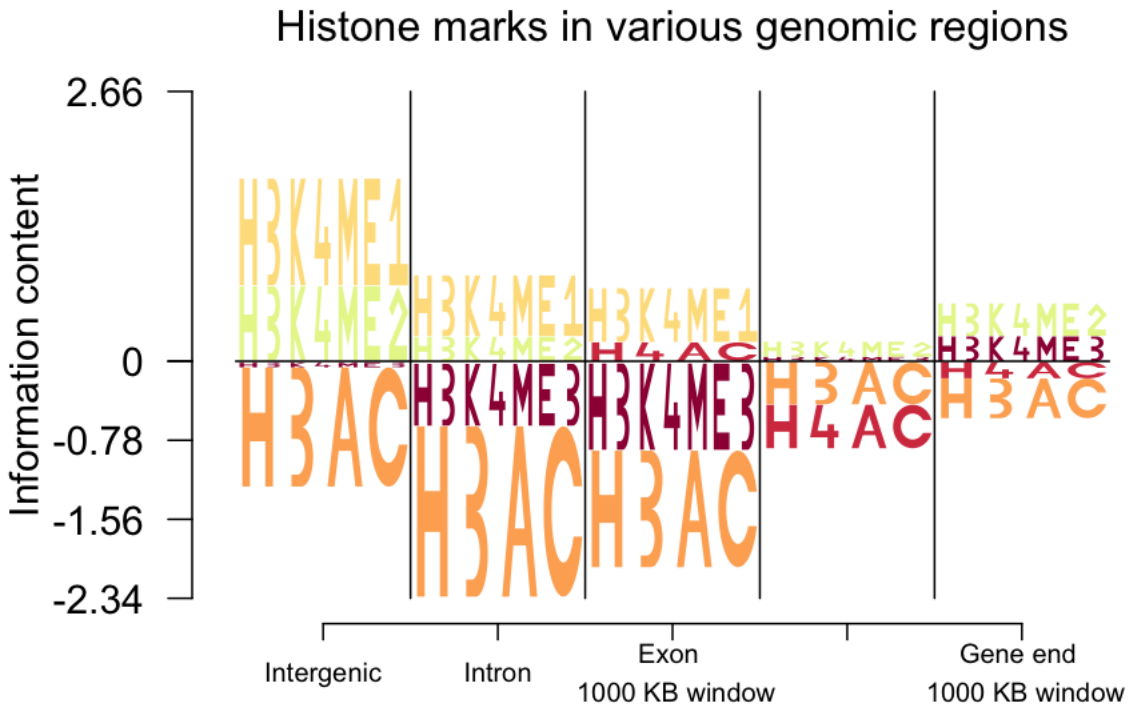
Note here that the histone mark symbols are alphanumeric, for example $H3K4ME1$. We now apply *Logolas* on this data.

```
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")
```



The *EDLogo* representation of the same table i presented below.

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

nlogomaker(mat,
           color_profile = color_profile,
           logoheight = "log",
           pop_name = "Histone marks in various genomic regions",
           xlab = "",
           ylab = "Information content",
           yscale_change = TRUE,
           col_line_split = "black")
```



6 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.

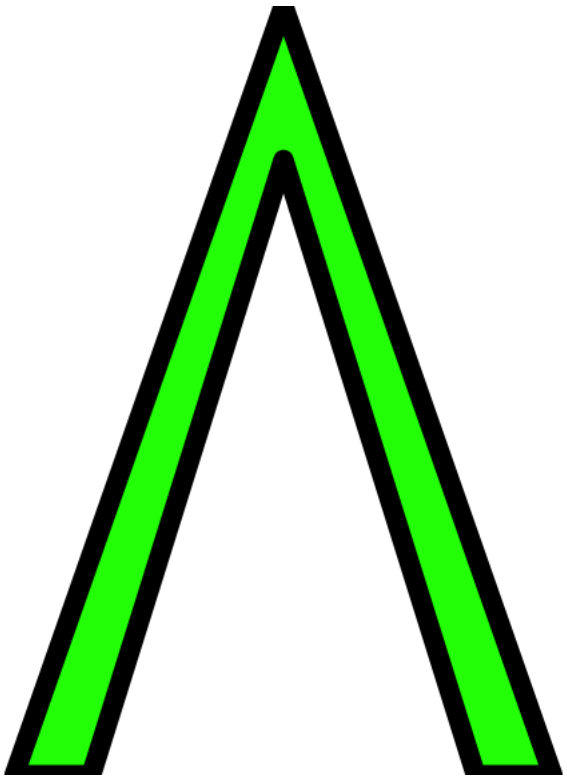
For example, if one wants to have the symbol Lambda as part of her logo, she can create it as follows

```
LAMBDAletter <- function(plot=FALSE,
                         fill_symbol = TRUE,
                         colfill="green",
                         lwd=10){
```

```
  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))

  colfill <- rep(colfill, length(unique(id)))

  if(plot){
    get_plot(x, y, id, fill, colfill, lwd = lwd, fill_symbol = fill_symbol)
  }

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

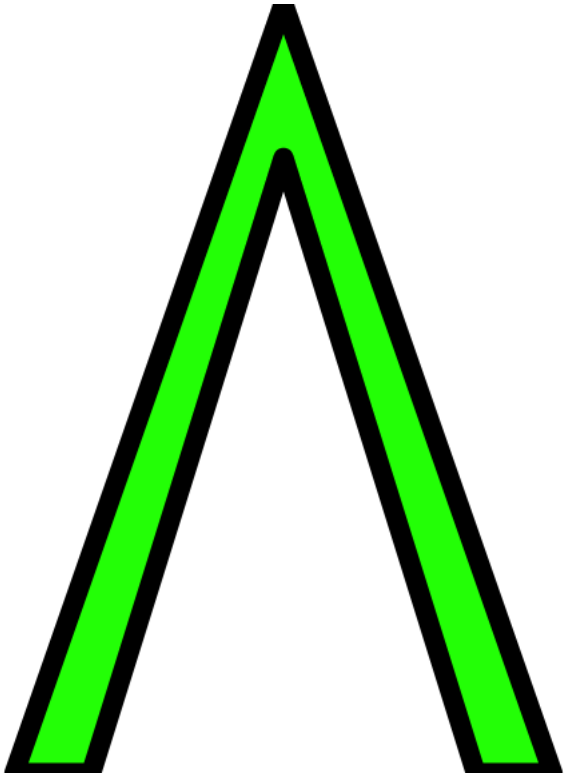
One can take a look at the symbol created in the following way.

```
lambda <- LAMBDAletter(plot=TRUE)
```



```
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))
```

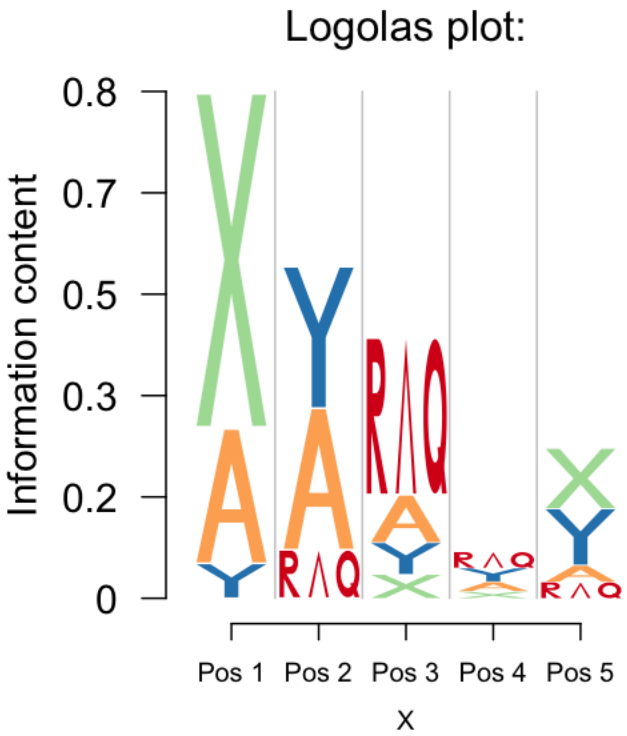


The function name has to be of the form '*letter"' where the user can be creative with the '"*"' part. Also the name must be in uppercase letters. The user can then add this symbol to the logo plot library and make logos of strings containing the above symbol.

7 Conclusion

Logolas provides a new logo representation called *EDLogo* that highlights both enrichment as well as depletion of symbols in a logo plot. It also allows the user to plot strings as logos, that extends the applicability of logo plots beyond DNA, RNA and protein sequences. We show a few examples of the string logo in modeling mutational signature profiles, histone marks composition and ecological species abundance compositions. *Logolas* also provides an adaptive scaling algorithm called Dirichlet adaptive shrinkage (dash) which scales the position weights for each position adapitvely taking into account the scales of the frequency of alignment at that position. In addition to the aforementioned features, *Logolas* also provides many novel customizations, fill and border styles, comptability of logo plots with ggplot2 graphics etc.

```
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")

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")
```



8 Acknowledgements

The authors 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 John Blischak, Kevin Luo, Hussein al Asadi and Alex White for helpful discussions.

9 Session Info

```
sessionInfo()
```

```
## R version 3.3.3 (2017-03-06)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: macOS Sierra 10.12.5
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] ggplot2_2.2.1  gridExtra_2.3  gridBase_0.4-7 Logolas_1.2.1  knitr_1.17
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.12       plyr_1.8.4         gtable_0.2.0       stats4_3.3.3
##  [5] magrittr_1.5       evaluate_0.10.1    scales_0.4.1       highr_0.6
##  [9] rlang_0.1.1.9000   stringi_1.1.5      lazyeval_0.2.0     labeling_0.3
## [13] BiocStyle_2.2.1    RColorBrewer_1.1-2 tools_3.3.3        stringr_1.2.0
## [17] munsell_0.4.3      seqLogo_1.40.0     colorspace_1.3-2   SQUAREM_2017.10-1
## [21] tibble_1.3.4
```

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.