

Supplementary Figure 2. Example Code Snippets for Visualizing Motif Alignments with motifStack

Contents

Supplementary Figure 2A-D. Examples of linear and radial dendrograms and code to generate them.	1
Supplementary Figure 2E-G. Merge motifs with different distance cutoffs and display these motif signatures in various layouts.	6
Supplementary Figure 2H-M. Use various color options to highlight different motif features.	10
Supplementary Figure 2N-O. Compare different Column Comparison Metrics (CCM) and alignment methods.	22
Supplementary Figure 2P. Improved alignment of fly HD family motifs using MatAlign compared to MotIV. The MatAlign alignment method is superior for discriminating between the closely related motifs in the fly HD family.	27
Supplementary Figure 2Q. Import PFM/PCMs files in Transfac, CisBP, or JASPAR format in batch mode.	31
Supplementary Figure 2R. Plot affinity logo from a position specific affinity matrix (PSAM).	33
Session information including the version of R , motifStack and other packages.	34

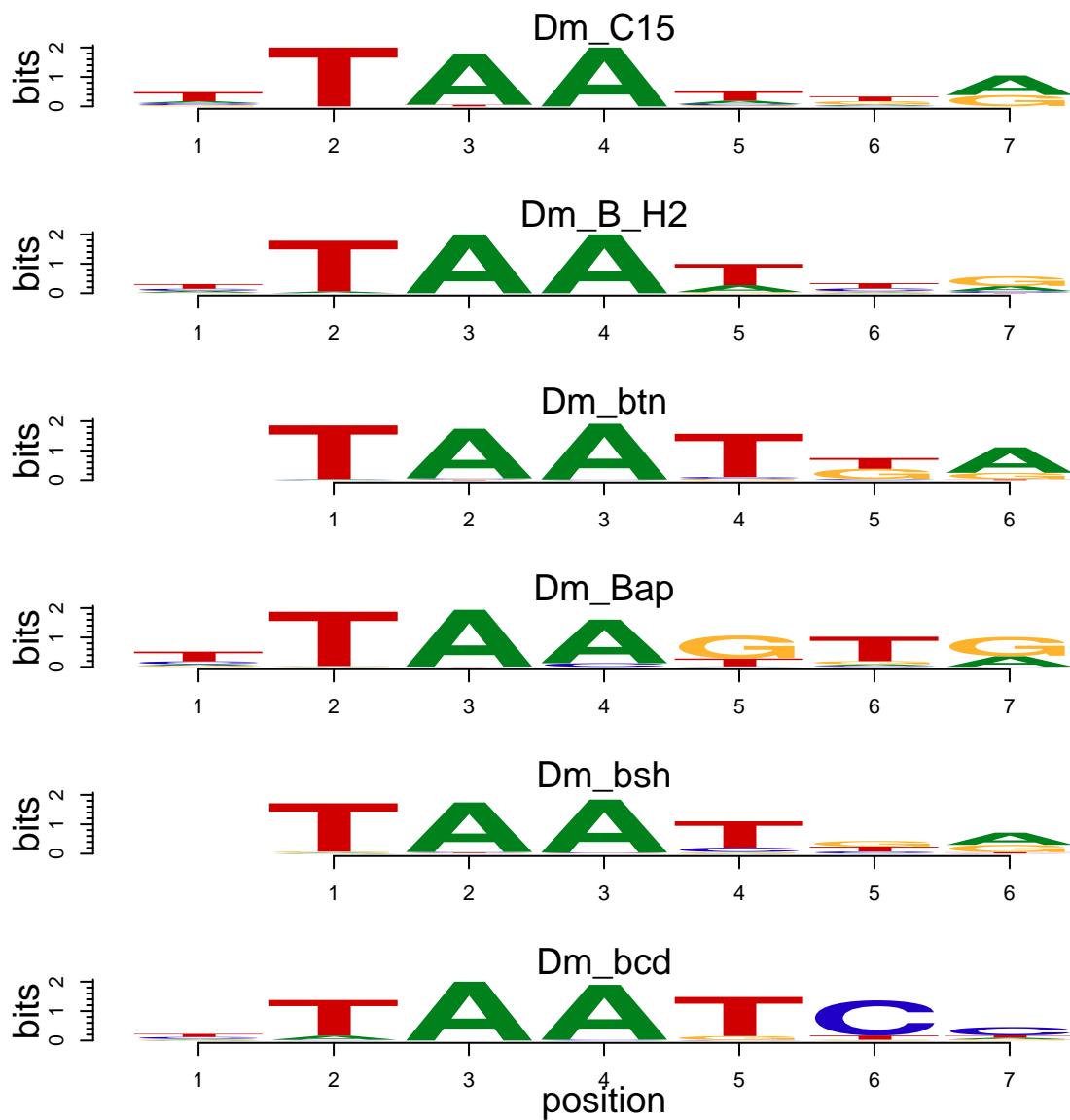
```
##load the library
library(motifStack)

##read pcms (position count matrices)
pcmpath <- "pcmsDatasetDM"
pcms <- readPCM(pcmpath)
##convert to pfms (position frequency matrices)
pfms<-lapply(pcms,pcm2pfm)
```

Supplementary Figure 2A-D. Examples of linear and radial dendrograms and code to generate them.

Plot motifs as a stack.

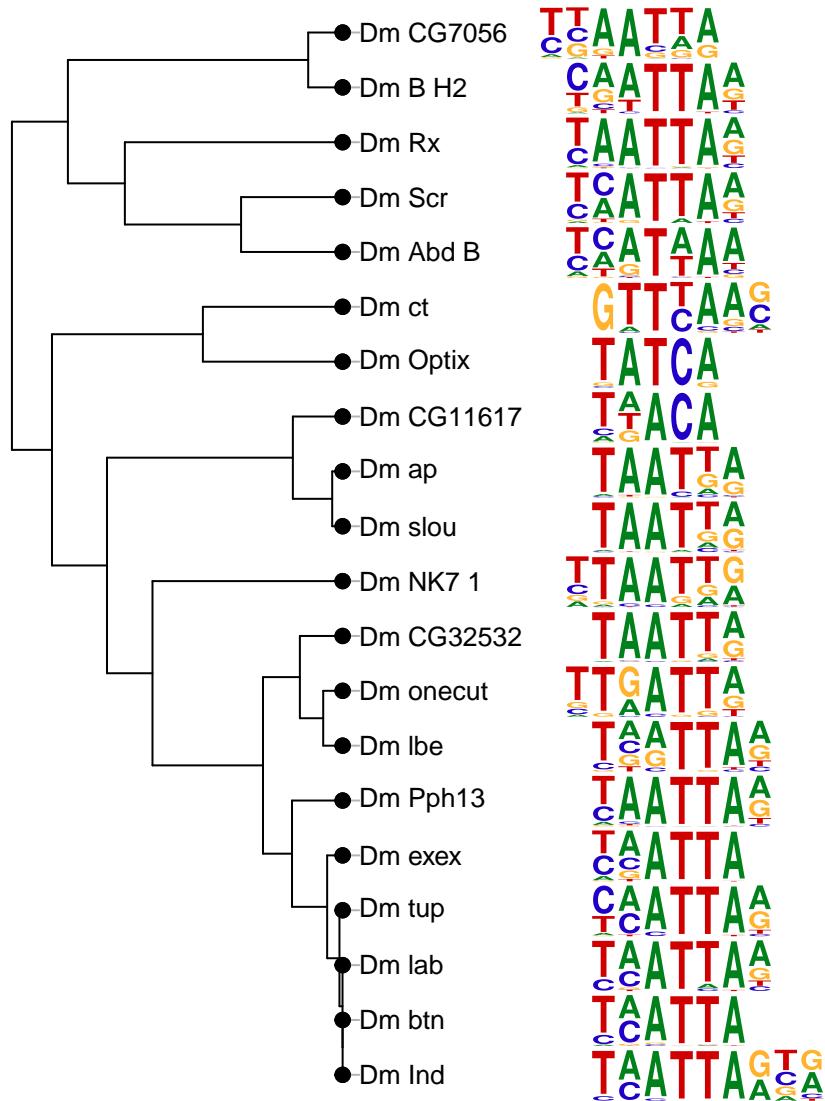
```
subset_pfms <- pfms[10:15]
motifStack(subset_pfms, layout="stack")
```



Supplementary Figure 2A: Plotted as a stack of motifs

Plot motifs as a linear tree with nucleotides drawn proportional to their frequency.

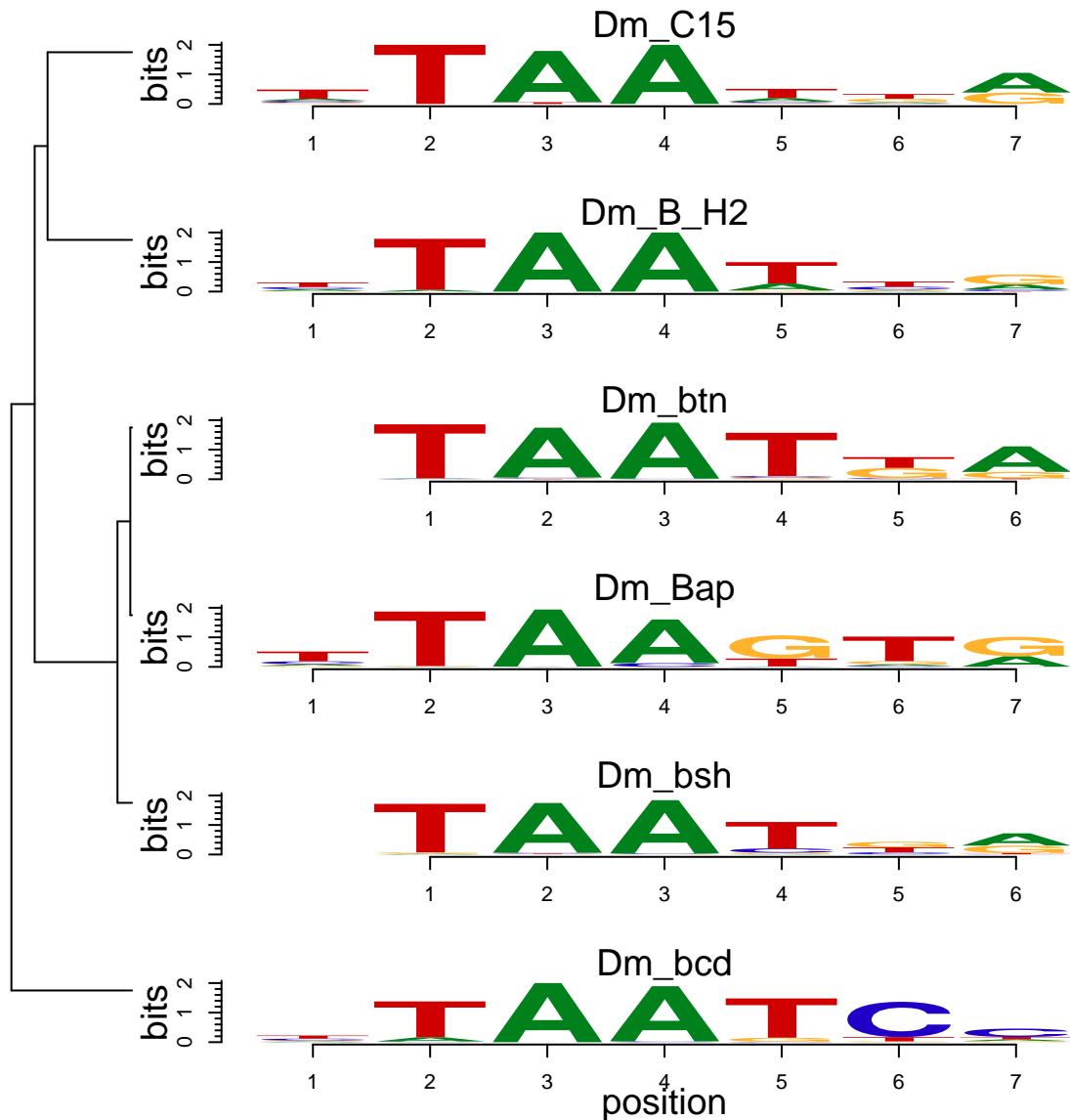
```
##try different style of sequence logo
motifStack(pfms[sample(1:length(pfms), 20)], layout="phylog", clabel.leaves=.8,
           f.logo=0.5, ic.scale=FALSE)
```



Supplementary Figure 2B: Plotted as a linear tree with nucleotides drawn proportional to their frequency

Plot motifs as a linear tree depicting information content (IC).

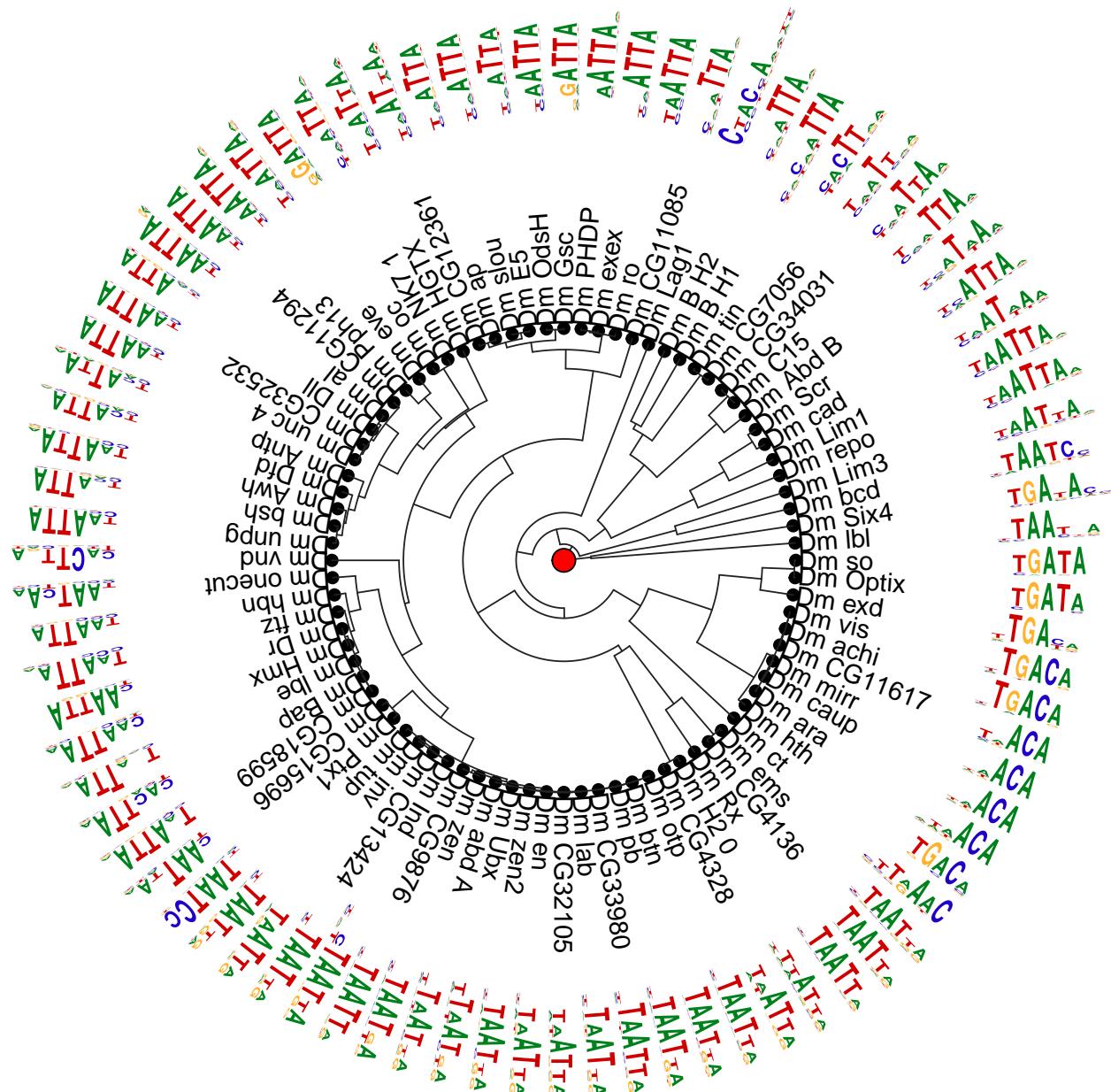
```
##By default, MotIV is used for clustering  
motifStack(subset_pfms, layout="tree", trueDist=TRUE)
```



Supplementary Figure 2C: Plotted as a linear tree with IC

Plot motifs as a radial tree.

```
motifStack(pfms, layout="radialPhylog")
```



Supplementary Figure 2D: Plotted as a radial tree

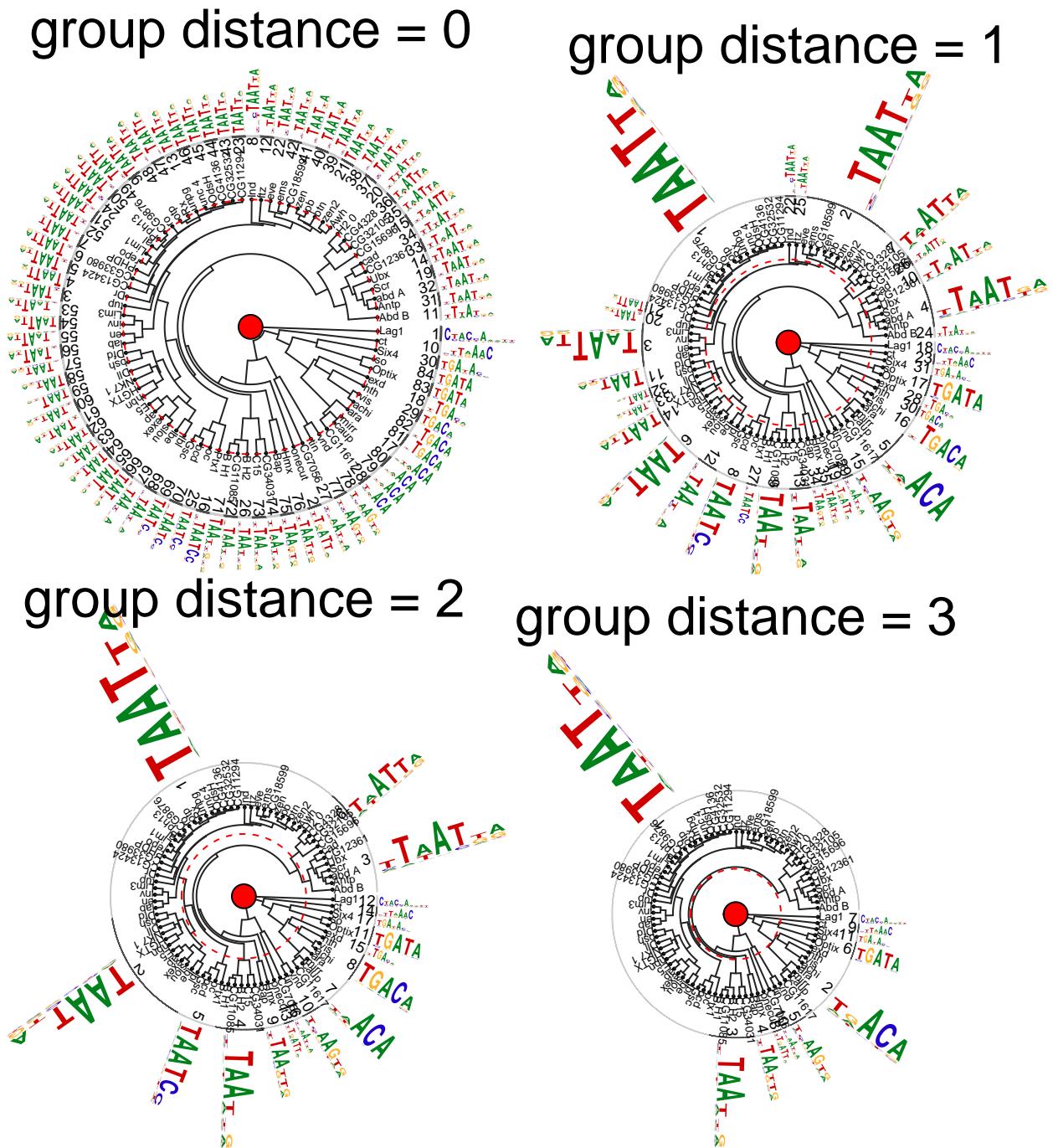
Supplementary Figure 2E-G. Merge motifs with different distance cutoffs and display these motif signatures in various layouts.

Before motifs are merged, the distances of the motifs are calculated using STAMP, MovIV or MatAlign. Here we show different ways to display the merged motifs and how different distance cutoffs affect the resulting motif signatures. The red dotted line indicates the distance cutoff used.

Merge motifs using different distance cutoffs.

```
outpath <- "output"
malign_path <- "./app/malign-v4a"
neighbor_path <- "./app/neighbor.app/Contents/MacOS/neighbor"
MatAlign2tree_path <- "./MatAlign2tree.pl"
pcmpath <- "pcmsDatasetDM"
pcms <- readPCM(pcmpath)
pfms<-lapply(pcms,pcm2pfm)
system(paste("perl", MatAlign2tree_path, "--in . --pcmpath", pcmpath,
            "--out", outpath,
            "--malign", malign_path,
            "--neighbor", neighbor_path,
            "--tree", "UPGMA"))
newickstrUPGMA <- readLines(con=file.path(outpath, "NJ.malign.distMX.nwk"))
phylogUPGMAmatAlign <- newick2phylog(newickstrUPGMA, FALSE)
##get the leaves of tree for ordering the pfms
matAlignLeaveNames <- names(phylogUPGMAmatAlign$leaves)
this_motifs <- pfms[matAlignLeaveNames]
matAlignLeaveNames <- gsub("^Dm_", "", matAlignLeaveNames)

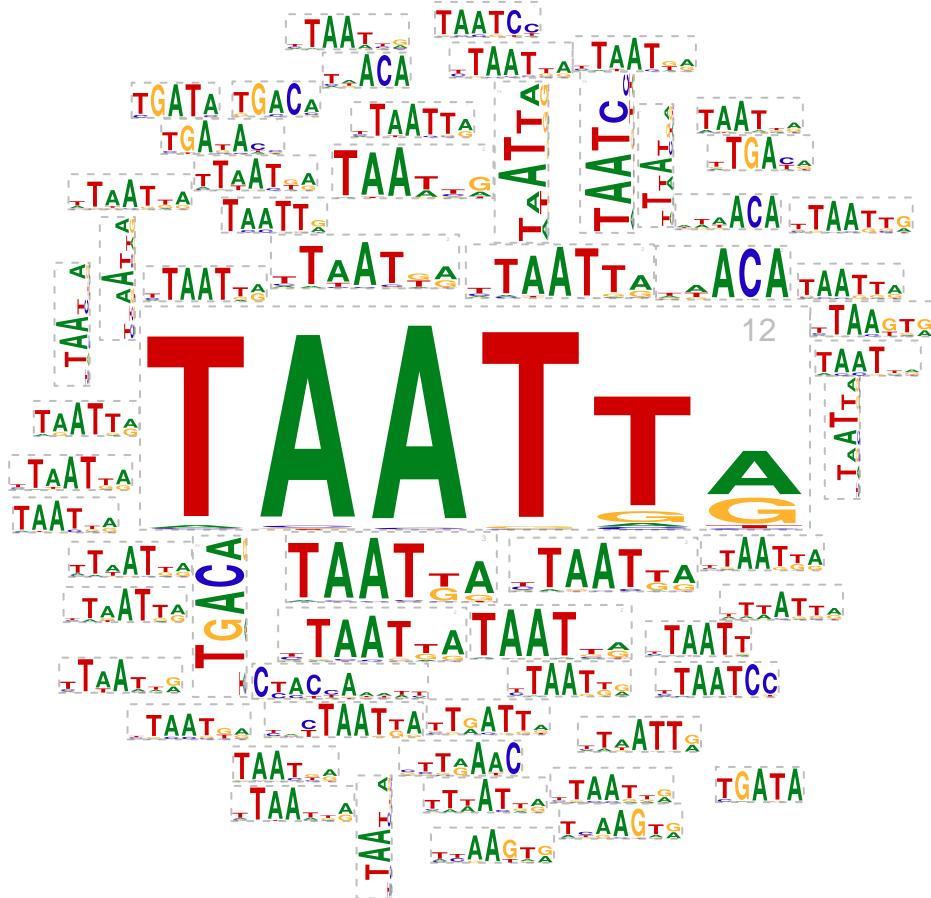
for(groupDistance in c(0, 1, 2, 3)){
  motifSig <- motifSignature(this_motifs,
                               phylogUPGMAmatAlign,
                               groupDistance=groupDistance,
                               min.freq=1)
  this_sig <- signatures(motifSig)
  ## get color set for the signature groups
  this_gpCol <- sigColor(motifSig)
  plotMotifStackWithRadialPhylog(phylog=phylogUPGMAmatAlign,
                                  pfms=this_sig,
                                  col.inner.label.circle=this_gpCol,
                                  inner.label.circle.width=0.02,
                                  labels.leaves=matAlignLeaveNames,
                                  cleaves=.2, circle=1, circle.motif=1.5,
                                  xlabel.leaves=.4, motifScale="logarithmic",
                                  angle=358, plotIndex=TRUE, IndexCex=.6,
                                  groupDistance=groupDistance)
  text(0, 2.3, label=paste("group distance =", groupDistance), cex=2)
}
```



Supplementary Figure 2E: Motif signature plotted as a radial tree by different distance

Plot motif signature as a word cloud in circle.

```
## get signature
gpDist <- c(.5, 1)
motifSig <- lapply(gpDist, motifSignature,
                    pfms=this_motifs,
                    phylog=phylogUPGMAmatAlign,
                    min.freq=1)
## motif cloud, cloud style
motifCloud(motifSig[[1]], layout="cloud", scale=c(9, .75))
```



Supplementary Figure 2F: Motif signature plotted as a word cloud with motif size representing the number of motifs contributed to the signature. The larger the size, the larger the number of motifs that share the motif signature

Plot motif signature as a word cloud in rectangle.

```
## motif cloud, rectangle style
for(i in 1:2){
  motifCloud(motifSig[[i]], layout="rectangles", ic.scale=FALSE)
  op <- par(mar = c(0, 0, 0, 0))
  text(.5, .985, label= paste("group distance =", gpDist[i]))
  par(op)
}
```



Supplementary Figure 2G: Motif signature plotted as a word cloud in rectangular by different distances

Supplementary Figure 2H-M. Use various color options to highlight different motif features.

motifStack offers five options to color a radial tree: the background of the inner circle, the text of motif names, the background of the motif names, the inner label ring and the outer label ring. Here are a few examples using these options to show the motif's data source, the computational algorithm that generated the motifs, the motif's information content (IC) and the TF.

```
getPairColor <- function(n=10L){
  if(n %% 2 != 0) n <- n+1
  n <- n/2
  n <- rainbow(n)#as.character(t(matrix(rainbow(n=n), ncol=2, byrow=FALSE)))
  n2 <- highlightCol(n, .5)
  as.character(t(cbind(n, n2)))
}

pairColor <- getPairColor(22)

pfmList2matrixList <- function(pfms){
  m <- lapply(pfms, function(.ele) as(.ele, "matrix"))
  names(m) <- unlist(lapply(pfms, function(.ele) .ele@name))
  m
}

getMotIVOut <- function(pfms, cc, align){
  jaspar.scores <-
    MotIV::readDBScores(
      file.path(".", "app", "scores",
                paste("JaspRand_", cc, "_", align, ".scores", sep="")))
  d <- MotIV::motifDistances(pfmList2matrixList(pfms), cc=cc, align=align)
  hc <- MotIV::motifHclust(d, method="average")
  phylog <- hclust2phylog(hc)
  pfms <- pfms[hc$order]
  aligned.pfms <- DNAmotifAlignment(pfms)
  leaveNames <- names(phylog$leaves)
  ## data source
  dataSource <- factor(grep("_M", leaveNames))
  levels(dataSource) <- c("yellow", "blue") ##c("Uniprobe", "CIS-BP")
  dataSource <- as.character(dataSource)
  ## algorithm
  algorithm <-
    factor(!grep("_M", leaveNames)) + grep("_bml", leaveNames))
  levels(algorithm) <-
    c("blue", "brown", "orange") ##("DREAM5", "Seed-And-Wobble", "BEEML")
  algorithm <- as.character(algorithm)
  ## motifs from same PBM data
  motifGroup <- factor(gsub("(.*?).*$", "\\\1", leaveNames))
  levels.motifGroup <- levels(motifGroup)
  levels(motifGroup) <- pairColor[1:length(levels(motifGroup))]
  colors.motifGroup <- levels(motifGroup)
  motifGroup <- as.character(motifGroup)
  return(list(aligned.pfms=aligned.pfms, unaligned.pfms=pfms, phylog=phylog,
             leaveNames=leaveNames,
             dataSource=dataSource,
             algorithm=algorithm,
```

```

        motifGroup=motifGroup,
        levels.motifGroup=levels.motifGroup,
        colors.motifGroup=colors.motifGroup))
}
##read_pcms
pcmpath <- "pcmsDatasetAlgorithm"
pcms <- readPCM(pcmpath)
##convert to pfms
pfms<-lapply(pcms,pcm2pfm)

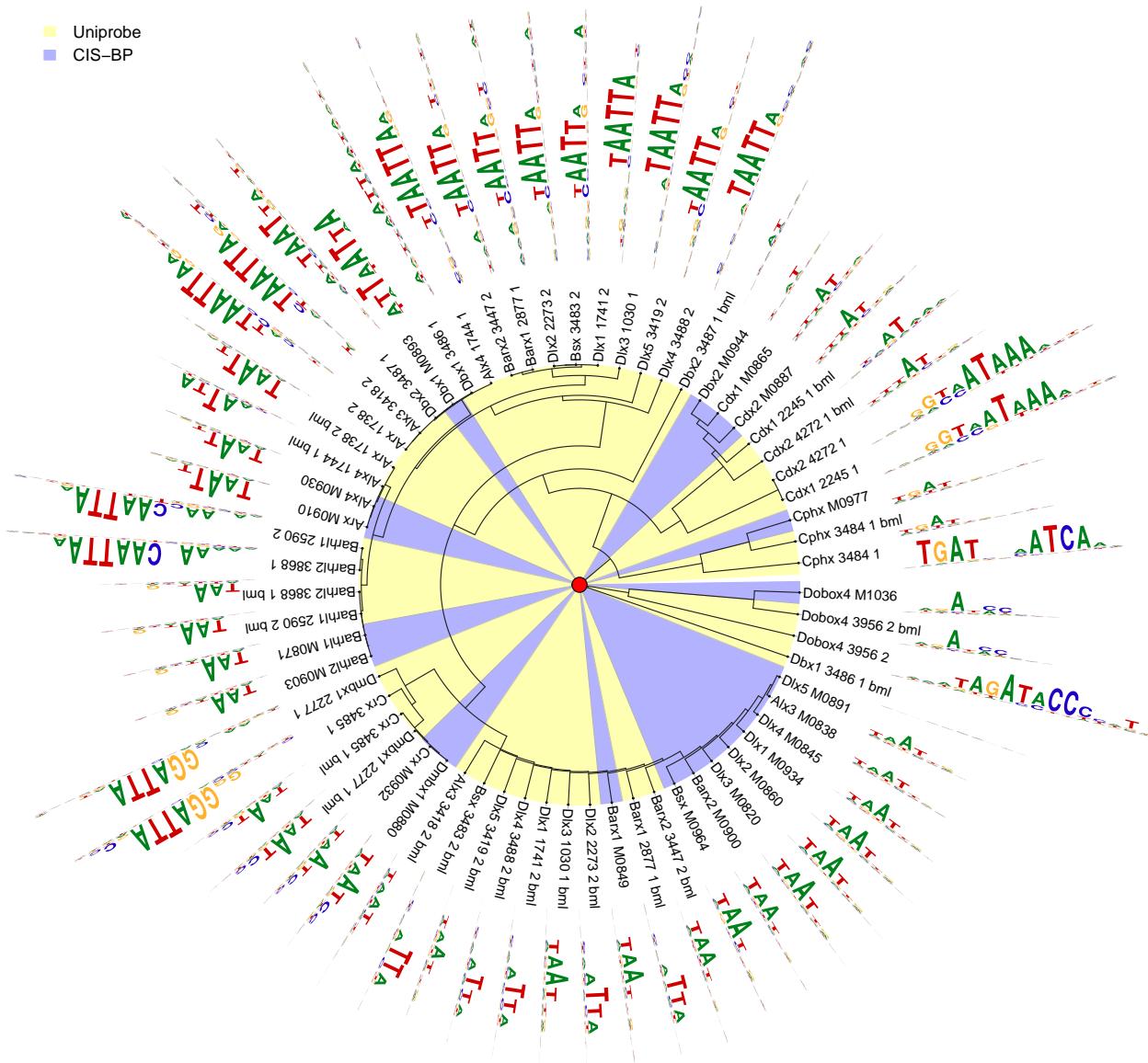
```

Use the color of the background of inner circle to distinguish different data sources with the parameter col.bg.

```

motIVout <- getMotIVOut(pfms, "PCC", "SWU")
attach(motIVout)
plotMotifStackWithRadialPhylog(phylog=phylog, pfms=unaligned.pfms,
                                labels.leaves=leaveNames,
                                col.bg=dataSource, col.bg.alpha=.3,
                                cleaves=.2, circle=1.1, circle.motif=1.6,
                                clabel.leaves=.8, angle=358)
legend(-2.3, 2.4, legend=c("Uniprobe", "CIS-BP"),
       fill= highlightCol(c("yellow", "blue"), alpha=.3),
       border="white", lty=NULL, bty = "n", cex=1)

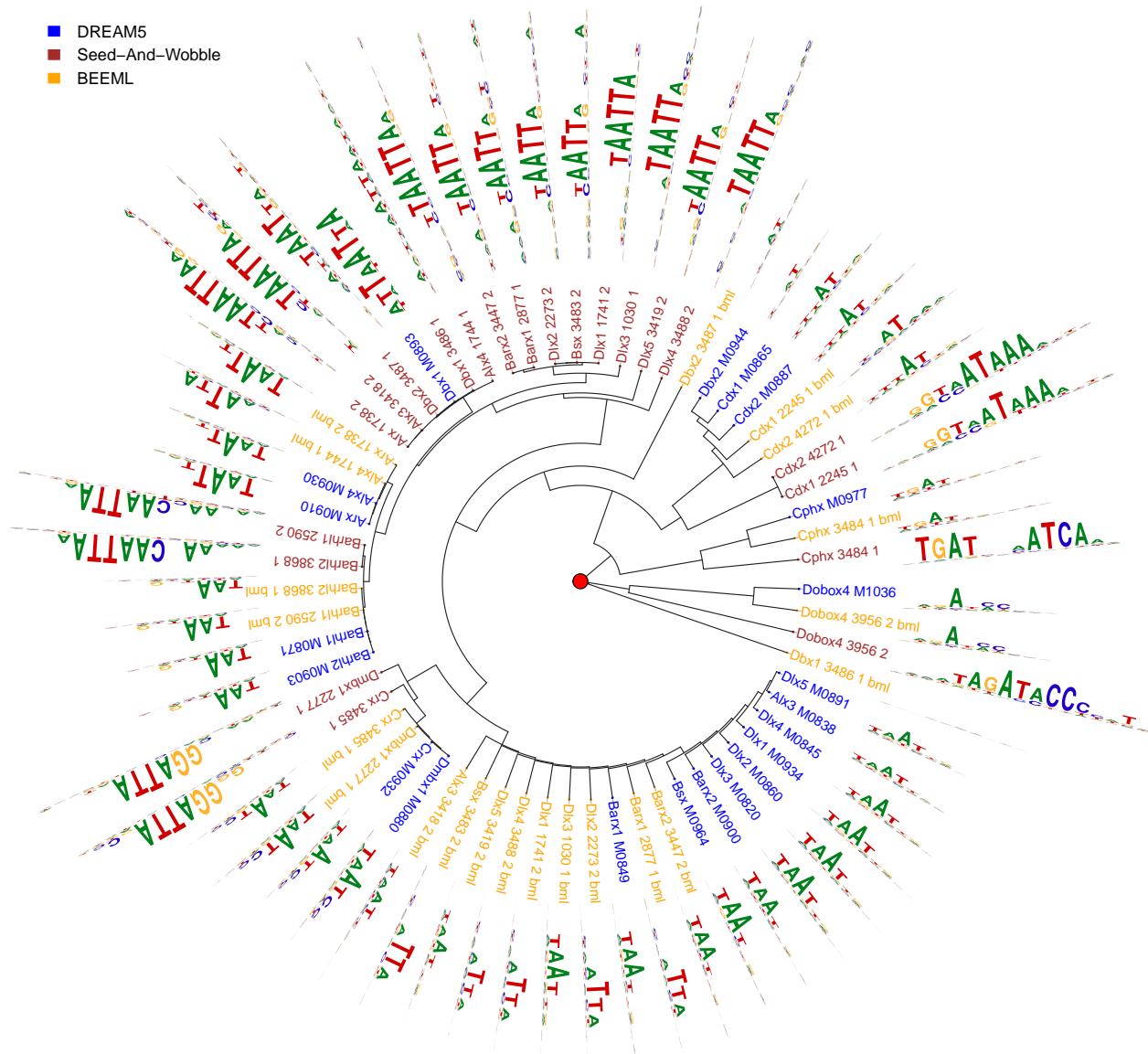
```



Supplementary Figure 2H: Use the color of the background of inner circle to distinguish different data sources with the parameter col.bg

Use the color of text of motif names to distinguish the computational algorithms used with the parameter col.leaves.

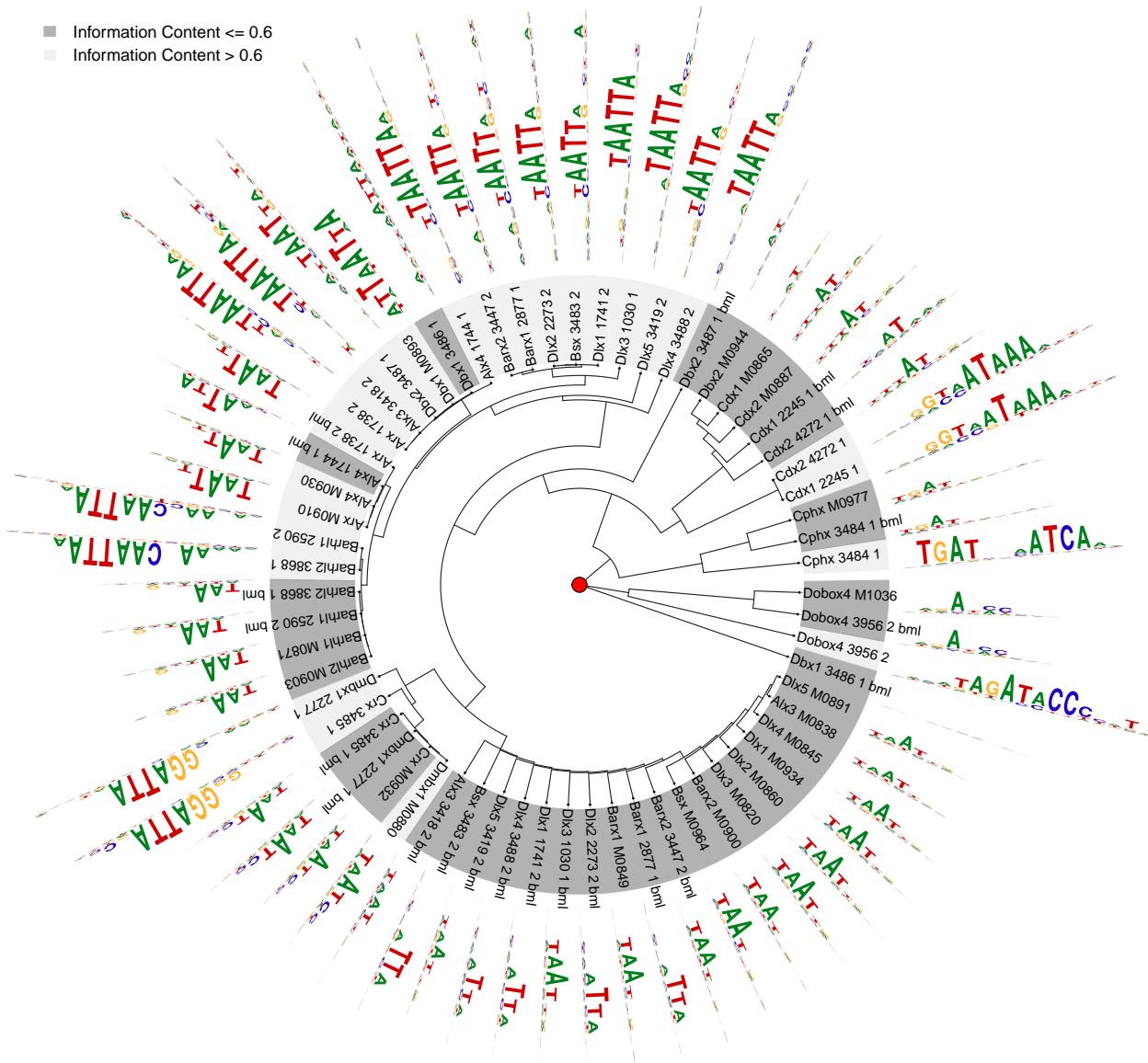
```
plotMotifStackWithRadialPhylog(phylog=phylog, pfms=unaligned.pfms,
                                labels.leaves=leaveNames,
                                col.leaves=algorithm,
                                cleaves=.2, circle=1.1, circle.motif=1.6,
                                clabel.leaves=.8, angle=358)
legend(-2.3, 2.4, legend=c("DREAM5", "Seed-And-Wobble", "BEEML"),
       fill= c("blue", "brown", "orange"),
       border="white", lty=NULL, bty = "n", cex=1)
```



Supplementary Figure 2I: Use the color of the text of motif names to separate the computational algorithms used with the parameter col.leaves

Use the color of the background of the motif names to distinguish motifs with high and low IC with the parameter col.leaves.bg.

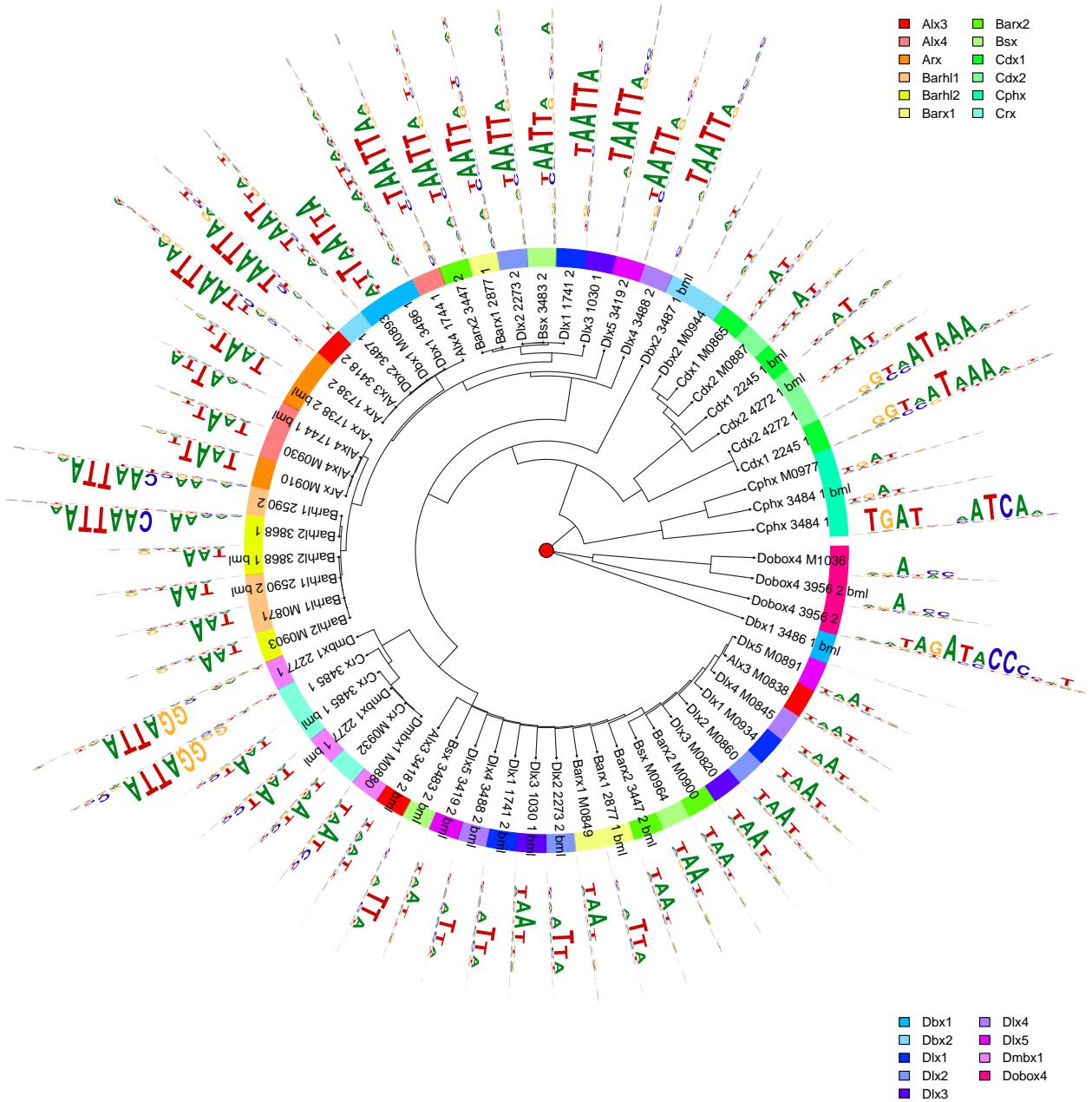
```
icgp <- ifelse(sapply(sapply(unaligned.pfms, getIC), mean) > 0.6,
                "lightgray", "black")
plotMotifStackWithRadialPhylog(phylog=phylog, pfms=unaligned.pfms,
                                labels.leaves=leaveNames,
                                col.leaves.bg=icgp,
                                col.leaves.bg.alpha=.3,
                                cleaves=.2, circle=1.1, circle.motif=1.6,
                                clabel.leaves=.8, angle=358)
legend(-2.3, 2.4,
       legend=c("Information Content <= 0.6", "Information Content > 0.6"),
       fill= highlightCol(c("black", "lightgray"), alpha=.3),
       border="white", lty=NULL, bty = "n", cex=1)
```



Supplementary Figure 2J: Use the color of the background of the motif names to distinguish motifs with high and low IC with the parameter col.leaves.bg

Use the color of the inner ring of the motif circle to distinguish the TF with the parameter col.inner.label.circle.

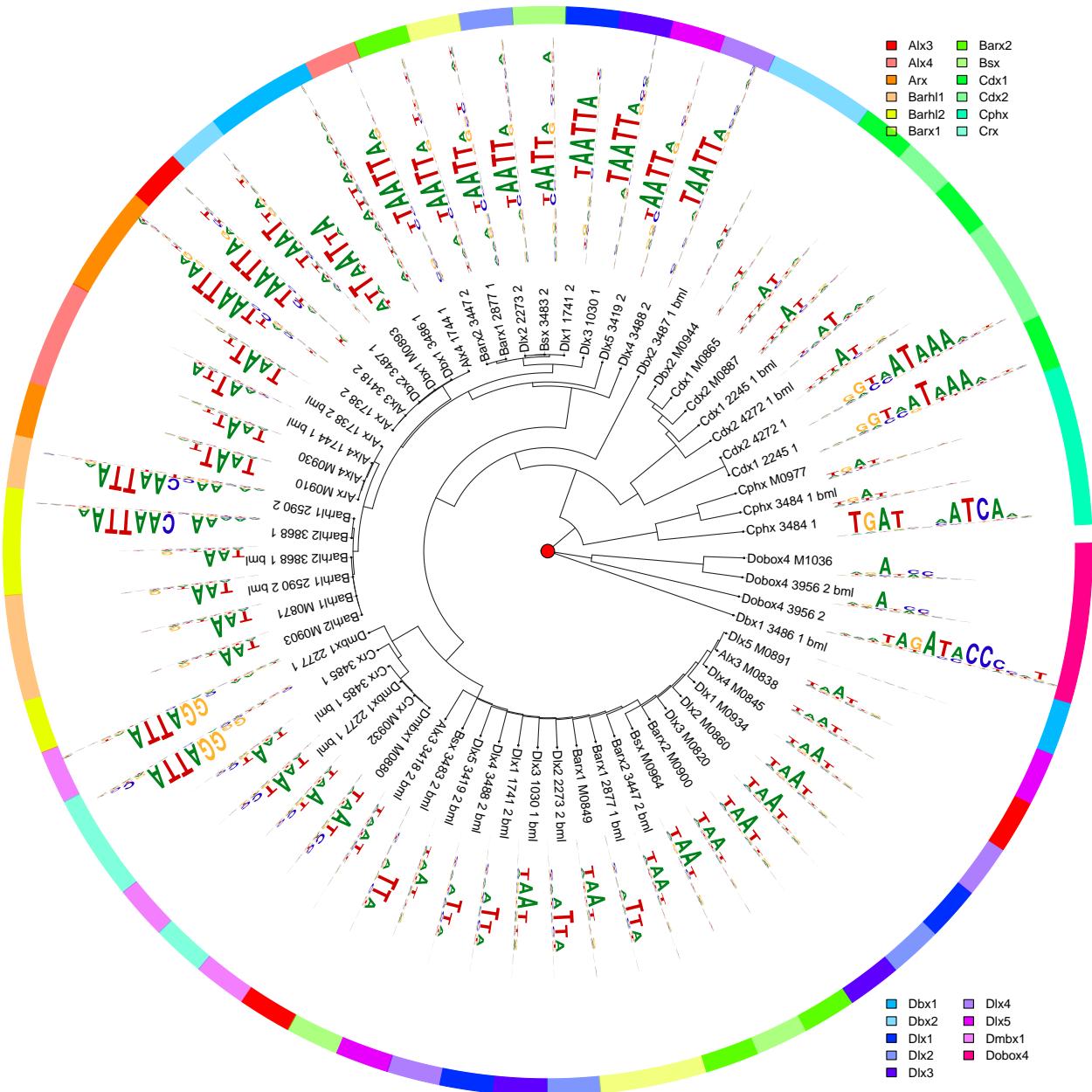
```
plotMotifStackWithRadialPhylog(phylog=phylog, pfms=unaligned.pfms,
                                labels.leaves=leaveNames,
                                col.inner.label.circle=motifGroup,
                                inner.label.circle.width=0.1,
                                cleaves=.2, circle=1.1, circle.motif=1.6,
                                clabel.leaves=.8, angle=358)
legend(1.5, 2.4, legend=levels.motifGroup[1:12],
       fill= colors.motifGroup[1:12],
       border="black", lty=NULL, bty = "n", ncol=2, cex=.8)
legend(1.5, -2, legend=levels.motifGroup[13:21],
       fill= colors.motifGroup[13:21],
       border="black", lty=NULL, bty = "n", ncol=2, cex=.8)
```



Supplementary Figure 2K: Use the color of the inner ring of the motif circle to distinguish the TFs with the parameter col.inner.label.circle

Use the color of the outer ring of the motif circle to distinguish the TFs with the parameter col.outer.label.circle.

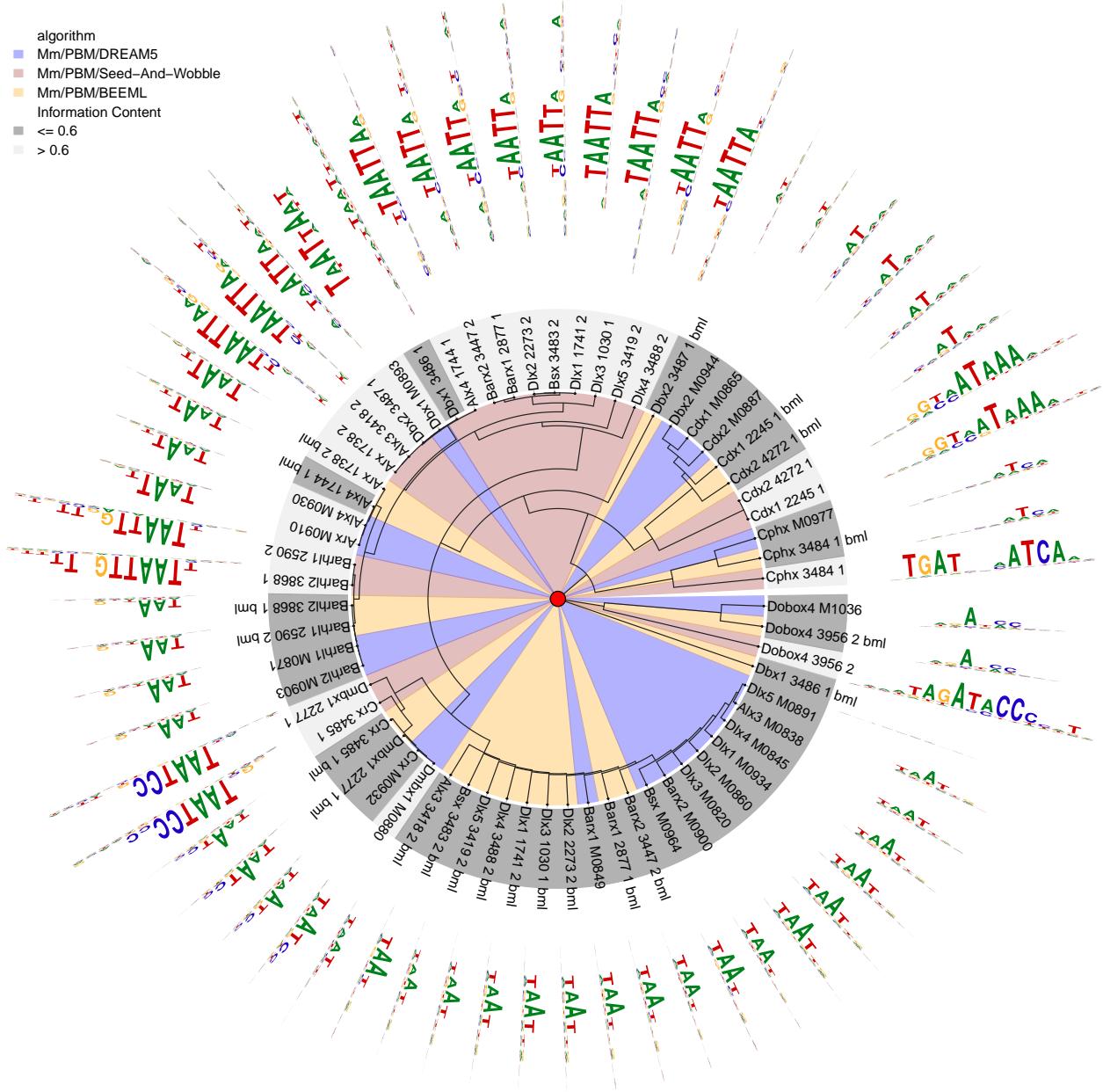
```
plotMotifStackWithRadialPhylog(phylog=phylog, pfms=unaligned.pfms,
                                labels.leaves=leaveNames,
                                col.outer.label.circle=motifGroup,
                                outer.label.circle.width=0.1,
                                cleaves=.2, circle=1.1, circle.motif=1.6,
                                clabel.leaves=.8, angle=358)
legend(1.5, 2.4, legend=levels.motifGroup[1:12],
       fill= colors.motifGroup[1:12],
       border="black", lty=NULL, bty = "n", ncol=2, cex=.8)
legend(1.5, -2, legend=levels.motifGroup[13:21],
       fill= colors.motifGroup[13:21],
       border="black", lty=NULL, bty = "n", ncol=2, cex=.8)
```



Use a combination of coloring options to visualize multiple motif features.

```
plotMotifStackWithRadialPhylog(phylog=phylog, pfms=aligned.pfms,
                                labels.leaves=leaveNames,
                                col.bg=algorithm, col.bg.alpha=.3,
                                col.leaves.bg=icgp,
                                col.leaves.bg.alpha=.3,
                                cleaves=.2, circle=1.1, circle.motif=1.6,
                                clabel.leaves=.8, angle=358)
legend(-2.3, 2.4,
       legend=c("algorithm", "Mm/PBM/DREAM5", "Mm/PBM/Seed-And-Wobble",
               "Mm/PBM/BEEML", "Information Content", "<= 0.6", "> 0.6"),
       fill= c("white",
              highlightCol(c("blue", "brown", "orange"), alpha=.3),
              "white", highlightCol(c("black", "lightgray"), alpha=.3)),
       border="white", lty=NULL, bty = "n", cex=.8)

detach(motIVout)
```



Supplementary Figure 2M: Use a combination of coloring options to visualize multiple motif feature

Supplementary Figure 2N-O. Compare different Column Comparison Metrics (CCM) and alignment methods.

Here we used a subset of mouse HD TFs with PBM data. For each TF, we generated three different motifs using three different computational methods. We compared how well different CCM and alignment methods could group together the motifs from the same TFs. This example illustrates how to set the CCM and the alignment method, using MotIV or MatAlign, and that different methods can substantially affect the resulting alignment as visualized in motifStack.

When clustering using MotIV, Pearson Correlation Coefficient (PCC) or Average Log Likelihood Ratio (ALLR) was used as the CCM, and Smith-Waterman Ungapped (SWU) or Needleman-Wunsch (NW) was used as the alignment method. When clustering the motifs using MatAlign, ALLR and SWU were used as CCM and alignment respectively.

To run the MatAlign examples, phylib (<http://evolution.genetics.washington.edu/phylib/progs.data.dist.html>) and MatAlign (<http://storno.wustl.edu/MatAlign/>) need to be installed first.

Visualize motif alignments generated by MotIV with different CCM and alignment methods.

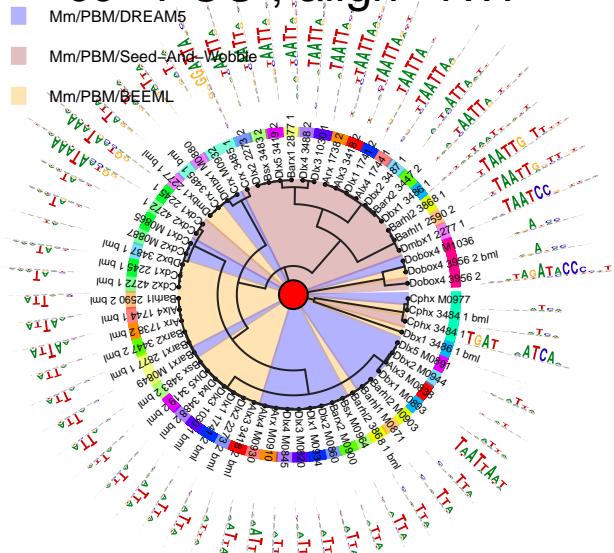
```

sta <- mapply(function(cc, align){
  motIVout <- getMotIVOut(pfms, cc, align)
  attach(motIVout)
  plotMotifStackWithRadialPhylog(phylog=phylog, pfms=aligned.pfms,
    labels.leaves=leaveNames,
    col.bg=algorithm, col.bg.alpha=.3,
    col.inner.label.circle=motifGroup,
    inner.label.circle.width=0.1,
    cleaves=.2, circle=1.1, circle.motif=1.6,
    clabel.leaves=.3, angle=358)

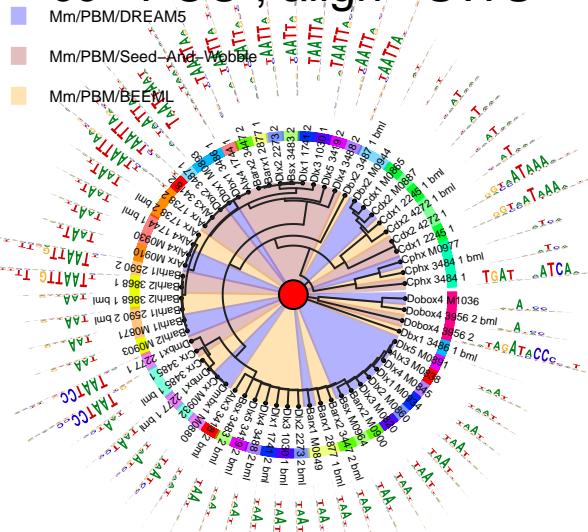
  legend(-2.3, 2.4,
    legend=c("Mm/PBM/DREAM5", "Mm/PBM/Seed-And-Wobble",
      "Mm/PBM/BEEMI"),
    fill= highlightCol(c("blue", "brown", "orange"), alpha=.3),
    border="white", lty=NULL, bty = "n", cex=.5)
  text(0, 2.3, label=paste("cc=", cc, "; align=", align), cex=1.5)
  cnt <- rle(motifGroup)
  cnt <- split(algorithm, rep(1:length(cnt$lengths), cnt$lengths))
  cnt <- table(sapply(cnt, function(.ele) length(unique(.ele))))
  cnt.1 <- vector("integer", 3)
  names(cnt.1) <- 1:3
  cnt.1[names(cnt)] <- cnt
  cnt.1["1"] <- (cnt.1["1"]+cnt.1["2"]*2+cnt.1["3"]*3)/3 - cnt.1["2"] - cnt.1["3"]
  text(0, -2.3,
    label=paste(names(cnt.1), cnt.1, sep=":", collapse="; "), cex=1.5)
  detach(motIVout)
}, c("PCC", "PCC", "ALLR", "ALLR"), c("NW", "SWU", "NW", "SWU"))

```

cc= PCC ; align= NW

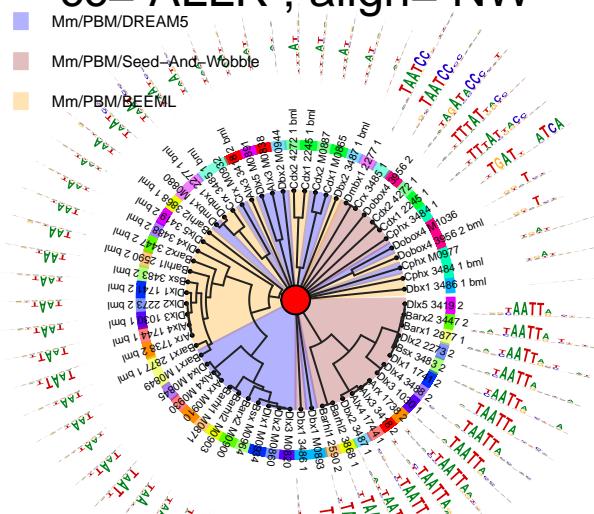


cc= PCC ; align= SWU



1:17; 2:2; 3:2

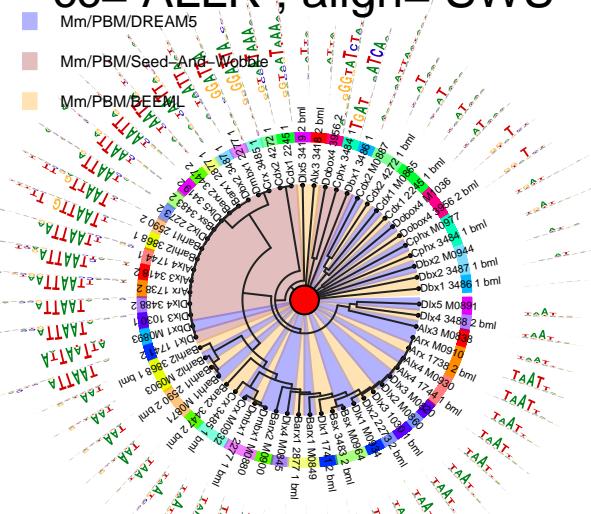
cc= ALLR ; align= NW



1:15; 2.6; 3:0

1:10; 2:9; 3:2

cc= ALLR ; align= SWU



1:8; 2:13; 3:0

Supplementary Figure 2N: Alignment of mouse PBM data by MotIV with different CCM and alignment methods

The number (1:X; 2:Y; 3:Z) on the bottom of the figure 2O and 2P represents the number of motifs from the same TF grouped by different algorithms. 3:Z means that Z number of TFs with 3 of 3 motifs clustered together; 2:Y means that Y number of TFs with 2 of 3 motifs clustered together; 1:X means that X number of TFs failed to cluster any of the 3 motifs together. Thus, the lower the X, the higher the Y and Z, the better the method performed. In this example, MatAlign performed better than MotIV even using the same CCM (ALLR) and alignment method (SWU).

Visualize motif alignments generated by MatAlign with ALLR as the MCC and SWU as the alignment method.

```
##read newick tree. Alignment is done by MatAlign
##The newick tree is generated by Neighbor, which is a part of phylip
outpath <- "output"
matalign_path <- "./app/matalign-v4a"
neighbor_path <- "./app/neighbor.app/Contents/MacOS/neighbor"
MatAlign2tree_path <- "./MatAlign2tree.pl"
system(paste("perl", MatAlign2tree_path, "--in . --pcmpath", pcopath,
            "--out", outpath,
            "--matalign", matalign_path,
            "--neighbor", neighbor_path,
            "--tree", "UPGMA"))
newickstrUPGMA <- readLines(con=file.path(outpath, "NJ.matalign.distMX.nwk"))
##convert it to phylog object
phylogUPGMAmatAlign <- newick2phylog(newickstrUPGMA, FALSE)

##get the leaves of phylog for reordering the pfms
leaveNames <- names(phylogUPGMAmatAlign$leaves)
this_motifs <- pfms[leaveNames]

## data source
dataSource <- factor(grep("_M", leaveNames))
levels(dataSource) <- c("yellow", "blue") ##c("Uniprobe", "CIS-BP")
dataSource <- as.character(dataSource)
## algorithm
algorithm <- factor(!grep("_M", leaveNames)) + grep("_bml", leaveNames)
levels(algorithm) <- c("blue", "brown", "orange") ##("DREAM5", "Seed-And-Wobble", "BEML")
algorithm <- as.character(algorithm)
## motifs from the same PBM data source
motifGroup <- factor(gsub("(.*?)_.*$", "\\\1", leaveNames))
levels.motifGroup <- levels(motifGroup)
levels(motifGroup) <- pairColor[1:length(levels(motifGroup))]
colors.motifGroup <- levels(motifGroup)
motifGroup <- as.character(motifGroup)

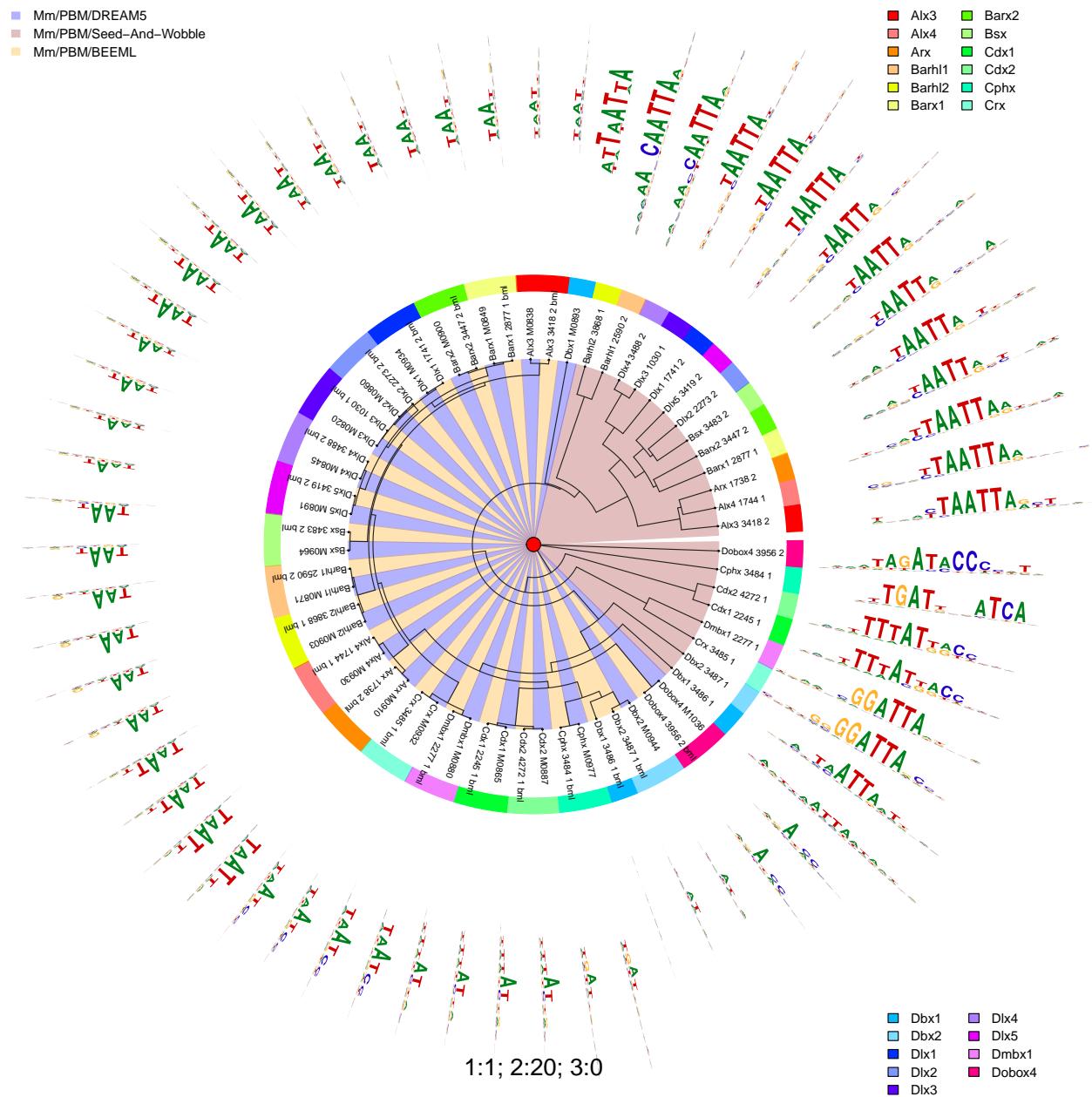
## draw the motifs
plotMotifStackWithRadialPhylog(phylog=phylogUPGMAmatAlign,
                                 pfms=DNAmotifAlignment(this_motifs),
                                 labels.leaves=leaveNames,
                                 col.bg=algorithm,
                                 col.bg.alpha=.3,
                                 col.inner.label.circle=motifGroup,
                                 inner.label.circle.width=.1,
                                 cleaves=.2, circle=1.1,
```

```

    circle.motif=1.6,
    clabel.leaves=.6, angle=358)

legend(1.5, 2.4, legend=levels.motifGroup[1:12],
       fill= colors.motifGroup[1:12],
       border="black", lty=NULL, bty = "n", ncol=2, cex=.8)
legend(1.5, -2, legend=levels.motifGroup[13:21],
       fill= colors.motifGroup[13:21],
       border="black", lty=NULL, bty = "n", ncol=2, cex=.8)
legend(-2.35, 2.4,
       legend=c("Mm/PBM/DREAM5", "Mm/PBM/Seed-And-Wobble", "Mm/PBM/BEEML"),
       fill= highlightCol(c("blue", "brown", "orange"), alpha=.3),
       border="white", lty=NULL, bty = "n", cex=.8)
cnt <- rle(motifGroup)
cnt <- split(algorithm, rep(1:length(cnt$lengths), cnt$lengths))
cnt <- table(sapply(cnt, function(.ele) length(unique(.ele))))
cnt.1 <- vector("integer", 3)
names(cnt.1) <- 1:3
cnt.1[names(cnt)] <- cnt
cnt.1["1"] <- (cnt.1["1"]+cnt.1["2"]*2+cnt.1["3"]*3)/3 - cnt.1["2"] - cnt.1["3"]
text(0, -2.3, label=paste(names(cnt.1), cnt.1, sep=":", collapse="; "), cex=1.5)

```

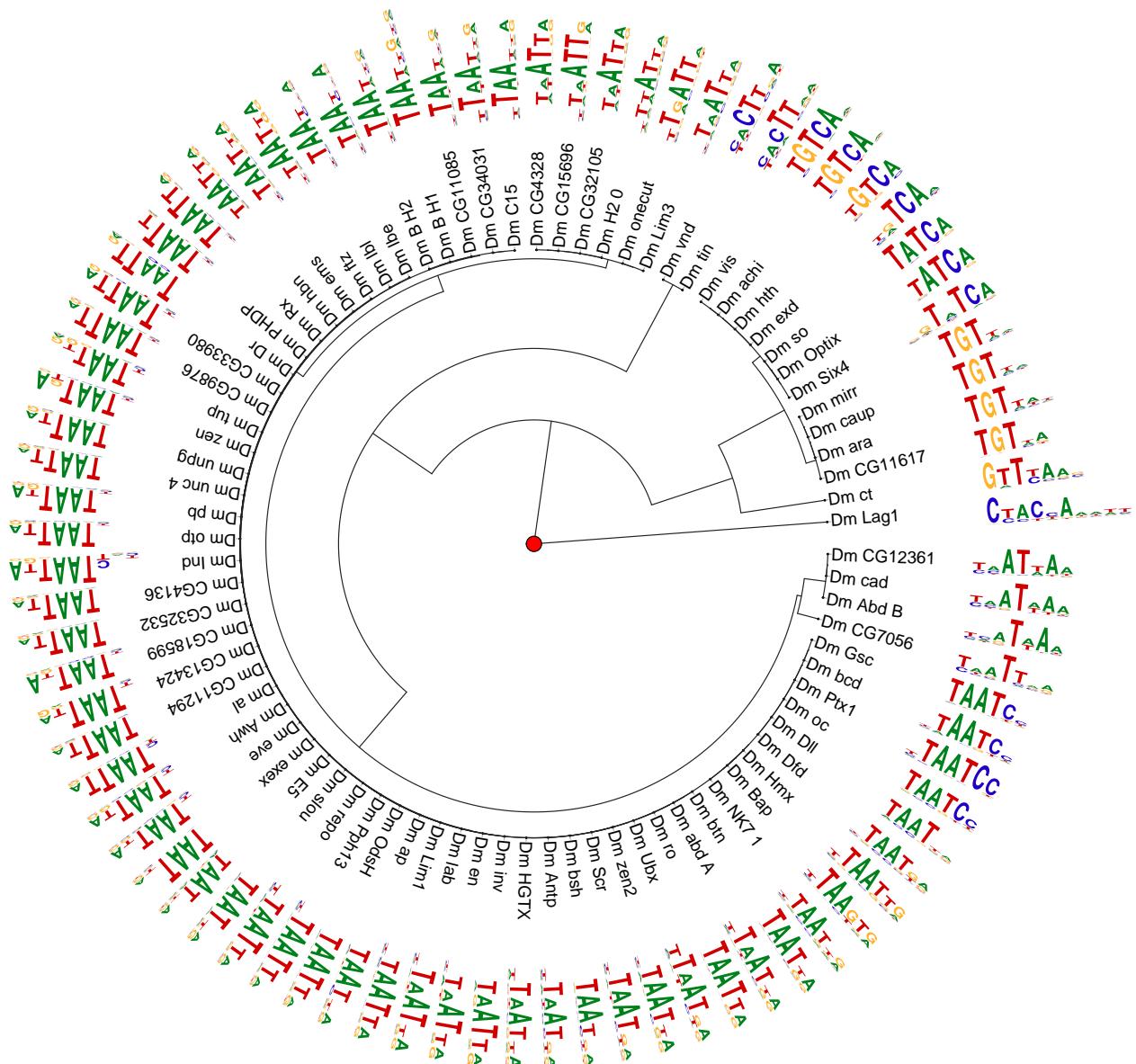


Supplementary Figure 2O: Alignment of mouse PBM data by MatAlign

Supplementary Figure 2P. Improved alignment of fly HD family motifs using MatAlign compared to MotIV. The MatAlign alignment method is superior for discriminating between the closely related motifs in the fly HD family.

```
pcmpath <- "pcmsDatasetDM"
pcms <- readPCM(pcmpath)
pfms<-lapply(pcms,pcm2pfm)
motIVout <- getMotIVOut(pfms, "ALLR", "SWU")
plotMotifStackWithRadialPhylog(phylog=motIVout$phylog,
                                pfms=motIVout$aligned.pfms,
                                labels.leaves=motIVout$leaveNames,
                                cleaves=.2, circle=1.2, circle.motif=1.6,
                                clabel.leaves=1,
                                motifScale="logarithmic",
                                angle=358,
                                plotIndex=FALSE)
text(0, 2.4, label="motIV: cc=ALLR; align=SWU", cex=1.5)
```

motIV: cc=ALLR; align=SWU



Supplementary Figure 2P.a: MotIV(ALLR and SWU) as the alignment method

```

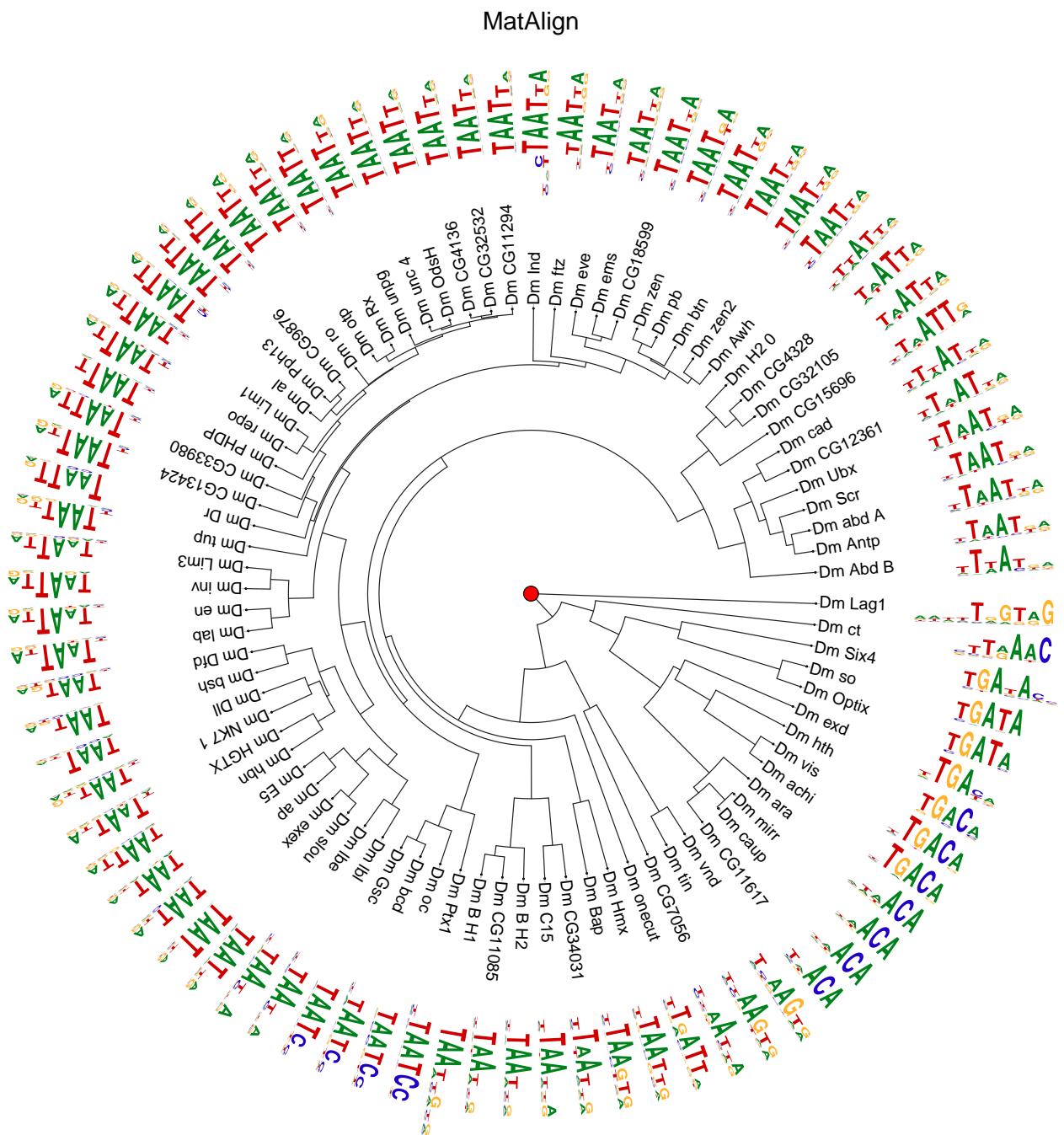
## function to read example data
getMatAlignOut <- function(pcopath, groupDistance=NA, trim=0.2){
  pcms <- readPCM(pcopath)
  pfms<-lapply(pcms,pcm2pfm)
  source("zzz.R")
  system(paste("perl", MatAlign2tree_path, "--in . --pcopath", pcopath,
              "--out", outpath,
              "--matalign", matalign_path,
              "--neighbor", neighbor_path,
              "--tree","UPGMA"))

  newickstrUPGMA <-
    readLines(con=file.path(outpath, "NJ.matalign.distMX.nwk"))
  phylog <- newick2phylog(newickstrUPGMA, FALSE)
  leaves <- names(phylog$leaves)
  motifs <- pfms[leaves]
  if(!is.na(groupDistance)){
    motifSig <-
      motifSignature(motifs, phylog,
                     groupDistance=groupDistance,
                     min.freq=1, trim=trim)
    sig <- signatures(motifSig)
    gpCol <- sigColor(motifSig)
  }else{
    motifSig <- NA
    sig <- NA
    gpCol <- NA
  }

  return(list(phylog=phylog, sig=sig, gpCol=gpCol,
             motifs=DNAmotifAlignment(motifs),
             leaves=leaves,
             unaligned.pfms=motifs))
}

matAlignOut <- getMatAlignOut(pcopath)
plotMotifStackWithRadialPhylog(phylog=matAlignOut$phylog,
                               pfms=matAlignOut$motifs,
                               labels.leaves=matAlignOut$leaves,
                               cleaves=.2, circle=1.2, circle.motif=1.6,
                               clabel.leaves=1, motifScale="logarithmic",
                               angle=358, plotIndex=FALSE)
text(0, 2.4, label="MatAlign", cex=1.5)

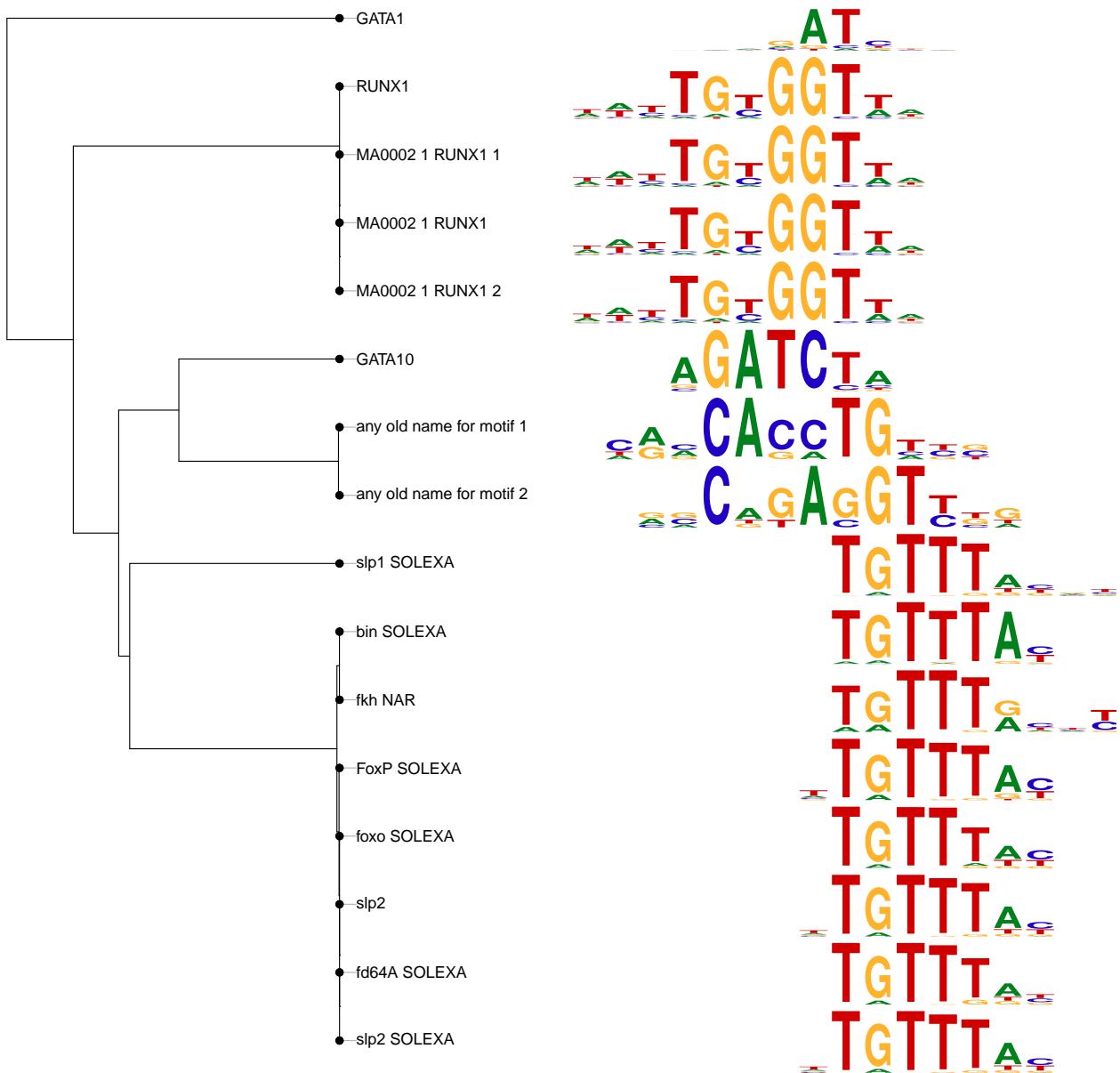
```



Supplementary Figure 2P.b: MatAlign (ALLR and SWU) as the alignment method

Supplementary Figure 2Q. Import PFMs/PCMs files in Transfac, CisBP, or JASPAR format in batch mode.

```
path <- system.file("extdata", package = "motifStack", mustWork = TRUE)
pcms <- importMatrix(dir(path, "*.pcm", full.names = TRUE), format = "pcm", to = "pcm")
JASPAR <- importMatrix(dir(path, "*.jaspar", full.names = TRUE))
pfms <- importMatrix(dir(path, "*.pfm", full.names = TRUE))
transfac <- importMatrix(file.path(path, c("transfac.like.test.transfac", "RUNX1.transfac")))
cisbp <- importMatrix(file.path(path, "PWM.cisbp"))
motifs <- unlist(c(pcms, JASPAR, pfms, transfac, cisbp))
motifs[sapply(motifs, class)=="pcm"] <-
  lapply(motifs[sapply(motifs, class)=="pcm"], pcm2pfm)
motifStack(motifs, layout = "phylog")
```



Supplementary Figure 2Q: Aligned motifs imported from PFMs/PCMs files in Transfac, CisBP, or JASPAR format in batch mode.

Supplementary Figure 2R. Plot affinity logo from a position specific affinity matrix (PSAM).

```
psam <- importMatrix(file.path(path, "PSAM.mxr"), format = "psam")[[1]]  
motifStack(psam)
```



Supplementary Figure 2R: PSAM plotted as an affinity logo.

Session information including the version of R , motifStack and other packages.

```
sessionInfo()

## R version 3.4.2 (2017-09-28)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Sierra 10.12.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib
##
## 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] stats4     parallel   grid       stats      graphics   grDevices utils
## [8] datasets   methods    base
##
## other attached packages:
## [1] motifStack_1.21.6   Biostrings_2.44.2   XVector_0.16.0
## [4] IRanges_2.10.5     S4Vectors_0.14.7   ade4_1.7-8
## [7] MotIV_1.32.0       BiocGenerics_0.22.1 grImport_0.9-0
## [10] XML_3.98-1.9
##
## loaded via a namespace (and not attached):
## [1] Rcpp_0.12.13          highr_0.6
## [3] plyr_1.8.4            compiler_3.4.2
## [5] GenomeInfoDb_1.12.3   bitops_1.0-6
## [7] tools_3.4.2           zlibbioc_1.22.0
## [9] digest_0.6.12         evaluate_0.10.1
## [11] lattice_0.20-35       BSgenome_1.44.2
## [13] Matrix_1.2-11         DelayedArray_0.2.7
## [15] yaml_2.1.14           seqLogo_1.42.0
## [17] GenomeInfoDbData_0.99.0 rtracklayer_1.36.6
## [19] stringr_1.2.0          knitr_1.17
## [21] htmlwidgets_0.9        rprojroot_1.2
## [23] Biobase_2.36.2         BiocParallel_1.10.1
## [25] rGADEM_2.24.0          rmarkdown_1.6
## [27] magrittr_1.5            scales_0.5.0
## [29] backports_1.1.1         Rsamtools_1.28.0
## [31] htmltools_0.3.6          matrixStats_0.52.2
## [33] GenomicRanges_1.28.6    GenomicAlignments_1.12.2
## [35] SummarizedExperiment_1.6.5 colorspace_1.3-2
## [37] stringi_1.1.5           munsell_0.4.3
## [39] RCurl_1.95-4.8
```