

dagLogo guide

Jianhong Ou, Lihua Julie Zhu

November 26, 2014

Contents

1	Introduction	1
2	Prepare environment	2
3	Examples of using dagLogo	2
3.1	Step 1, fetch sequences	2
3.2	Step 2, build background model	3
3.3	Step 3, do test	4
3.4	Step 4, graphical representation results	4
4	using dagLogo to analysis Catobolite Activator Protein	9
5	References	11
6	Session Info	11

1 Introduction

A sequence logo has been widely used as a graphical representation of an alignment of multiple amino acid or nucleic acid sequences. There is a package seqlogo[1] implemented in R to draw DNA sequence logos. And another package motifStack[2] was developed for drawing sequence logos for Amino Acid, DNA and RNA sequences. motifStack also has the capability for graphical representation of multiple motifs.

IceLogo[3] is a tool developed in java to visualize significant conserved sequence patterns in an alignment of multiple peptide sequence against background sequences. Compare to webLogo[4], which relying on information theory, iceLogo builds on probability theory. It is reported that iceLogo has a more dynamic nature and is correcter and completer in the analysis of conserved sequence patterns.

However iceLogo can only compare conserved sequences to reference sequences peptide by peptide. As we know, some conserved sequence patterns are not conserved by peptides but by groups such as

charge, chemistry, hydrophobicity and etc.

Here we developed a R package: dagLogo based on iceLogo to visualize significant conserved sequence patterns in groups.

2 Prepare environment

You will need ghostscript: the full path to the executable can be set by the environment variable `R_GSCMD`. If this is unset, a GhostScript executable will be searched by name on your path. For example, on a Unix, linux or Mac "gs" is used for searching, and on Windows the setting of the environment variable `GSC` is used, otherwise commands "gswi64c.exe" then "gswin32c.exe" are tried.

Example on Windows: assume that the gswin32c.exe is installed at `C:\Program Files\gs\gs9.06\bin`, then open R and try:

```
> Sys.setenv(R_GSCMD="\"C:\Program Files\gs\gs9.06\bin\gswin32c.exe\"")
```

3 Examples of using dagLogo

3.1 Step 1, fetch sequences

You should have interesting peptides position info and the identifiers for fetching sequences via biomaRt.

```
> library(dagLogo)
> library(biomaRt)
> mart <- useMart("ensembl", "dmelanogaster_gene_ensembl")
> dat <- read.csv(system.file("extdata", "dagLogoTestData.csv", package="dagLogo"))
> dat <- dat[1:5,] ##subset to speed sample
> dat
```

	entrez_geneid	NCBI_acc	NCBI_site	molecular_weight	peptide
1	44149	NP_524708	K328	85400.81Da	AGIASEAQk*YQA
2	44149	NP_524708	K43	85400.81Da	ALSk*FSDSDVYLPYEK
3	44149	NP_524708	K123	85400.81Da	AMQDATAQMALLQFISSGLk*K
4	44149	NP_524708	K409	85400.81Da	CASIAk*DAMSHGLK
5	44149	NP_524708	K446	85400.81Da	DGISEVFDk*FGGTVLANACGPCIGQWDR

```
peptide..7..7.
1 GIASEAQkYQAKILS
2 ASKVALSkFSDSDVYL
3 QFISSGLkKVAVPST
4 GRCASIAkDAMSHGL
5 GISEVFDkFGGTVLA
```

```
> try({
+   seq <- fetchSequence(as.character(dat$entrez_geneid),
+     anchorPos=as.character(dat$NCBI_site),
+     mart=mart,
+     upstreamOffset=7,
+     downstreamOffset=7)
+   head(seq@peptides)
+ })
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]	[,13]	[,14]	[,15]
[1,]	"G"	"I"	"A"	"S"	"E"	"A"	"Q"	"K"	"Y"	"Q"	"A"	"K"	"I"	"L"	"S"
[2,]	"A"	"S"	"K"	"V"	"A"	"L"	"S"	"K"	"F"	"D"	"S"	"D"	"V"	"Y"	"L"
[3,]	"Q"	"F"	"I"	"S"	"S"	"G"	"L"	"K"	"K"	"V"	"A"	"V"	"P"	"S"	"T"
[4,]	"G"	"R"	"C"	"A"	"S"	"I"	"A"	"K"	"D"	"A"	"M"	"S"	"H"	"G"	"L"
[5,]	"G"	"I"	"S"	"E"	"V"	"F"	"D"	"K"	"F"	"G"	"G"	"T"	"V"	"L"	"A"

Sometimes you may already have the peptides sequences in hand. You will use `formatSequence` function to prepare an object of `dagPeptides` for further testing. To use `formatSequence`, you need prepare the proteome by `prepareProteome` function.

```
> dat <- unlist(read.delim(system.file("extdata",
+   "grB.txt", package="dagLogo"),
+   header=F, as.is=TRUE))
> head(dat)
```

	V11	V12
"GHISVKEPTPSIASDISLPATQELRQRLR"	"EREMFDKASLKLGLDKAVLQMSGRENATN"	
V13	V14	
"XXXXXXMSDIVVVTDLIAVGLKRGSDLLS"	"GQDQEEEEIEIDILMDTEEEELMRAEDTEQLK"	
V15	V16	
"ESYATDNEKMTSTPETLLEEIEAKNRELIA"	"VENKERTLKRLLLQDQENSLQDNRTSSDSP"	

```
> ##prepare proteome from a fasta file
> proteome <- prepareProteome(fasta=system.file("extdata",
+   "HUMAN.fasta",
+   package="dagLogo"))
> ##prepare object of dagPeptides
> seq <- formatSequence(seq=dat, proteome=proteome,
+   upstreamOffset=14, downstreamOffset=15)
```

3.2 Step 2, build background model

Once you have an object of `dagPeptides` in hand, you can start to build background model for DAG test. The background could be random subsequence of whole proteome or your inputs. If the background was built from whole proteome or proteome without your inputs, an object of `Proteome` is required.

To prepare a proteome, there are two methods, from a fasta file or from UniProt webservice. Last

example shows how to prepare proteome from a fasta file. Here we show how to prepare proteome via UniProt webservice.

```
> if(interactive()){  
+   library(UniProt.ws)  
+   taxId(UniProt.ws) <- 9606  
+   proteome <- prepareProteome(UniProt.ws=UniProt.ws)  
+ }
```

Then the proteome can be used for background model building.

```
> bg <- buildBackgroundModel(seq, bg="wholeGenome", proteome=proteome)
```

3.3 Step 3, do test

Test can be done without any change of the symbol pattern or with changes of grouped peptides by such as charge, chemistry, hydrophobicity and etc.

```
> t0 <- testDAU(seq, bg)  
> t1 <- testDAU(seq, bg, group="classic")  
> t2 <- testDAU(seq, bg, group="charge")  
> t3 <- testDAU(seq, bg, group="chemistry")  
> t4 <- testDAU(seq, bg, group="hydrophobicity")
```

3.4 Step 4, graphical representation results

We can use heatmap (Figure 1) or logo (Figure 2,3,4,5) to show the results.

```
> dagHeatmap(t0)  
> dagLogo(t0)  
> dagLogo(t1, namehash=nameHash(t1@group), legend=TRUE)  
> dagLogo(t2, namehash=nameHash(t2@group), legend=TRUE)  
> dagLogo(t3, namehash=nameHash(t3@group), legend=TRUE)  
> dagLogo(t4, namehash=nameHash(t4@group), legend=TRUE)
```

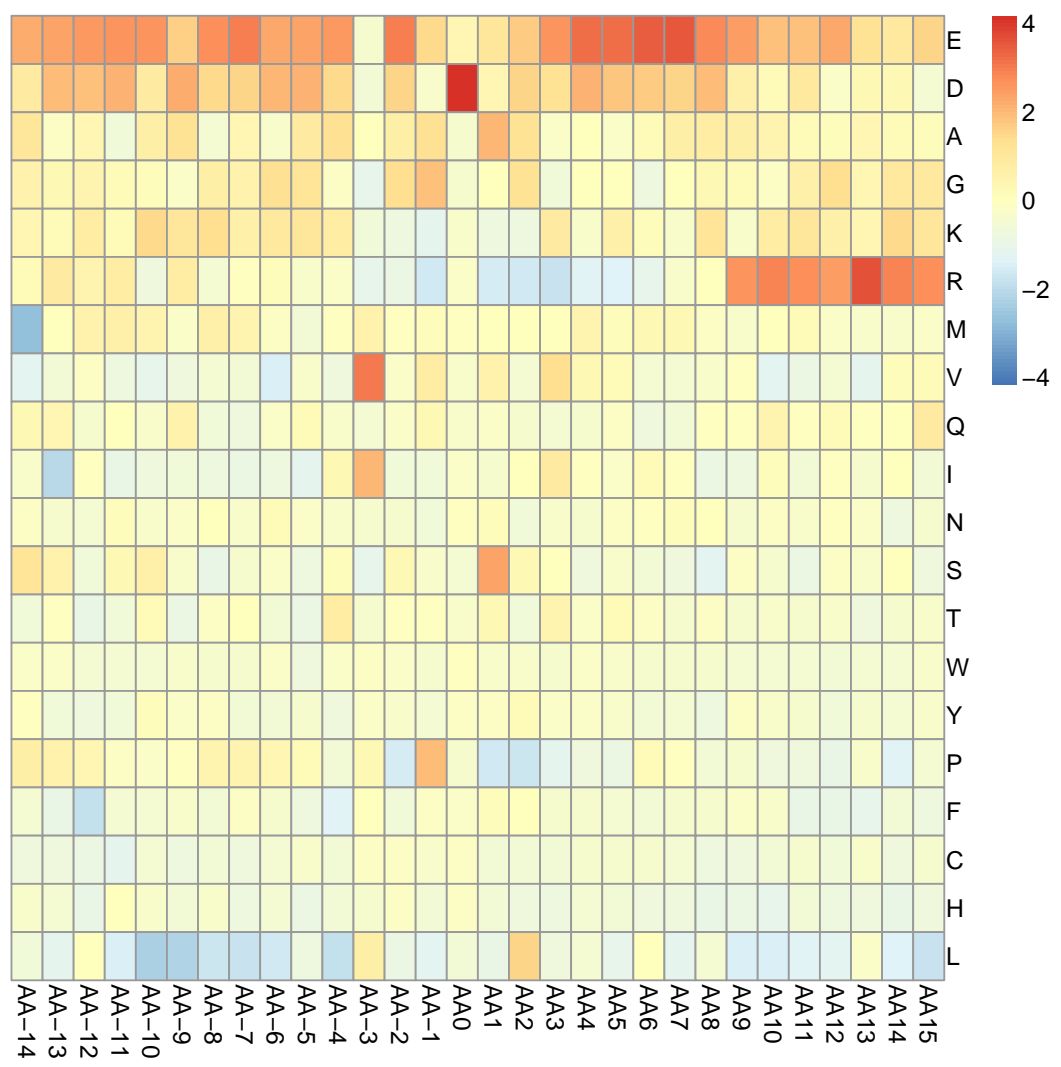


Figure 1: **heatmap** Plot a heatmap to show the results

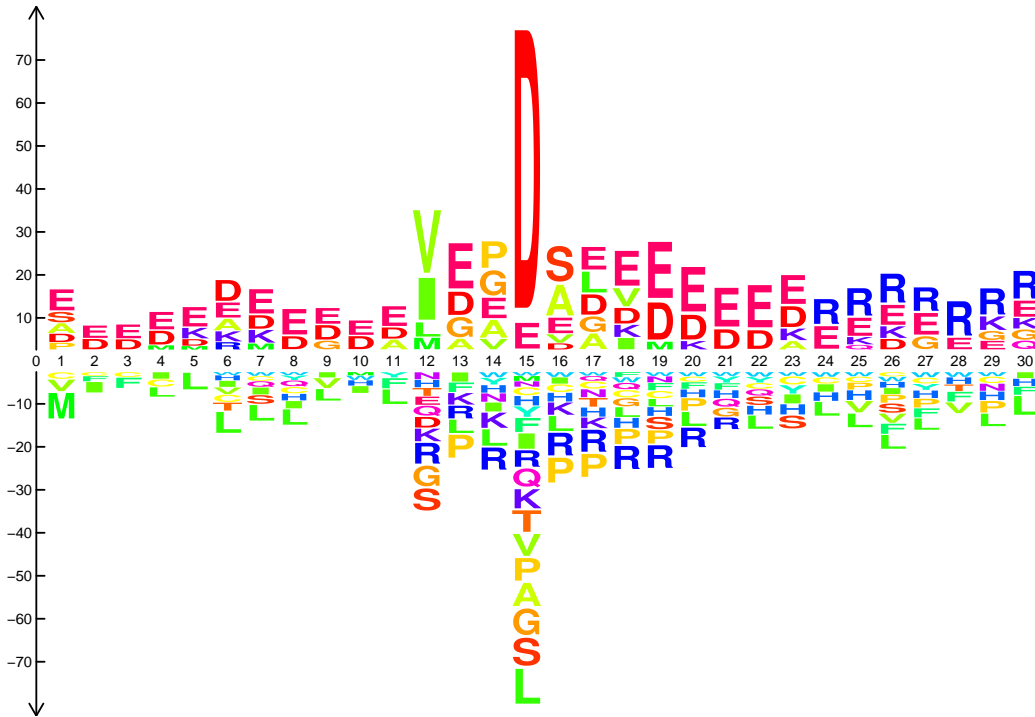


Figure 2: **dagLogo1** Plot a logo to show the ungrouped results

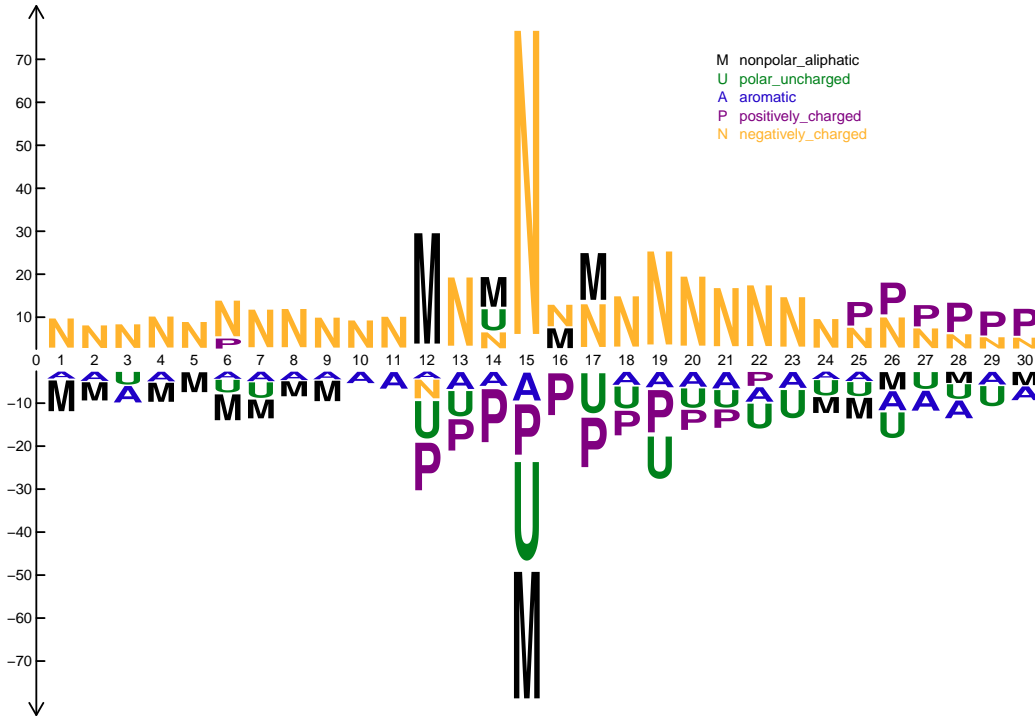


Figure 3: **dagLogo2** Plot a logo to show the classic grouped results

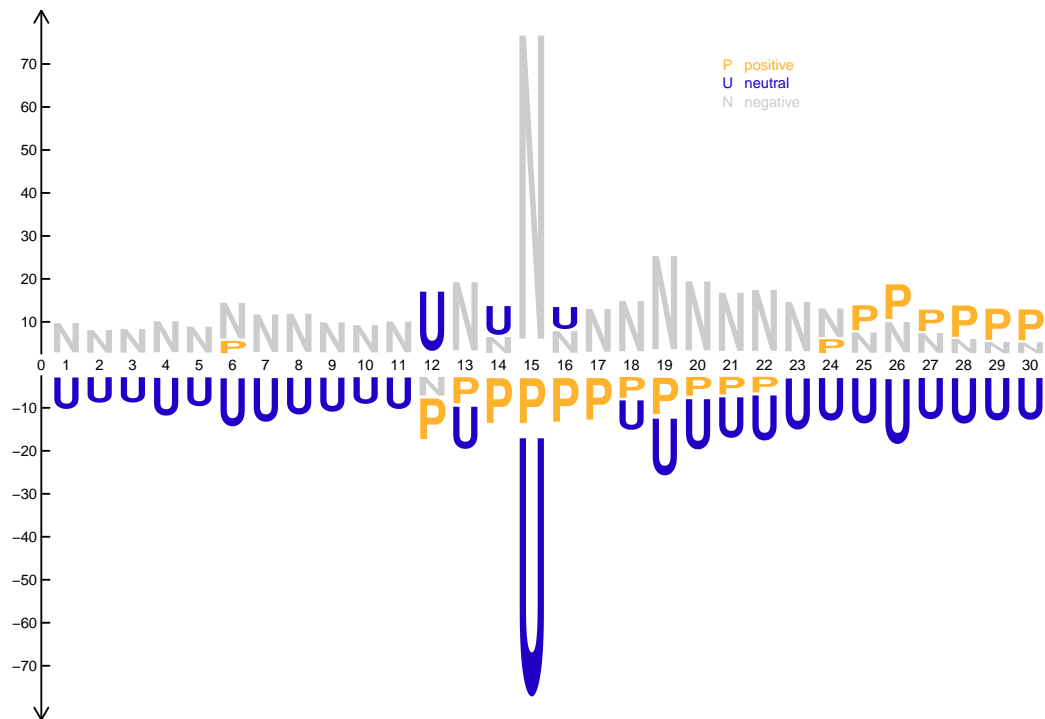


Figure 4: **dagLogo3** Plot a logo to show the charge grouped results

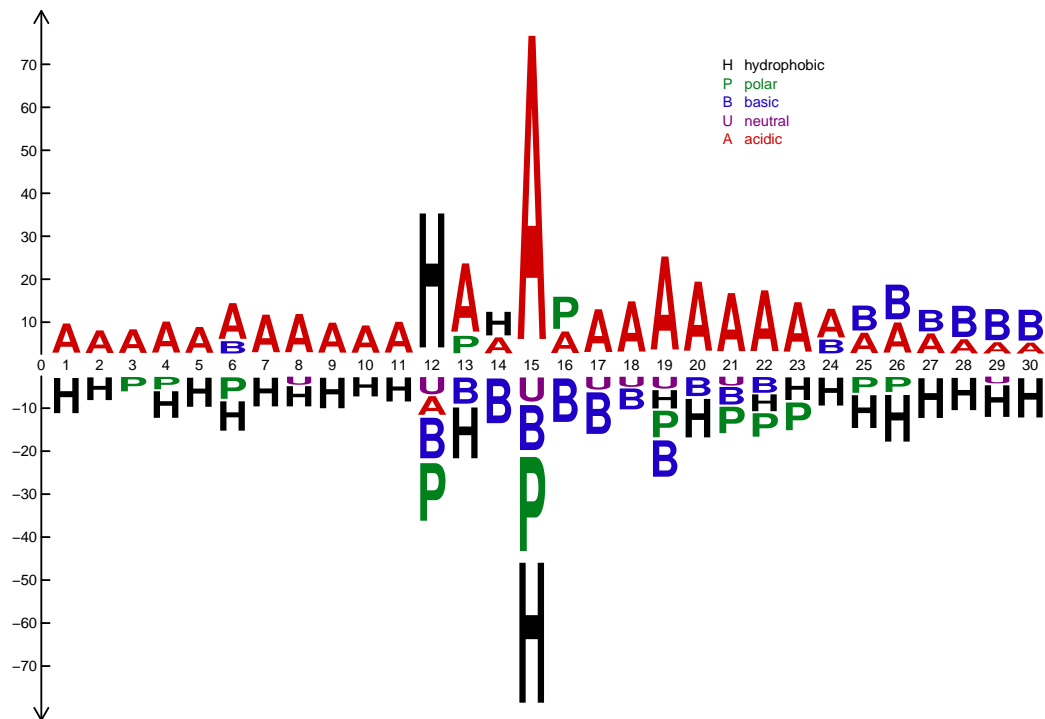


Figure 5: **dagLogo4** Plot a logo to show the chemistry grouped results

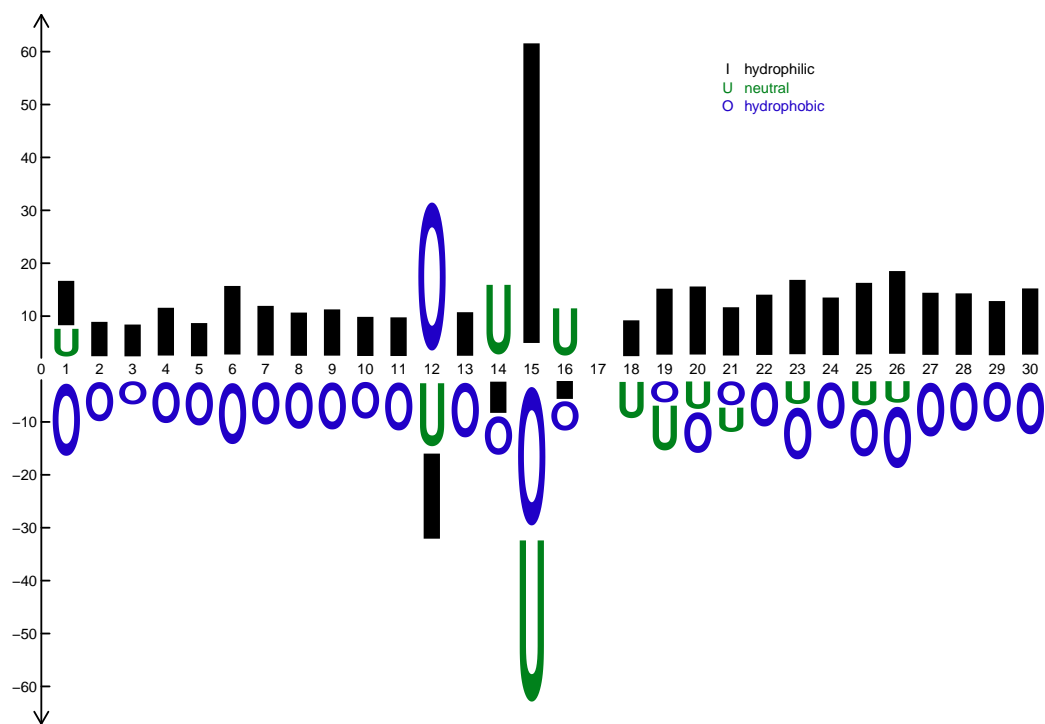


Figure 6: **dagLogo5** Plot a logo to show the hydrophobicity grouped results

4 using dagLogo to analysis Catobolite Activator Protein

CAP (Catabolite Activator Protein, also known as CRP for cAMP Receptor Protein) is a transcription promoter that binds at more than 100 sites within the E. coli genome.

The motif of the DNA-binding helix-turn-helix motif of the CAP family is drawn by motifStack as Figure 7.

```
> library(motifStack)
> protein<-read.table(file.path(find.package("motifStack"),"extdata","cap.txt"))
> protein<-t(protein[,1:20])
> motif<-pcm2pfm(protein)
> motif<-new("pfm", mat=motif, name="CAP",
+           color=colorset(alphabet="AA",colorScheme="chemistry"))
> plot(motif)
```

If we use dagLogo to plot the motif, it will be shown as Figure 8. Residues 7-13 form the first helix, 14-17 the turn and 18-26 the DNA recognition helix. The glycine at position 15 appears to be critical in forming the turn.

```
> library(Biostrings)
> cap <- as.character(readAAStringSet(system.file("extdata",
+                                               "cap.fasta",
```

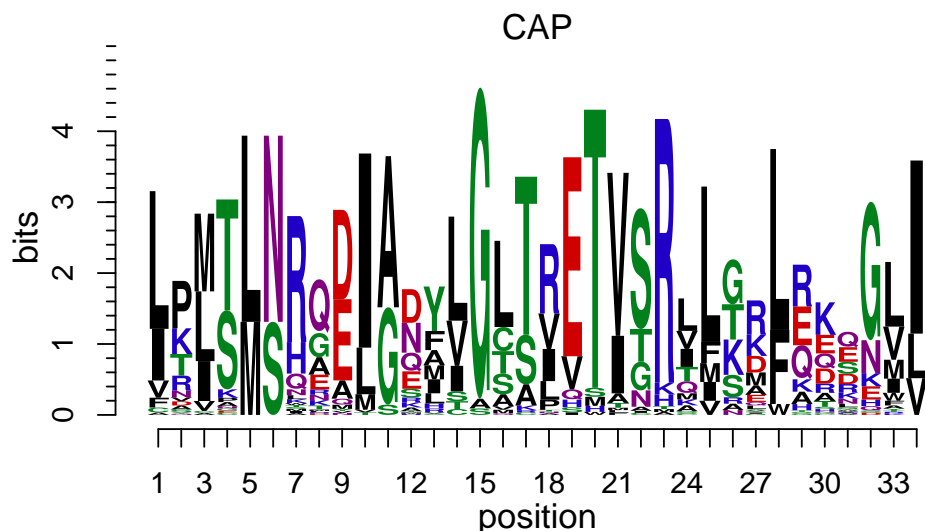


Figure 7: **Catobolite Activator Protein Motif** The DNA-binding helix-turn-helix motif of the CAP family plotted by motifStack



Figure 8: **Catololite Activator Protein Motif** The DNA-binding helix-turn-helix motif of the CAP family plotted by dagLogo

```
+
+                                     package="dagLogo"))))
> data(ecoli.proteome)
> seq <- formatSequence(seq=cap, proteome=ecoli.proteome)
> bg <- buildBackgroundModel(seq, bg="wholeGenome",
+                             proteome=ecoli.proteome,
+                             permutationSize=10L)
> t0 <- testDAU(seq, bg)
> dagLogo(t0)
```

If the peptides are grouped by chemistry and then plot, it will be shown as Figure 9. Positions 10, 14, 16, 21 and 25 are partially or completely buried and therefore tend to be populated by hydrophobic amino acids, which are very clear if we group the peptides by chemistry.

```
> t1 <- testDAU(seq, bg, group="chemistry")
> dagLogo(t1, namehash=nameHash(t1@group), legend=TRUE)
```

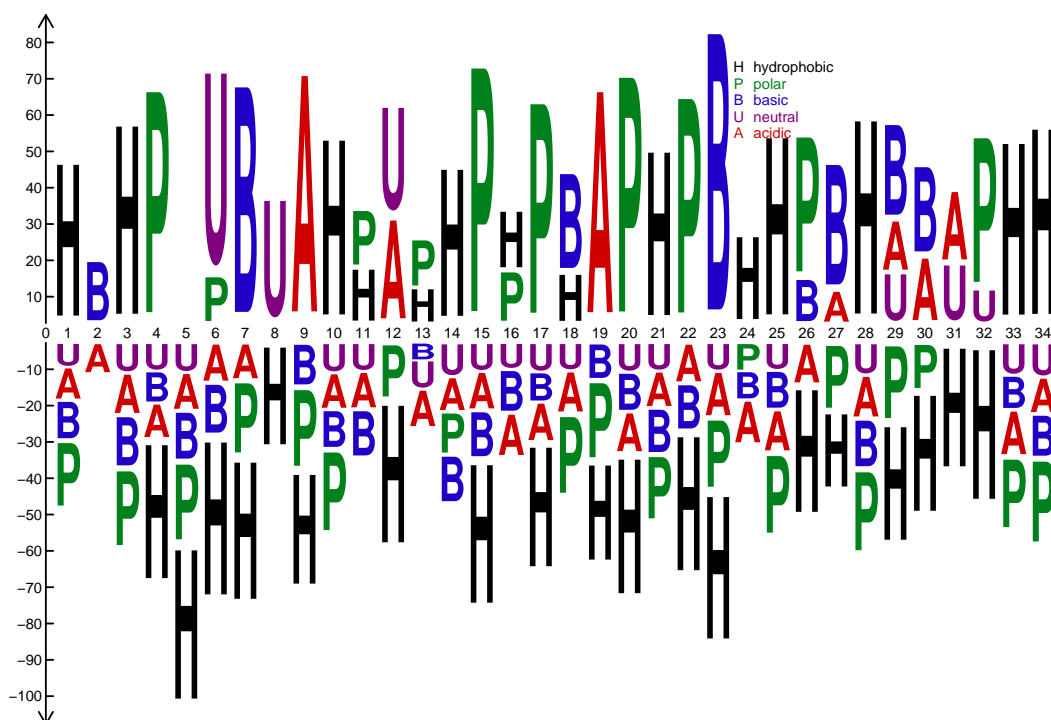


Figure 9: **Catobolite Activator Protein Motif** The DNA-binding helix-turn-helix motif of the CAP family grouped by chemistry

5 References

References

- [1] seqLogo: Sequence logos for DNA sequence alignments. R package version 1.5.4.
- [2] motifStack: Plot stacked logos for single or multiple DNA, RNA and amino acid sequence. R package version 1.5.4.
- [3] Colaert and Helsens et al. Improved visualization of protein consensus sequences by iceLogo. Nature methods (2009) vol. 6 (11) pp. 786-7 (pid: 19876014)
- [4] Crooks GE and Brenner SE et al. WebLogo: A sequence logo generator. Genome Research (2004) 14:1188-1190

6 Session Info

```
> toLatex(sessionInfo())
```

- R version 3.1.2 (2014-10-31), i386-w64-mingw32

- Locale: LC_COLLATE=C, LC_CTYPE=English_United States.1252, LC_MONETARY=English_United States.1252, LC_NUMERIC=C, LC_TIME=English_United States.1252
- Base packages: base, datasets, grDevices, graphics, grid, methods, parallel, stats, stats4, utils
- Other packages: BiocGenerics 0.12.1, Biostrings 2.34.0, IRanges 2.0.0, MotIV 1.22.0, S4Vectors 0.4.0, XML 3.98-1.1, XVector 0.6.0, ade4 1.6-2, biomaRt 2.22.0, dagLogo 1.4.1, grImport 0.9-0, motifStack 1.10.1
- Loaded via a namespace (and not attached): AnnotationDbi 1.28.1, BBmisc 1.8, BSgenome 1.34.0, BatchJobs 1.5, Biobase 2.26.0, BiocParallel 1.0.0, BiocStyle 1.4.1, DBI 0.3.1, GenomeInfoDb 1.2.3, GenomicAlignments 1.2.1, GenomicRanges 1.18.3, RColorBrewer 1.0-5, RCurl 1.95-4.3, RSQLite 1.0.0, Rsamtools 1.18.2, base64enc 0.1-2, bitops 1.0-6, brew 1.0-6, checkmate 1.5.0, codetools 0.2-9, digest 0.6.4, fail 1.2, foreach 1.4.2, iterators 1.0.7, lattice 0.20-29, pheatmap 0.7.7, rGADEM 2.14.0, rtracklayer 1.26.2, sendmailR 1.2-1, seqLogo 1.32.1, stringr 0.6.2, tools 3.1.2, zlibbioc 1.12.0