Genomic-Benchmarks Test

Table of contents

1 Downloading data

Python Code

```python
# Listing available datasets
from genomic_benchmarks.data_check import list_datasets
list_datasets()
# Inspecting each dataset to select two
from genomic_benchmarks.data_check import info as info_gb
info_gb("human_nontata_promoters") # <- This one will be used
info_gb("human_ensembl_regulatory")
info_gb("human_enhancers_cohn")    # <- This one will also be used
info_gb("human_enhancers_ensembl")
# Downloading datasets
from genomic_benchmarks.loc2seq import download_dataset
import os
os.chdir("/home/davidfm/Projects/UBMI-IFC/EnhaProm/datasets/GenomicBenchmarks")
download_dataset("human_nontata_promoters", version=0)
download_dataset("human_enhancers_cohn", version=0)
```

2 Formatting data

The downloaded data came in the form of two directories with many *.txt* files inside, and seeing that at least '*getShape()*' function from '*DNAshapeR*' library required FASTA files to work and it felt right to collect all sequences in a single file, I transfered all sequences of each cis-regulatory element to their single respective FASTA file and gave them a simple header.

Bash Code

```
cd /home/davidfm/Projects/UBMI-IFC/EnhaProm/datasets/GenomicBenchmarks/
awk 'BEGIN{counter=0}{print ">promoter_"counter"|train|positive";
     print $0; counter+=1}' human_nontata_promoters/train/positive/*.txt \
     > promoters_train_positive.fasta
awk 'BEGIN{counter=0}{print ">enhancer_"counter"|train|positive";
     print $0; counter+=1}' human_enhancers_cohn/train/positive/*.txt \
     > enhancers_train_positive.fasta
```

3 Libraries used

Here is a list of the R libraries used for the following analysis, as well as a reference to my own functions to be used in the characterization step.

R Code

```
# For genome-functions.R
library(stringr)
library(stringi)
library(primes)
# For parallel computing
library(doParallel)
library(foreach)
# For biological functions:
#   - Local/Global alignments
#   - DNA Shape computing
library(Biostrings)
library(DNAshapeR)
# For plotting
library(ggplot2)
library(dplyr)
library(plyr)
# For pretty tables
library(knitr)
library(kableExtra)
# For my own functions
source("/home/davidfm/Projects/UBMI-IFC/EnhaProm/scripts/genome-functions.R")
```

4 Characterizing sequences

Here, we characterize a subset from each of our datasets: 1638 sequences per regulatory element; 3276 in total. However, there's a circumstance about the sequences that has to be noted:

- Both datasets have considerably different elements lengthwise:
    1. All promoters have a length of 251 nucleotides.
    2. All enhancers have a length of 500 nucleotides.

```
# Scanning sequences
prom_fasta <- "datasets/GenomicBenchmarks/promoters_train_positive.fasta"
enha_fasta <- "datasets/GenomicBenchmarks/enhancers_train_positive.fasta"
prom_seqs <- scan(prom_fasta, character(), quote = "")[seq(2, 29484, 2)]
enha_seqs <- scan(enha_fasta, character(), quote = "")[seq(2, 20842, 2)]
```

Given some previous tests done to 'sequences_characterizer()' I came to the conclusion that parallel computing might provide a higher and more complex set of data in a feasible time span.

```
# Prepairing clusters for parallel computing
corescluster <- makeCluster(6)
registerDoParallel(corescluster)
# Characterizing sequences and exporting to CSV
list_seqs <- list(promoters = prom_seqs, enhancers = enha_seqs)
reg_elems <- c("promoters", "enhancers")
for (reg_elem in reg_elems) {
  foreach(i = 1:6) %dopar% {
    library(stringr) # for some reason we have to specify for
    library(stringi) # all the libraries that the sequence
    library(primes)  # characterizer needs
    i_start <- ((i - 1) * 273) + 1
    i_final <- i * 273
    if (i > 1) {     # Conditional so that only the first CSV has headers
      write.table(sequences_characterizer(list_seqs[[reg_elem]][i_start:i_final],
                                          k_max = 6, optim = TRUE),
                  paste("datasets/GB-Testing/test", reg_elem, "-minitraining_",
                        i, ".csv", sep = ""), sep = ",",
                  row.names = FALSE, col.names = FALSE)
    } else {
      write.csv(sequences_characterizer(list_seqs[[reg_elem]][i_start:i_final],
                                        k_max = 6, optim = TRUE),
                paste("datasets/GB-Testing/", reg_elem, "-minitraining_",
                      i, ".csv", sep = ""), row.names = FALSE)
    }
  }
}
```

5 Concatenating CSV's

It was decided to produce many files instead of appending over the same CSV table in order to apply a sort of quality control checkup after the parallel computing was done. This, because when forcing the process RAM overload was possible and the function could die mid-process. Since our data was separated in six tables per cis-regulatory element, here we only join each set together in a single CSV.

Bash Code

```
cat datasets/GB-Testing/testpromoters-minitraining_*.csv \
    > datasets/GB-Testing/test-1638-promoters-6mers.csv
cat datasets/GB-Testing/testenhancers-minitraining_*.csv \
    > datasets/GB-Testing/test-1638-enhancers-6mers.csv
```

6 Data description

Table 1: Overview of each column generated by '*sequences-characterizer()*'

Column	Description	Section Processed
A	Percentage of Alanines	per Sequence
T	Percentage of Thymines	
C	Percentage of Cytosines	
G	Percentage of Guanines	
temp	Melting Temperature	
shan	Shannon Coefficient	
kN.M_prod	KSG Product	per Kmer: each possible kmer of size N is identified on a scale of 1 to 4^N denoted by M. `i.e. k3.1 (or the first kmer of size 3) would be AAA; k4.2 would be AAAC`
kN.M_barc	Barcode Profile	
kN.M_pals	Palindrome Profile	
kN.M_revc	Reverse Complement Profile	

Prior to our primary analysis, it feels reasonable to explain the columns per sequence produced by our tabulator '*sequences_characterizer()*'. From here we'll first describe the ones computed sequence-wise, the ones computed kmer-wise, then the ones computed over each kmer distribution, and finally the ones corresponding to DNA-Shape; this feature comes at last considering '*getShape()*' function makes its' own dataframe. The ones in **black** are already integrated in the table, the ones in **red** are yet to be adjoined:

Per sequence

- From *'genome-functions.R'*:
  - **A, T, C, G** - *Nucleotide Percentages* per sequence.
  - **temp** - *Melting Temperature*: Temperature at which DNA's double helix dissociates into single strands. *It's dependent on GC percentage and sequence length.*
  - **shan** - *Shannon Entropy Coefficient*: Statistical quantifier of information in a system. Measures the uncertainty a set of data has. In this case is a nucleotide-diversity metric. *It's dependent on nucleotide percentages.*
- From *'Biostrings'*:
  - **la_sc** - *Local Alignment Score*
  - **la_id** - *Local Alignment Identity*
  - **ga_sc** - *Global Alignment Score*
  - **ga_id** - *Global Alignment Identity*

Per kmer

- From *'genome-functions.R'*:
  - **kN.M_prod** - *KSG Product*: Its obtained by multiplying **Kmer-Percentage * Shannon Entropy * GC-Percentage** (in concept). However, this is a little tricky since you don't want the product to be zero if a sequence is solely composed of A/T nucleotides.
  - **kN.M_barc** - *Barcode Profile*:
  - **kN.M_pals** - *Palindromes' Profile*:
  - **kN.M_revc** - *Reverse Complement Profile*:

Per kmer distribution

Per non-defined kmer

- From *'DNAshapeR'*:
  Its produced a row per sequence and a set vectors (EP, MGW, HelT, Roll & ProT as default) obtained when dividing each sequence in kmers (length non-defined by me) of whose length depends on sequence-length, so I would classify it as per
  - **sh_ep** - Shape EP
  - **sh_mgw** - Shape MGW
  - **sh_helt** - Shape HelT

- **sh_roll** - Shape Roll
- **sh_prot** - Shape ProT

7 Primary analysis

Firstly we read our CSV tables into our conviniently named dataframes.

We could have concatenated both tables together, however I prefer to keep them in different files.

R Code

```
setwd("/home/davidfm/Projects/UBMI-IFC/EnhaProm")
testpromoters <- read.csv("datasets/GB-Testing/test-1638-promoters-6mers.csv",
                          check.names = F)
testenhancers <- read.csv("datasets/GB-Testing/test-1638-enhancers-6mers.csv",
                          check.names = F)
```

First we get an overviwew of the dimensions of our data:

```
deparse(substitute(testpromoters))
deparse(substitute(testenhancers))
dim(testpromoters)
dim(testenhancers)
```

```
[1] "testpromoters"
[1]  1638 21830
```

```
[1] "testenhancers"
[1]  1638 21830
```

It's noticeable the fact that we have way more columns than rows in this test table. Let's get a glimpse of the records corresponding to the first three promoters.

```
kable(testpromoters[1:3,1:18])
```

A	T	C	G	temp	shan
0.1673307	0.2231076	0.3306773	0.2788845	87.21315	1.956136
0.2629482	0.2788845	0.2549801	0.2031873	81.00598	1.990374
0.3625498	0.2031873	0.2470120	0.1872510	80.02590	1.948719

k2.1_prod	k2.1_barc	k2.1_pals	k2.1_revc	k2.2_prod	k2.2_barc
9	1.959765	2.177403e+09	1.374020e+12	17.11198	1.531862
17	2.633165	1.138401e+17	2.971115e+17	26.44579	2.624612
38	5.347025	3.981015e+36	8.100763e+29	24.89016	1.855662

k2.2_pals	k2.2_revc	k2.3_prod	k2.3_barc	k2.3_pals	k2.3_revc
3.845422e+15	3.525451e+11	26.44579	2.445328	4.788062e+13	1.305836e+26
2.607331e+20	6.308689e+14	24.89016	2.513656	1.320117e+14	6.435389e+19
1.156904e+25	3.573059e+12	34.22397	3.431445	6.011592e+17	7.178542e+17

Let's get a glimpse of the records corresponding to the first three enhancers.

```
kable(testenhancers[1:3,1:18])
```

A	T	C	G	temp	shan
0.240	0.236	0.234	0.290	85.0392	1.993988
0.204	0.286	0.210	0.300	84.4652	1.978249
0.250	0.280	0.258	0.212	82.8252	1.992923

k2.1_prod	k2.1_barc	k2.1_pals	k2.1_revc	k2.2_prod	k2.2_barc
36	10.432345	1.885437e+37	8.632413e+30	24.89016	6.027803
20	6.153015	5.673403e+20	1.061673e+53	23.33452	4.984254
31	9.936447	1.640775e+32	5.563175e+39	46.66905	9.301359

k2.2_pals	k2.2_revc	k2.3_prod	k2.3_barc	k2.3_pals	k2.3_revc
1.579134e+29	1.707234e+31	76.22611	13.43518	3.557032e+41	5.830130e+44
4.485619e+32	1.810188e+35	73.11484	13.29662	3.273426e+39	3.755909e+39
9.951038e+37	3.612544e+25	60.66976	12.53356	4.255600e+33	3.010280e+57

```
meanpromoters <- colMeans(testpromoters)
meanenhancers <- colMeans(testenhancers)
sdpromoters <- apply(testpromoters, 2, sd)
sdenhancers <- apply(testenhancers, 2, sd)
# names_test <- rep(names(testpromoters),2)
cre_summary <- data.frame(Type = factor(rep(c("Promoter","Enhancer"),
                                      each = length(testpromoters))),
                          Field = rep(names(testpromoters),2),
                          Means = c(meanpromoters, meanenhancers),
                          StDevs = c(sdpromoters, sdenhancers))
# head(cre_summary)
kable(cre_summary[c(1:10,21831:21840),])
```

	Type	Field	Means	StDevs
1	Promoter	A	1.911256e-01	7.145510e-02
2	Promoter	T	1.995826e-01	7.599340e-02

	Type	Field	Means	StDevs
3	Promoter	C	2.962655e-01	8.067840e-02
4	Promoter	G	3.130263e-01	8.524230e-02
5	Promoter	temp	8.720208e+01	5.379461e+00
6	Promoter	shan	1.891724e+00	9.437000e-02
7	Promoter	k2.1_prod	1.196276e+01	8.995306e+00
8	Promoter	k2.1_barc	1.500971e+00	1.229003e+00
9	Promoter	k2.1_pals	2.634405e+51	1.065029e+53
10	Promoter	k2.1_revc	9.789346e+62	3.960331e+64
21831	Enhancer	A	2.653712e-01	5.853000e-02
21832	Enhancer	T	2.678205e-01	5.815610e-02
21833	Enhancer	C	2.351111e-01	5.648440e-02
21834	Enhancer	G	2.316972e-01	5.435370e-02
21835	Enhancer	temp	8.269434e+01	3.785299e+00
21836	Enhancer	shan	1.959202e+00	3.892580e-02
21837	Enhancer	k2.1_prod	4.093346e+01	1.835697e+01
21838	Enhancer	k2.1_barc	1.208127e+01	5.751107e+00
21839	Enhancer	k2.1_pals	2.567318e+112	1.038419e+114
21840	Enhancer	k2.1_revc	3.321646e+121	1.344343e+123

```
# Get only 'prod' columns of each kmer
k2 <- n_ki(2); k3 <- n_ki(3); k4 <- n_ki(4); k5 <- n_ki(5); k6 <- n_ki(6)
kmer_sections_indexes <- c(1,
                           k2,
                           k2 + 1,
                           k2 + k3,
                           k2 + k3 + 1,
                           k2 + k3 + k4,
                           k2 + k3 + k4 + 1,
                           k2 + k3 + k4 + k5,
                           k2 + k3 + k4 + k5 + 1,
                           k2 + k3 + k4 + k5 + k6)
prod_indexes <- seq(7,21827,4)
barc_indexes <- seq(8,21828,4)
pals_indexes <- seq(9,21829,4)
revc_indexes <- seq(10,21830,4)
all_prod_indexes <- c(prod_indexes, 21830 + prod_indexes)
all_barc_indexes <- c(barc_indexes, 21830 + barc_indexes)
all_pals_indexes <- c(pals_indexes, 21830 + pals_indexes)
all_revc_indexes <- c(revc_indexes, 21830 + revc_indexes)
kable(head(testenhancers[, barc_indexes])[,c(1:8,length(barc_indexes))])
```

k2.1_barc	k2.2_barc	k2.3_barc	k2.4_barc	k2.5_barc	k2.6_barc	k2.7_barc	k2.8_barc	k6.4096_barc
10.432345	6.027803	13.435177	5.182017	9.672032	14.452432	2.0213095	13.731694	0.0000000
6.153015	4.984254	13.296620	6.098819	10.185028	7.732096	1.3635487	10.150355	0.3058662

k2.1_barc	k2.2_barc	k2.3_barc	k2.4_barc	k2.5_barc	k2.6_barc	k2.7_barc	k2.8_barc	k6.4096_barc
9.936447	9.301359	12.533556	6.353834	10.894642	9.025510	1.3565393	16.335286	0.0000000
23.252879	8.744180	13.868229	9.853451	12.204317	3.958776	0.0930320	8.444184	0.0000000
3.814939	7.617881	9.290377	5.365261	10.457174	13.017956	1.4893074	14.727846	2.7938687
14.929810	6.939986	12.525913	11.864119	10.686542	2.227940	0.9421584	7.264738	0.0000000

```
kable(head(testenhancers[496:500, barc_indexes], 5)[kmer_sections_indexes])
```

k2.1_barc	k2.16_barc	k3.1_barc	k3.64_barc	k4.1_barc
21.956001	19.037416	10.818389	10.1549028	4.0524987
13.176071	24.867054	4.338962	12.7780698	0.7618167
17.160666	15.872687	7.995707	5.5079687	3.3905390
9.811821	2.290862	3.488092	0.6337946	1.1768199
9.711428	9.532818	1.272394	2.3765456	0.0000000

k4.256_barc	k5.1_barc	k5.1024_barc	k6.1_barc	k6.4096_barc
4.4361757	1.945930	1.662122	0.8196523	0.6108495
6.6035027	0.084327	2.394564	0.0000000	0.9545003
1.3116586	1.420021	0.000000	0.7415583	0.0000000
0.0000000	0.000000	0.000000	0.0000000	0.0000000
0.3163498	0.000000	0.000000	0.0000000	0.0000000

```
kable(cre_summary[c(prod_indexes[1:5],(21830+prod_indexes)[1:5]),])
```

	Type	Field	Means	StDevs
7	Promoter	k2.1_prod	11.962760	8.995306
11	Promoter	k2.2_prod	15.840314	6.536126
15	Promoter	k2.3_prod	27.649084	9.205828
19	Promoter	k2.4_prod	8.638156	6.974283
23	Promoter	k2.5_prod	22.012519	7.995609
21837	Enhancer	k2.1_prod	40.933455	18.356973
21841	Enhancer	k2.2_prod	39.367630	9.765868
21845	Enhancer	k2.3_prod	58.095082	13.575118
21849	Enhancer	k2.4_prod	31.738828	12.985948
21853	Enhancer	k2.5_prod	57.394191	11.806409

```
subset_cre_nucl <- cre_summary[c(1:4,21830+(1:4)),]
subset_cre_temp <- cre_summary[c(5,21830+5),]
subset_cre_shan <- cre_summary[c(6,21830+6),]
subset_cre_prod <- cre_summary[c(prod_indexes[17:64],(21830+prod_indexes)[17:64]),]
```

```
subset_cre_barc <- cre_summary[c(barc_indexes[17:64],(21830+barc_indexes)[17:64]),]
subset_cre_pals <- cre_summary[c(pals_indexes[17:64],(21830+pals_indexes)[17:64]),]
subset_cre_revc <- cre_summary[c(revc_indexes[17:64],(21830+revc_indexes)[17:64]),]
kable(cbind(subset_cre_prod[subset_cre_prod$Type=="Promoter",],
      subset_cre_prod[subset_cre_prod$Type=="Enhancer",])[,c(2,1,3,4,5,7,8)][1:10,])
```

	Field	Type	Means	StDevs	Type.1	Means.1	StDevs.1
71	k3.1_prod	Promoter	3.733822	4.441567	Enhancer	15.169109	10.176757
75	k3.2_prod	Promoter	3.208713	2.679694	Enhancer	9.520357	4.701251
79	k3.3_prod	Promoter	5.069128	3.448887	Enhancer	13.390625	5.694958
83	k3.4_prod	Promoter	2.363513	2.747660	Enhancer	9.863853	5.715001
87	k3.5_prod	Promoter	3.646951	3.485361	Enhancer	12.881026	5.906393
91	k3.6_prod	Promoter	5.124804	3.324229	Enhancer	11.042922	5.162449
95	k3.7_prod	Promoter	3.650813	3.084850	Enhancer	3.254059	3.143177
99	k3.8_prod	Promoter	3.756481	2.794403	Enhancer	11.358009	4.689413
103	k3.9_prod	Promoter	5.426771	4.453803	Enhancer	14.490419	6.240096
107	k3.10_prod	Promoter	9.318083	4.745578	Enhancer	15.674036	6.345508

```
field_order_nucl <- subset_cre_nucl$Field[1:4]
field_order_prod <- subset_cre_prod$Field[1:48]
field_order_barc <- subset_cre_barc$Field[1:48]
```

```
subset_cre_temp$Means
```

```
[1] 87.20208 82.69434
```

```
subset_cre_temp$StDevs
```

```
[1] 5.379461 3.785299
```

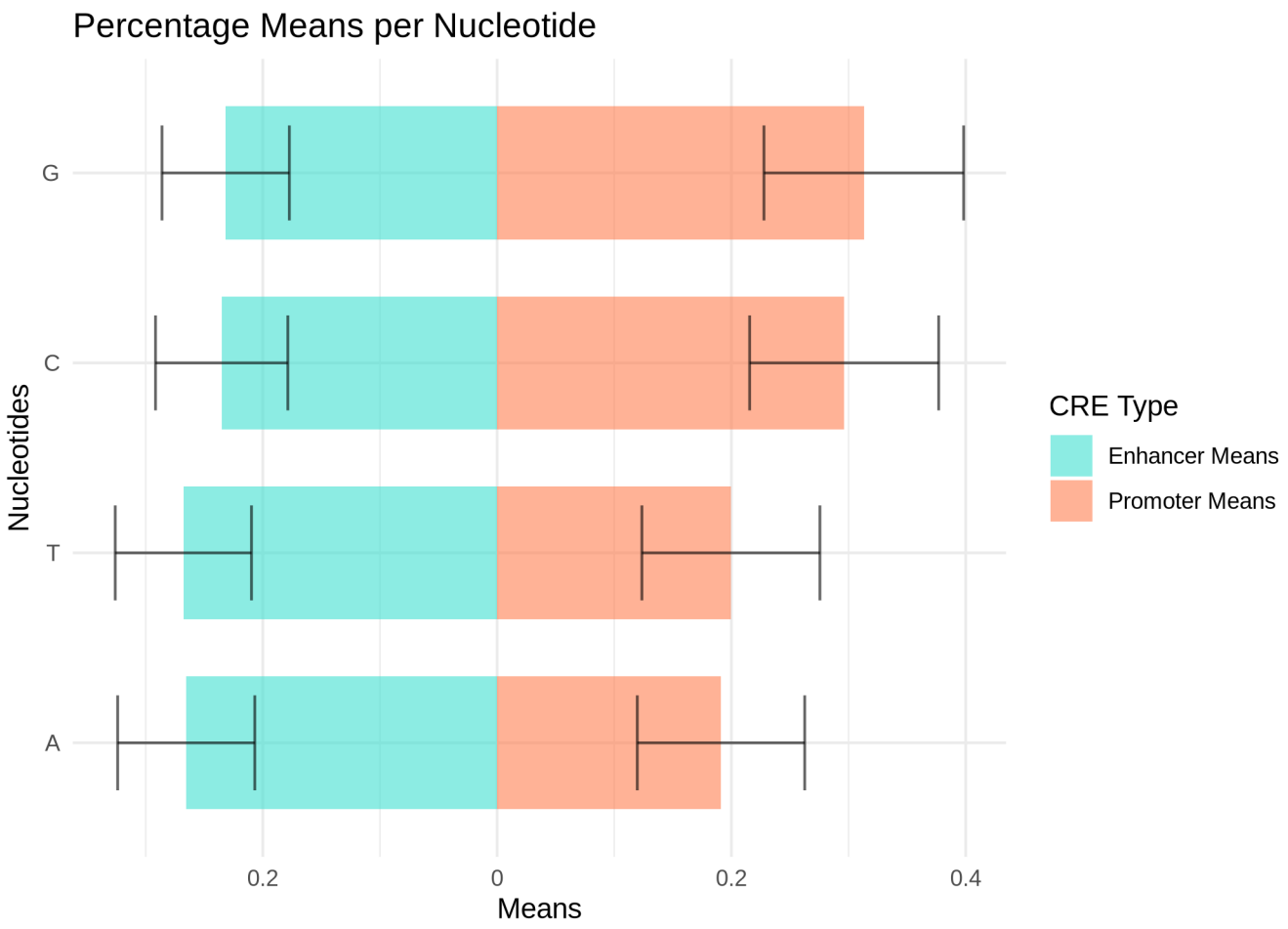
```
subset_cre_shan$Means
```

```
[1] 1.891724 1.959202
```

```
subset_cre_shan$StDevs
```

```
[1] 0.09436998 0.03892581
```

```r
ggplot(subset_cre_nucl) +
  geom_bar(aes(x = factor(Field, levels = field_order_nucl),
               y = ifelse(Type == "Enhancer", -Means, Means),
               fill = paste(Type, "Means")),
           stat = "identity", position = "identity",
           alpha = 0.6, width = 0.7) +
  geom_errorbar(aes(x = factor(Field, levels = field_order_nucl),
                    ymin = ifelse(Type == "Enhancer",
                                  -Means + StDevs, Means - StDevs),
                    ymax = ifelse(Type == "Enhancer",
                                  -Means - StDevs, Means + StDevs)),
                width = 0.5, colour = "black", alpha = 0.6) +
  coord_flip() +
  scale_y_continuous(breaks=seq(-1, 1, 0.2), labels=abs(seq(-1, 1, 0.2))) +
  scale_fill_manual(values = c("turquoise", "coral")) +
  labs(y = "Means", x = "Nucleotides",
       title = "Percentage Means per Nucleotide",
       fill = "CRE Type") +
  theme_minimal()
```



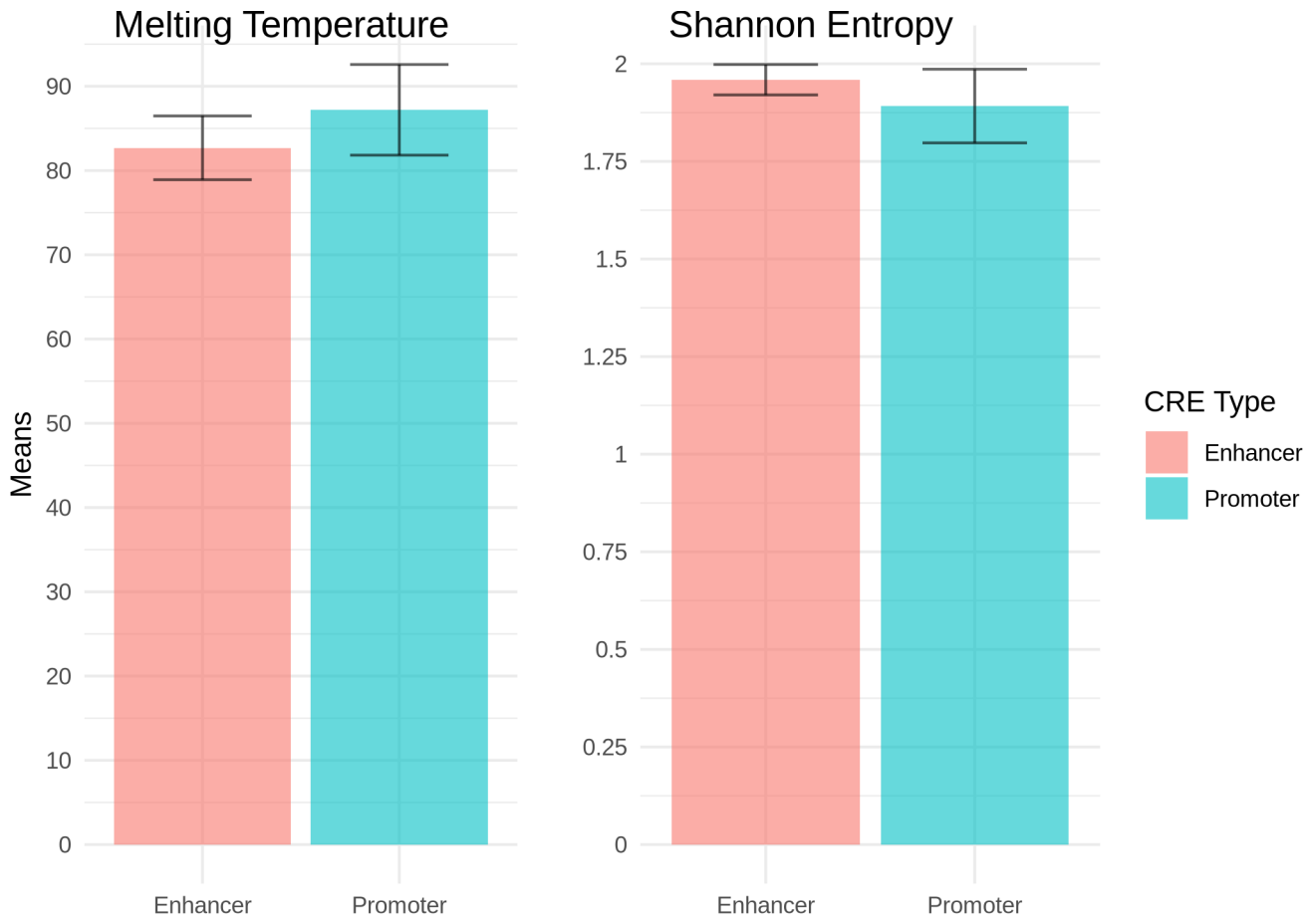

```
library(cowplot)

temp_plt <- ggplot(subset_cre_temp) +
  geom_bar(aes(x = factor(Type),
               y = Means, fill=Type),
           stat = "identity", position = "identity", alpha = 0.6) +
  geom_errorbar(aes(x = factor(Type),
                    ymin = Means - StDevs,
                    ymax = Means + StDevs),
                width = 0.5, colour = "black", alpha = 0.6) +
  labs(y = "Means", x = "",
       # title = "Melting Temperature Means",
       fill = "") +
  scale_y_continuous(breaks = seq(0, 100, 10),
                     labels = seq(0, 100, 10)) +
  guides(fill = "none") +
  theme_minimal()

shan_plt <- ggplot(subset_cre_shan) +
  geom_bar(aes(x = factor(Type),
```

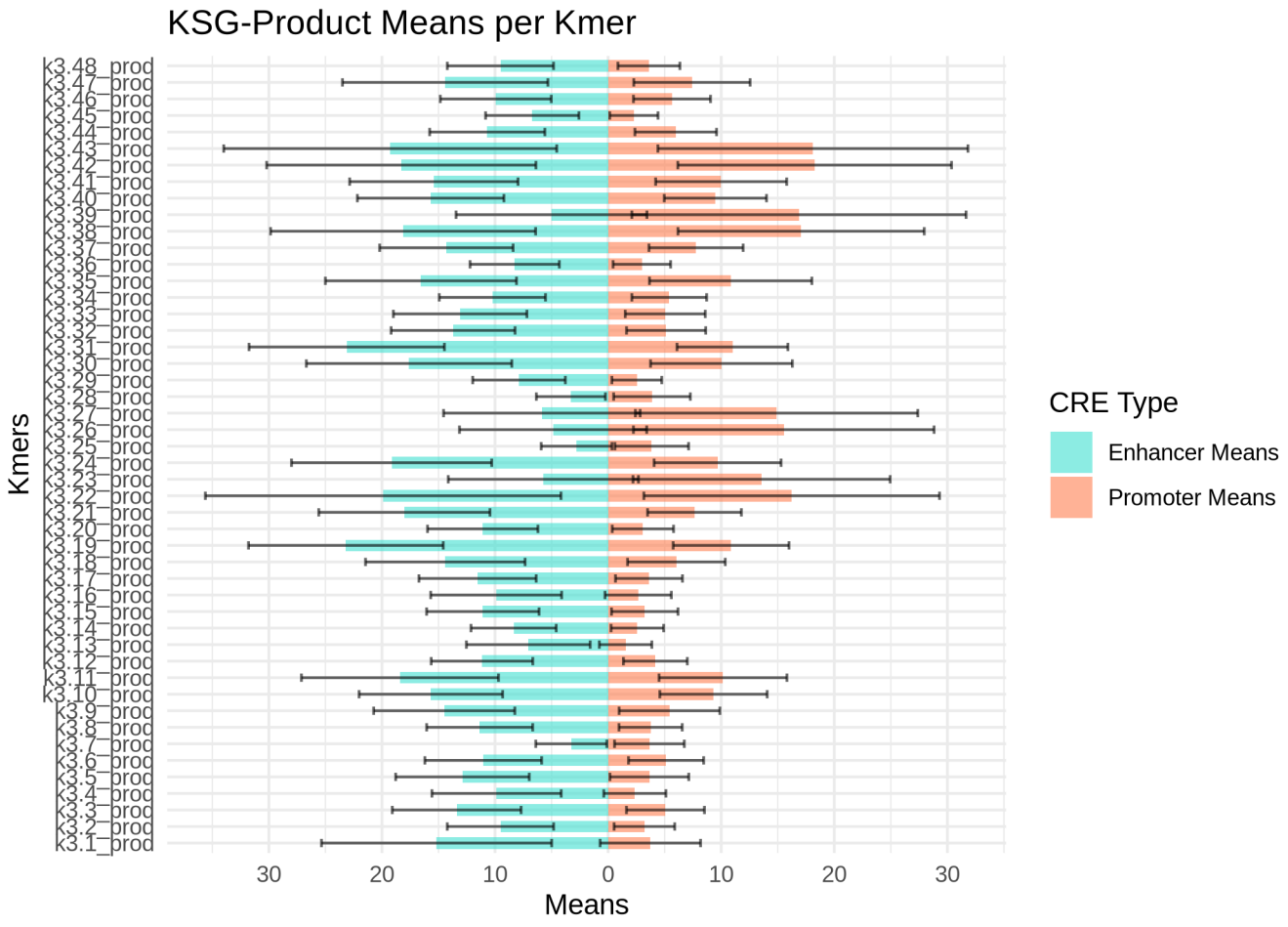
```
              y = Means, fill=Type),
           stat = "identity", position = "identity", alpha = 0.6) +
  geom_errorbar(aes(x = factor(Type),
                    ymin = Means - StDevs,
                    ymax = Means + StDevs),
                width = 0.5, colour = "black", alpha = 0.6) +
  labs(y = "", x = "",
       # title = "Shannon Entropy Means",
       fill = "CRE Type") +
  scale_y_continuous(breaks=seq(0, 2, 0.25), labels=seq(0, 2, 0.25)) +
  theme_minimal()

plot_grid(temp_plt, shan_plt, ncol = 2, rel_widths = c(1,1.5),
          labels = c("Melting Temperature", "Shannon Entropy"),
          label_fontface = "plain",hjust = c(-0.35, -0.48), vjust = 1)
```

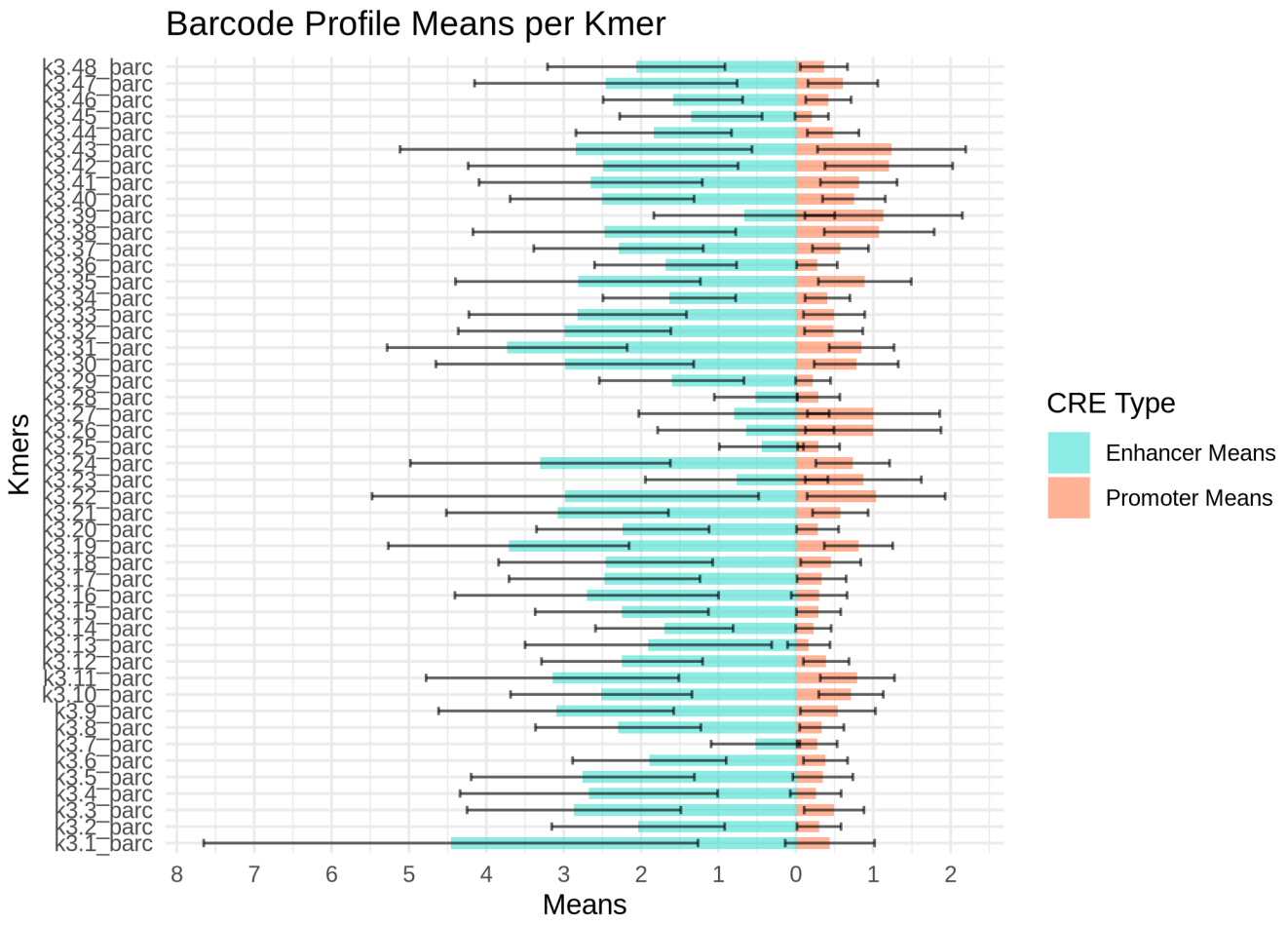



```
ggplot(subset_cre_prod) +
  geom_bar(aes(x = factor(Field, levels = field_order_prod),
               y = ifelse(Type == "Enhancer", -Means, Means),
```

```
                          fill = paste(Type, "Means")),
              stat = "identity", position = "identity",
              alpha = 0.6, width = 0.7) +
geom_errorbar(aes(x = factor(Field, levels = field_order_prod),
                  ymin = ifelse(Type == "Enhancer",
                                -Means + StDevs, Means - StDevs),
                  ymax = ifelse(Type == "Enhancer",
                                -Means - StDevs, Means + StDevs)),
               width = 0.5, colour = "black", alpha = 0.6) +
coord_flip() +
scale_y_continuous(breaks=seq(-30, 30, 10), labels=abs(seq(-30, 30, 10))) +
scale_fill_manual(values = c("turquoise", "coral")) +
labs(y = "Means", x = "Kmers",
     title = "KSG-Product Means per Kmer",
     fill = "CRE Type") +
theme_minimal()
```



```r
ggplot(subset_cre_barc) +
  geom_bar(aes(x = factor(Field, levels = field_order_barc),
               y = ifelse(Type == "Enhancer", -Means, Means),
                         fill = paste(Type, "Means")),
               stat = "identity", position = "identity",
               alpha = 0.6, width = 0.7) +
  geom_errorbar(aes(x = factor(Field, levels = field_order_barc),
                    ymin = ifelse(Type == "Enhancer",
                                  -Means + StDevs, Means - StDevs),
                    ymax = ifelse(Type == "Enhancer",
                                  -Means - StDevs, Means + StDevs)),
                width = 0.5, colour = "black", alpha = 0.6) +
  coord_flip() +
  scale_y_continuous(breaks=seq(-10, 10, 1), labels=abs(seq(-10, 10, 1))) +
  scale_fill_manual(values = c("turquoise", "coral")) +
  labs(y = "Means", x = "Kmers",
       title = "Barcode Profile Means per Kmer",
       fill = "CRE Type") +
  theme_minimal()
```


Barcode Profile Means per Kmer
Kmers
k3.48_barc
k3.47_barc
k3.46_barc
k3.45_barc
k3.44_barc
k3.43_barc
k3.42_barc
k3.41_barc
k3.40_barc
k3.39_barc
k3.38_barc
k3.37_barc
k3.36_barc
k3.35_barc
k3.34_barc
k3.33_barc
k3.32_barc
k3.31_barc
k3.30_barc
k3.29_barc
k3.28_barc
k3.27_barc
k3.26_barc
k3.25_barc
k3.24_barc
k3.23_barc
k3.22_barc
k3.21_barc
k3.20_barc
k3.19_barc
k3.18_barc
k3.17_barc
k3.16_barc
k3.15_barc
k3.14_barc
k3.13_barc
k3.12_barc
k3.11_barc
k3.10_barc
k3.9_barc
k3.8_barc
k3.7_barc
k3.6_barc
k3.5_barc
k3.4_barc
k3.3_barc
k3.2_barc
k3.1_barc
8
7
6
5
4
3
2
1
0
1
2
Means
CRE Type
Enhancer Means
Promoter Means