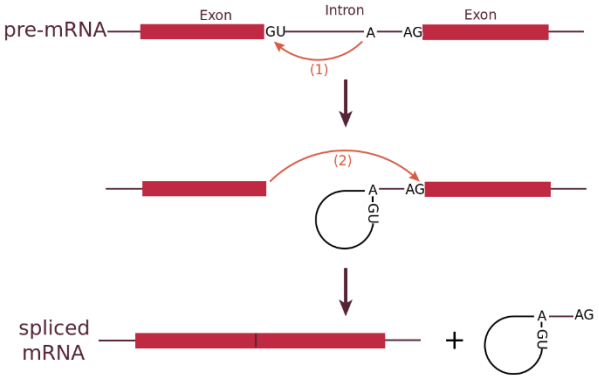
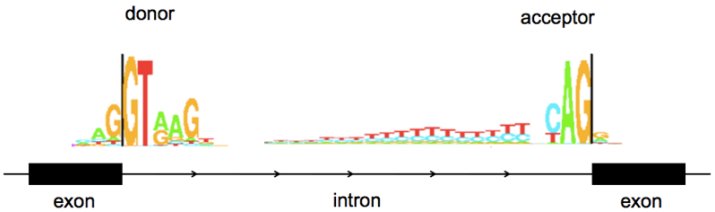
SESSION 13. PRACTICE

Finding genes: In the world of snurps

RNA splicing and it's sequence features





https://en.wikipedia.org/wiki/RNA_splicing
http://www.cureffi.org/2015/10/09/is-prnp-mrna-alternatively-spliced/

Identification of splice sites with a PSSM

	1	2	3	4	5	6	7	8	9
S1	C	A	G	G	T	A	G	G	G
S2	C	A	G	G	T	T	A	C	A
S3	A	A	G	G	T	A	T	G	T
S4	G	A	G	G	T	G	A	G	C
S5	G	A	G	G	T	A	A	A	C
S6	A	G	A	G	T	A	A	G	G
S7	C	G	G	G	T	G	G	G	T
S8	G	T	G	G	T	G	A	T	T
S9	A	C	A	G	T	A	A	C	T
S10	C	T	T	G	T	A	A	G	T

Frequency

	1	2	3	4	5	6	7	8	9
A	3	5	2	0	0	6	7	1	1
T	0	2	1	0	10	1	1	1	5
C	4	1	0	0	0	0	0	2	2
G	3	2	7	10	0	3	2	6	2

Pseudocount

	1	2	3	4	5	6	7	8	9
A	4	6	3	1	1	7	8	2	2
T	1	3	2	1	11	2	2	2	6
C	5	2	1	1	1	1	1	3	3
G	4	3	8	11	1	4	3	7	3

Identification of splice sites with a PSSM

	1	2	3	4	5	6	7	8	9
A	4	6	3	1	1	7	8	2	2
T	1	3	2	1	11	2	2	2	6
C	5	2	1	1	1	1	1	3	3
G	4	3	8	11	1	4	3	7	3

Probability

	1	2	3	4	5	6	7	8	9
A	0.29	0.43	0.21	0.07	0.07	0.50	0.57	0.14	0.14
T	0.07	0.21	0.14	0.07	0.79	0.14	0.14	0.14	0.43
C	0.36	0.14	0.07	0.07	0.07	0.07	0.07	0.21	0.21
G	0.29	0.21	0.57	0.79	0.07	0.29	0.21	0.50	0.21

Log observed/expected ratio

	1	2	3	4	5	6	7	8	9
A	0.13	0.54	-0.15	-1.25	-1.25	0.69	0.83	-0.56	-0.56
T	-1.25	-0.15	-0.56	-1.25	1.15	-0.56	-0.56	-0.56	0.54
C	0.36	-0.56	-1.25	-1.25	-1.25	-1.25	-1.25	-0.15	-0.15
G	0.13	-0.15	0.83	1.15	-1.25	0.13	-0.15	0.69	-0.15

$$\frac{\text{Number of As at position 1}}{\text{Total number of sequences at position 1}} = \frac{4}{14} = 0.29$$

$$log(\frac{Actual\ probability\ of\ A\ at\ position\ 1}{Expected\ probability\ of\ A\ at\ position\ 1}) = \log(\frac{0.29}{0.25}) = 0.13$$

TCCTGTCC**CAGGTAGGG**AA

score = 5.11

Basic Shell Commands

```
$ cd [User_Folder]
$ mkdir Session13
$ cd Session13
```

Constructing a PSSM

```
$cp /home/biguser/tutor/Session13/splice5.txt .
$less splice5.txt
```

CAGGTAGGG
CAGGTAACA
AAGGTAAGT
GAGGTGAGC
GAGGTAAAC
AAAGTAAGG

Constructing a PSSM

```
$ vi make_matrix5.py
```

```
#!/usr/bin/python

import sys, math

splice5 = sys.argv[1] #'splice5.txt'
number_of_sequences = 0

for line in open(splice5):
        line = line.rstrip()
        if number_of_sequences == 0:
                msa_matrix = [[]]
        if number_of_sequences > 0:
                msa_matrix.append([])
        for j in range(0, 9):
                msa_matrix[number_of_sequences].append(line[j])
        number_of_sequences += 1

print msa_matrix
```

Constructing a PSSM

```
$ python make_matrix5.py splice5.txt
```

```
[kyungtae@biglab-master session12]$ python make_matrix5.py splice6.txt
[['C', 'A', 'G', 'G', 'T', 'A', 'G', 'G', 'G'], ['C', 'A', 'G', 'G', 'T', 'A', 'A', 'C', 'A'],
 ['A', 'A', 'G', 'G', 'T', 'A', 'A', 'G', 'T'], ['G', 'A', 'G', 'G', 'T', 'G', 'A', 'G', 'C'],
 ['G', 'A', 'G', 'G', 'T', 'A', 'A', 'A', 'C'], ['A', 'A', 'A', 'G', 'T', 'A', 'A', 'G', 'G']]
```

1	2	3	4	5	6	7	8	9
C	A	G	G	T	A	G	G	G
C	A	G	G	T	A	A	C	A
A	A	G	G	T	A	A	G	T
G	A	G	G	T	G	A	G	C
G	A	G	G	T	A	A	A	C
A	A	A	G	T	A	A	G	G

Constructing a PSSM

```
#print msa_matrix

bases = ['A', 'T', 'C', 'G']
pssm = [[]]

for i in range(0, 4):
        if i > 0:
                pssm.append([])
        for j in range(0, 9):
                pssm[i].append(1.0)
                for k in range(0, number_of_sequences):
                        if msa_matrix[k][j] == bases[i]:
                                pssm[i][j] += 1

print pssm
```

pseudocount

Constructing a PSSM

```
$ python make_matrix5.py splice5.txt
```

```
[kyungtae@biglab-master session12]$ python make_matrix5.py splice6.txt
[[3.0, 7.0, 2.0, 1.0, 1.0, 6.0, 6.0, 2.0, 2.0], [1.0, 1.0, 1.0, 1.0, 7.0, 1.0, 1.0, 1.0, 2.0],
 [3.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 2.0, 3.0], [3.0, 1.0, 6.0, 7.0, 1.0, 2.0, 2.0, 5.0, 3.0]]
```

	1	2	3	4	5	6	7	8	9
A	3	7	2	1	1	6	6	2	2
T	1	1	1	1	7	1	1	1	2
C	3	1	1	1	1	1	1	2	3
G	3	1	6	7	1	2	2	5	3

Constructing a PSSM

```
#print pssm

for i in range(0, 4):
        for j in range(0, 9):
                pssm[i][j] = math.log(pssm[i][j] / (number_of_sequences
 + 4)*4) / math.log(2)
                print pssm[i][j],
        print ''
```

$$log_2(\frac{Actual\ probability\ of\ nucleotide\ at\ position\ J}{Expected\ probability\ of\ nucleotide\ at\ position\ J}) =$$
$$log_2(\frac{Actual\ probability}{0.25}) = log_2(Actual\ probability*4)$$

Syntax

```
math.log(x, base)
```

Parameter Values

Parameter	Description
x	Required. Specifies the value to calculate the logarithm for. If the value is 0 or a negative number, it returns a ValueError. If the value is not a number, it returns a TypeError
base	Optional. The logarithmic base to use. Default is 'e'

$$\frac{logB}{logA} = logAB$$

Constructing a PSSM

```
$ python make_matrix5.py splice5.txt
```

```
[kyungtae@biglab-master session12]$ python make_matrix5.py splice6.txt
0.26 1.49 -0.32 -1.32 -1.32 1.26 1.26 -0.32 -0.32
-1.32 -1.32 -1.32 -1.32 1.49 -1.32 -1.32 -1.32 -0.32
0.26 -1.32 -1.32 -1.32 -1.32 -1.32 -1.32 -0.32 0.26
0.26 -1.32 1.26 1.49 -1.32 -0.32 -0.32 1.0 0.26
```

	1	2	3	4	5	6	7	8	9
A	0.26	1.49	-0.32	-1.32	-1.32	1.26	1.26	-0.32	-0.32
T	-1.32	-1.32	-1.32	-1.32	1.49	-1.32	-1.32	-1.32	-0.32
C	0.26	-1.32	-1.32	-1.32	-1.32	-1.32	-1.32	-0.32	0.26
G	0.26	-1.32	1.26	1.49	-1.32	-0.32	-0.32	1.0	0.26

```
$ python make_matrix5.py splice5.txt > matrix5.txt
```

Scoring with a PSSM

```
$ cp /home/biguser/tutor/Session13/amyloid.fa .
$ less amyloid.fa
```

```
>amyloid
agccatcacttgtctctaataaataactcccattgattttccagctcagggctcaccact
ccttaccgtaagcgcaggaggagactggaaaatcactcacatattattggtgctcttcct
cccccatcctcacccaaggtgcatataaaccctgaataacctgaagtctaagggcatgaa
tatcagacgctaggggacagccactgtgttgtctgctaccctcatcctggtcactgctt
ctgctataacagccctaggccaggaatatgaacaagccgctgctttggatctctgtcctc
accagcctcctggaagcctttgctcacacaggtaaggaggtgaaggaatggtcaagaatc
ataaagtgagaaaataggttgaagctgagatatcttttccctgcatttatactgaaggtc
attatctttctttctttatcccgcagacctcagtgggaaggtgtttgtatttcctagaga
```

Scoring with a PSSM

```
$ vi score5.py
```

```
#!/usr/bin/python

import sys, re

matrix5 = sys.argv[1] #'matrix5.txt'
amyloid = sys.argv[2] #'amyloid.fa'

i = 0
for line in open(matrix5):
        line = line.rstrip()
        if i == 0:
                pssm = [[]]
        if i > 0:
                pssm.append([])
        col = line.split()
        for j in range(0, 9):
                pssm[i].append(float(col[j]))
        i += 1
```

```
seq = ''
for line in open(amyloid):
        if not re.search('>', line):
                line = line.rstrip()
                seq += line
```

:

:

Scoring with a PSSM

```
print 'pos\tscore' # print header

seq = seq.upper()
bases = ['A', 'T', 'C', 'G']

for k in range(0, len(seq) -  8)
        test = seq[k:k + 9]
        score = 0
        for j in range(0, 9):
                base = test[j]
                for b in range(0, 4):
                        if bases[b] == base:
                                score += pssm[b][j]

        score = 2 ** score # convert log2 to real values
                           # ** is the exponentiation operator
        pos = k + 3 # We want to print a position
                    # next to the exon-intron
                    # junction
        print pos, '\t', score
```

Scoring with a PSSM

```
$ python score5.py matrix5.txt amyloid.fa
```

pos	score
3	3.50757912018e-11
4	6.77423367512e-06
5	5.77683587552e-09
6	6.27879781272e-10
7	1.61302923014e-09
8	7.45036126041e-07
9	0.000130590740603
10	7.02109803694e-11
11	0.0682186378424
12	2.7462545595e-10
13	5.19995200311e-07
14	5.74149862186e-10
15	6.68280376884e-05

```
$ python score5.py matrix5.txt amyloid.fa > score5.txt
```

Visualization with R

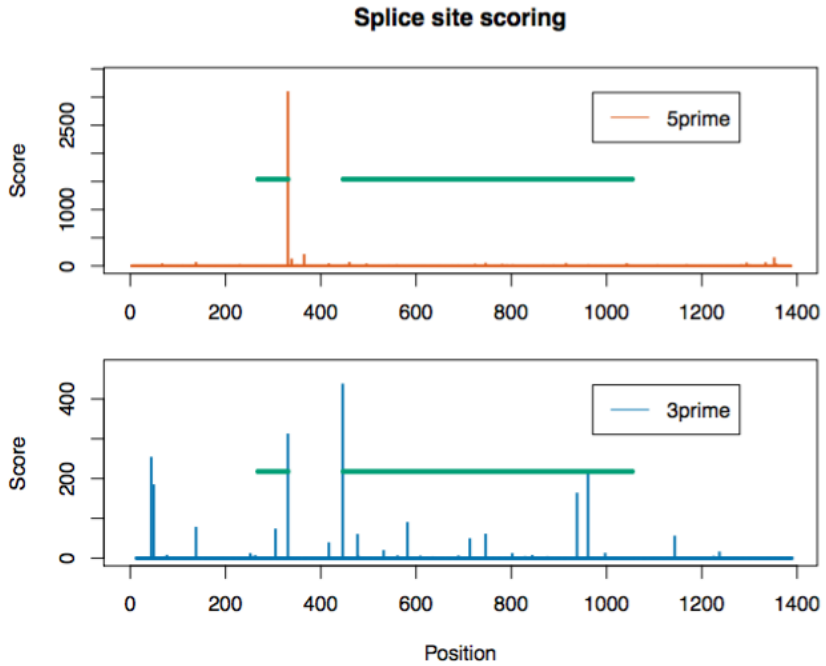
```
$ cp /home/biguser/tutor/Session13/score3.txt .
$ cp /home/biguser/tutor/Session13/amyloid.r .
$ vi amyloid.r
```

```
# plot the results of splice site prediction

#define some colours
rgb <- c("#009E73", "#D55E00", "#0072B2")

# make two graphs on top of each other
par(mfrow = c(2, 1))

# First consider the 5' splice site prediction and read the
#   output from
# the Perl code
data <- read.table("score5.txt", sep = "\t", header = TRUE)

# for the plot, we need to know about the sequence length
seqlen <- max(data$pos)

# for the plot we need to know about the maximum score
max_score <- max(data$score)

# make a plot for the 5' splice site data
plot(0, type = "n", lwd = 2, xlim = c(0, seqlen),
    ylim = c(0, max_score * 1.1), main = "Splice site scoring",
    xlab = "Position", ylab = "Score")

#print a legend
legend(seqlen * 0.7, max_score, "5prime", col = rgb[2],
    lwd = 1)

# plot the splice site scores
for (i in (1:seqlen)) {
    lines(c(data$pos[i], data$pos[i]), c(0, data$score[i]), col = rgb[2],
        lw = 2)
}
```

```
# plot the location of exons (we do not know these from the
#   prediction)
lines(c(268, 331), c(max_score/2, max_score/2), col = rgb[1],
    lw = 4)
lines(c(447, 1054), c(max_score/2, max_score/2), col = rgb[1],
    lw = 4)

# Now consider the 3' splice site and read the output from
# the Perl code
data <- read.table("score3.txt", sep = "\t", header = TRUE)

max_score <- max(data$score)

plot(0, type = "n", lwd = 2, xlim = c(0, seqlen),
    ylim = c(0, max_score * 1.1), xlab = "Position", ylab = "Score")

legend(seqlen * 0.7, max_score, "3prime", col = rgb[3],
    lwd = 1)

for (i in (1:seqlen)) {
    lines(c(data$pos[i], data$pos[i]), c(0, data$score[i]), col = rgb[3],
        lw = 2)
}

# plot the location of exons as in the previous graph
lines(c(268, 331), c(max_score/2, max_score/2), col = rgb[1],
    lw = 4)
lines(c(447, 1054), c(max_score/2, max_score/2), col = rgb[1],
    lw = 4)
```

Visualization with R

```
$ Rscript amyloid.r
Open "Rplots.pdf"
```



Exercise

- When you construct a PSSM, you divided observed frequency by expected frequency ($M_{ij} = \log(F_{ij}/F_{exp})$, $F_{exp} = 0.25$). In real, however, those four bases (A, C, G, T) are not evenly distributed across a genome, which means that expected frequencies for each base are not equally 0.25 (1/4). It would be much more precise to use 'observed frequency across the matrix' ($M_{ij} = \log(F_{ij}/F_{exp})$, $F_{exp} = P_i/\text{Total P}$, i=(A,C,G,T)). Correct a Python script 'make_matrix5.py' to calculate M_{ij} which is divided by 'observed frequency across matrix'.

	1	2	3	4	5	6	7	8	9
A	0.29	0.43	0.21	0.07	0.07	0.50	0.57	0.14	0.14
T	0.07	0.21	0.14	0.07	0.79	0.14	0.14	0.14	0.43
C	0.36	0.14	0.07	0.07	0.07	0.07	0.07	0.21	0.21
G	0.29	0.21	0.57	0.79	0.07	0.29	0.21	0.50	0.21

Log-odd: $\log(F_{ij}/F_{exp})$

	1	2	3	4	5	6	7	8	9
A	0.06	0.46	-0.23	-1.33	-1.33	0.62	0.75	-0.64	-0.64
T	-1.20	-0.11	-0.51	-1.20	1.19	-0.51	-0.51	-0.51	0.59
C	0.92	0.00	-0.69	-0.69	-0.69	-0.69	-0.69	0.41	0.41
G	-0.20	-0.49	0.49	0.81	-1.59	-0.20	-0.49	0.36	-0.49

$F_{exp} = P_i/\text{Total P}$, i=(A,C,G,T)

Exercise

```
for i in range(0, 4):
        baseFreq = sum(pssm[i])/(sum(pssm[0]) + sum(pssm[1]) + sum(pssm[2])
+ sum(pssm[3])) ##
        for j in range(0, 9):
#               pssm[i][j] = math.log(pssm[i][j] / (number_of_sequences + 4)
*4) / math.log(2)
                pssm[i][j] = math.log(pssm[i][j] / (number_of_sequences + 4)
 / baseFreq) / math.log(2) ##
                print pssm[i][j],
        print ''
```

```
0.265919483949 1.12652055534 -1.56469363877 -12.0404270697 -12.0404270697
 0.930397831623 1.34839746828 -1.95031465008 -0.826107948939
-1.38692779482 -1.09028903869 -2.0553648226 -12.1691069887 1.71982647648
-3.58039235307 -1.67225321126 -2.35052481117 0.603001353401
0.958039470832 -0.536795483751 -2.53027563566 -11.493171641 -11.493171641
 -2.78581250892 -1.30211242647 -1.76185260998 -0.194536234266
-2.46759736139 -3.05676750648 -0.323757755326 -0.00230509133455 -13.89123
85565 -1.26119900381 -3.0536106233 -0.301587409954 -2.21175845696
```