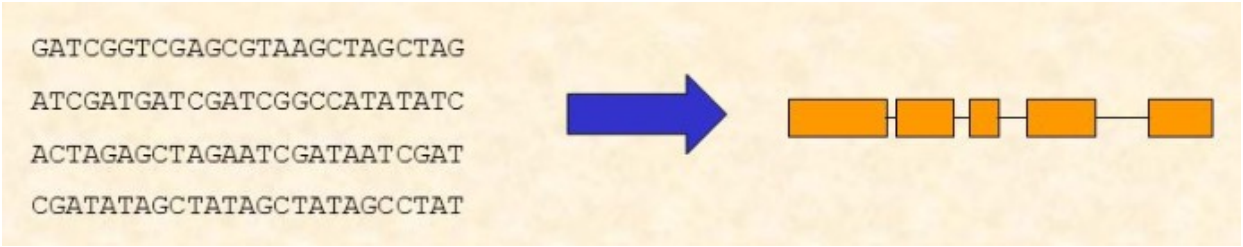
SESSION 13. FINDING GENES

In the world of snurps

Methods of gene prediction

- Computational gene prediction
  - De novo (or ab initio) approaches
    - Use statistical signals in DNA sequences that are characteristics of protein-coding genes
  - Homology-based approaches
    - Aligning to known mRNA or protein sequences or even HMMs to a genomic sequence
    - Require that known mRNAs and protein sequence information is available



cDNA sequencing(short or long) → full-length gene

Signals for *de novo* prediction

- Signals for de novo prediction
  - Words (nucleotide k-mer) frequencies from coding sequences (Bishop 1994) + Start/Stop codons
  - Exon-Intron boundary in Eukaryotes
  - Length of exons and introns (not random)
  - Polyadenylation signals (AATAAA)

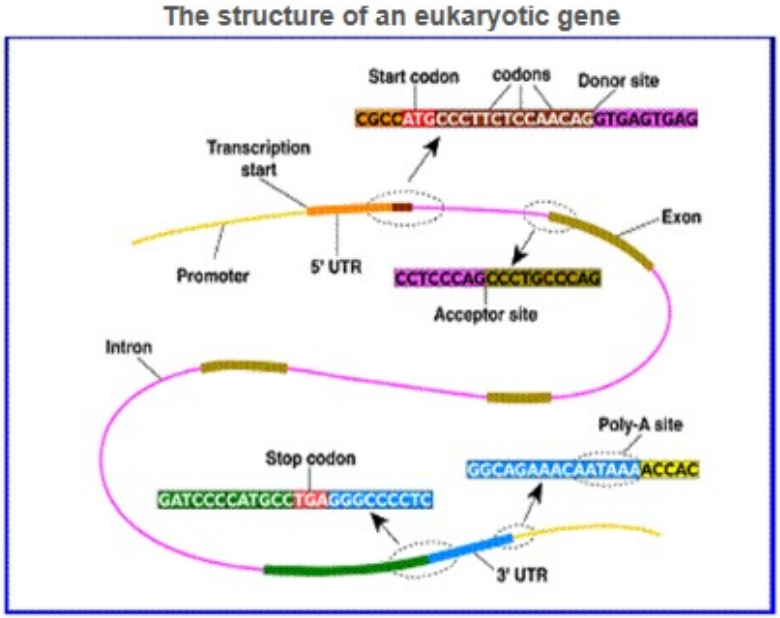
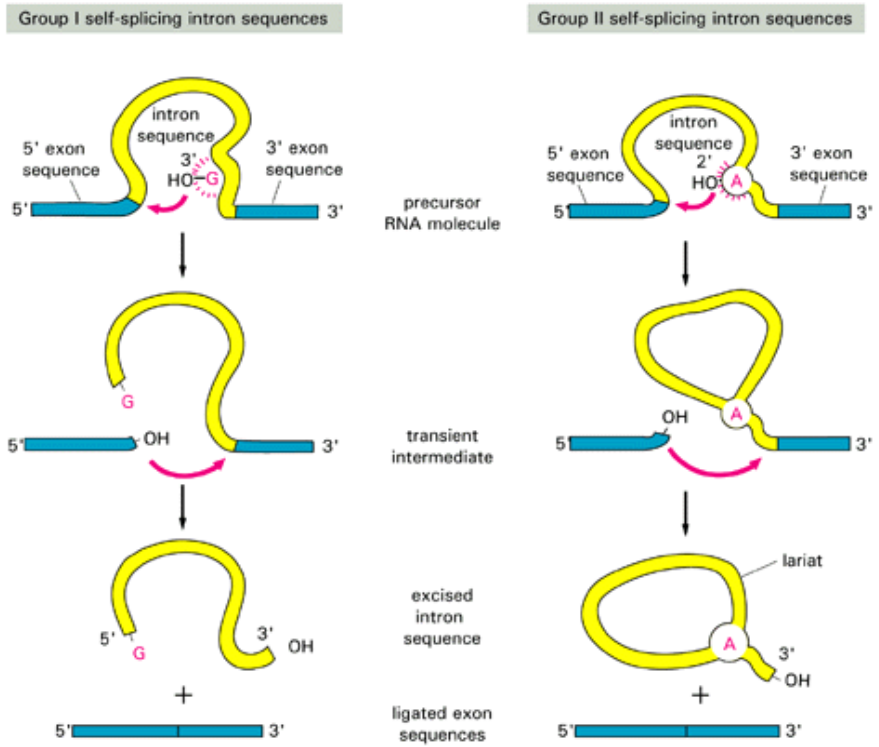


Figure 3. The structure of an eukaryotic gene (source: unknown).

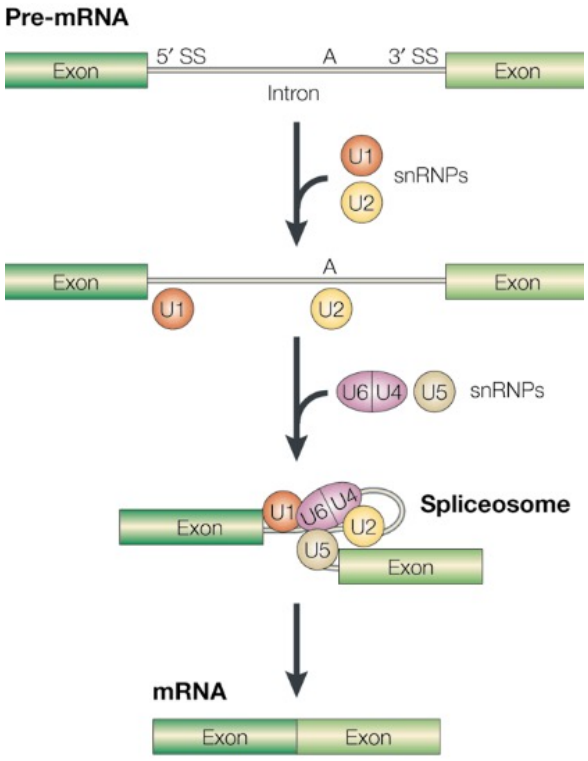
The splicing machinary

- Splicing
  - Two sequential trans-esterification reactions by spliceosomes
  - First, a branch site adenosine attacks the 5’ splice site (donor site)
  - Second, the free 3’ end of the first exon attacks the 3’ splice site (acceptor site)
  - Then, two exons become covalently joined and the intron is released in the form of a lariat structure

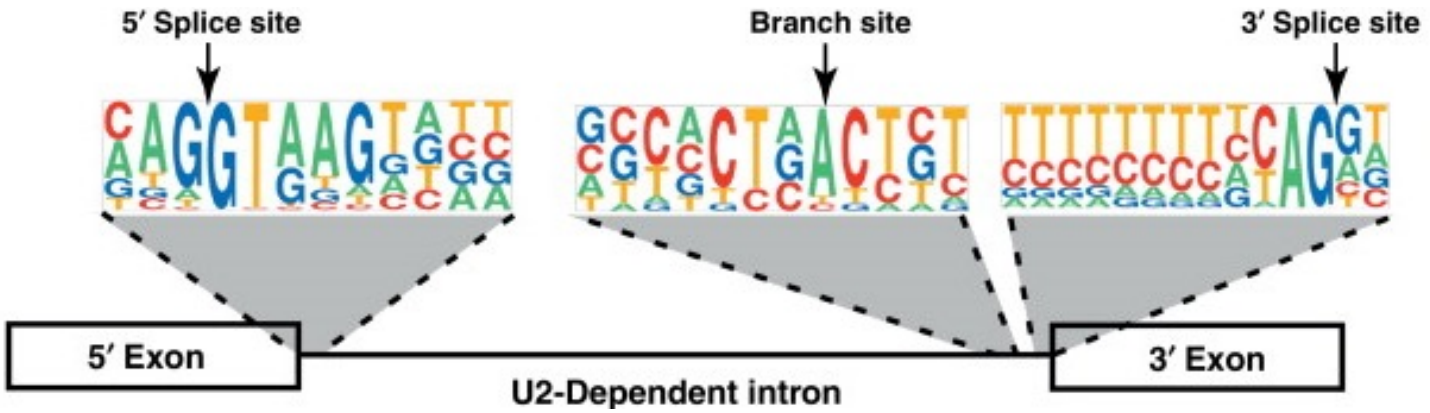


The spliceosomes

- Spliceosomes (RNA+protein complexes)
  - Includes a number of snRNPs (pronounced snurps)
  - U1, U2, U4, U5, and U6 snRNPs
  - U1 interacts with 5' splice site and U2 interacts with the branch site



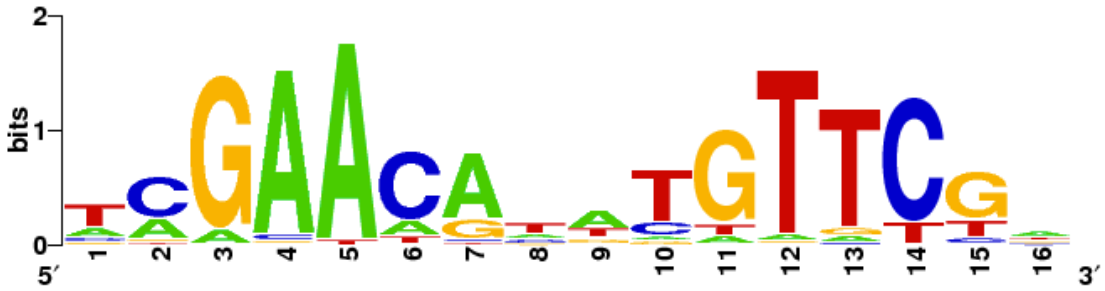
Exon-junction motif analysis reveals splice signals (GU-AG + branch site)



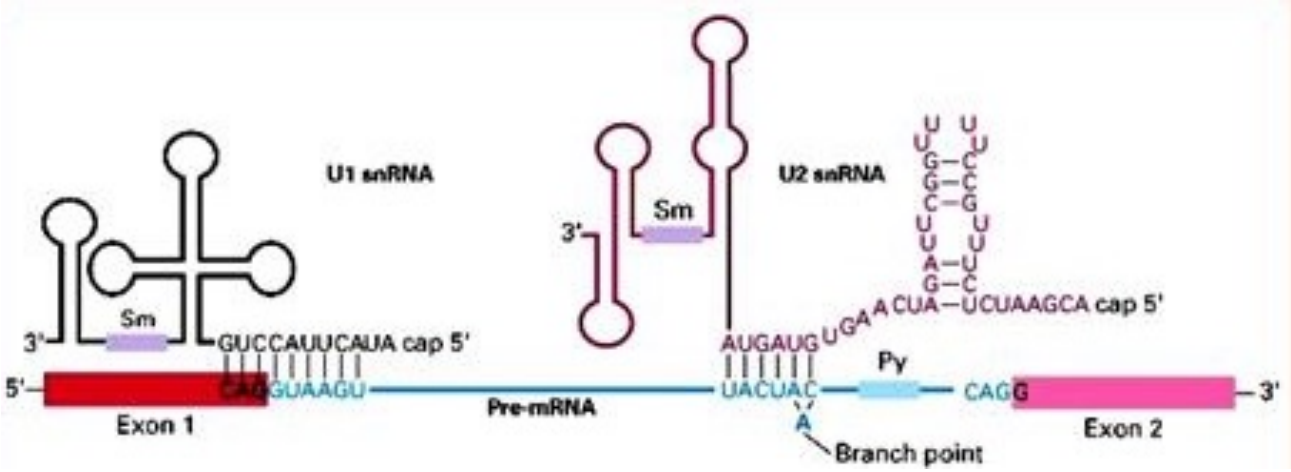
Sequence logo height = 2-entropy(nt)
Proportion = proportion of ACGT

Entropy = $-\sum p \log_2 p$

GCCACTA
GCCACTA
CTCAGAC
CTCACGT
AGCTGCT
TTCTGTC
TTCTGTC
ATCTGTC



Exon-junction signals can be defined by U1 and U2 base pairing



Splice site 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

Splice site PSSM

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-odd : log (Fij/Fexp)

	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

Or

	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

TCCTGTCCCAGGTAGGGAA

Fexp = .25

Fexp = Pi / Total P, i={A, C, G, T}

make_matrix.py

```
#!/usr/bin/python

import math, sys
splicefile = sys.argv[1]
number_of_sequences = 0  # we want to count the number of
                         # sequences in the file splice5.txt
for line in open(splicefile):
    line = line.rstrip()
    if number_of_sequences == 0:
        msa_matrix = [[] * 9]
            # two dimensional array to
            # to store multiple alignment
            # create first empty row
        print msa_matrix
    if number_of_sequences > 0:
        msa_matrix.append([])  # add one row
    for j in range(0, 9):  # fill the row with numbers
        msa_matrix[number_of_sequences].append(line[j])
        if number_of_sequences==0: print msa_matrix
    number_of_sequences += 1
```

```
# produce count matrix
bases = ['A', 'T', 'C', 'G']
pssm = [[] * 9]

for i in range(0, 4):
    if i > 0:
        pssm.append([])
    for j in range(0, 9):
        # add pseudocount = 1 to each of th
        pssm[i].append(1.0)
        # add counts to the pssm matrix
        for k in range(0, number_of_sequenc
            if msa_matrix[k][j] == bases[i]
                pssm[i][j] += 1

# from count matrix produce PSSM by
# calculating the log odds values
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 PSSM
    print ''
```

```
GAGGTAAAC
AAAGTAAGG
CAGGTGGGT
GTGGTGAGT
ACAGTAAGT
CTTGTAAGT
AAGGTACTT
GAGGTAAGC
CAGGTTTGT
AAGGTAGGC
GAGGTGAGC
ACTGTACGT
AAGGTACCA
CAGGTCAGT
TCCGTGAGT
AAGGTAAAA
ACAGTAAGT
GAGGTATTT
AAGGTCAGT
CAGGTAATG
CAGGTGAGT
CAGGTACAG
TAAGTACGT
GAGGTAAGG
ATCGTAAGT
AAAGTAAGT
AAGGTGCGG
CAGGTAAGG
GAGGTAAGG
AAGGTCAGT
CAGGTAATG
AAAGTAAGT
CAGGTAAAA
TGGGTGAGT
AAGGTACGG
ACGGTGAGC
CAGGTGGGG
CTGGTGGGT
CAGGTGAGG
GAGGTGCGG
CAGGTAAGG
TGGGTGAGT
GGAGTAAGT
AGGGTGAGG
ATGGTGAGT
"splice5.txt" 15169L, 151690C
```

```
python make_matrix.py splice5.txt >matrix5.txt
```

```
0.417127811215887 1.27772888260454 -1.41348531150622 -11.8892187424726 -11.8892187424726 1.08160615889 1.49960579555099 -1.79910632280832 -0.674899621671848
-1.10703954864141 -0.810400792510635 -1.77547657642343 -11.8892187424726 1.99971472266178 -3.30050410689035 -1.39236496508457 -2.07063656499176 0.882889599579502
0.561992369359715 -0.932842585222944 -2.92632273713535 -11.8892187424726 -11.8892187424726 -3.18185961039173 -1.69815952794096 -2.15789971144755 -0.590583335738372
-0.465577547396757 -1.05474769248839 1.67826205867043 1.99971472266178 -11.8892187424726 0.740820810183483 -1.05159080930121 1.70043240404217 -0.209738642967168
```

score.py

```
#!/usr/bin/python
import re, sys
# input file
pssmfile = sys.argv[1]
# read matrix
i = 0
for line in open(pssmfile):
    line = line.rstrip()
    if i == 0:
        pssm = [[] * 9]
    if i > 0:
        pssm.append([])  # add one row
    col = re.split(' ', line)
    for j in range(0, 9):  # fill the row with numbers
        pssm[i].append(float(col[j]))
    i += 1

# read sequence to be analyzed
seq = ''
for line in open('amyloid.fa'):

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

```
print 'pos\tscore'  # print header
seq = seq.upper()
bases = ['A', 'T', 'C', 'G']

# score with the matrix
for k in range(0, len(seq) - 9):
    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
                        # ** is the ex
    pos = k + 3  # We want to print a
                 # next to the exon-in
                 # junction
    print pos, '\t', score
```

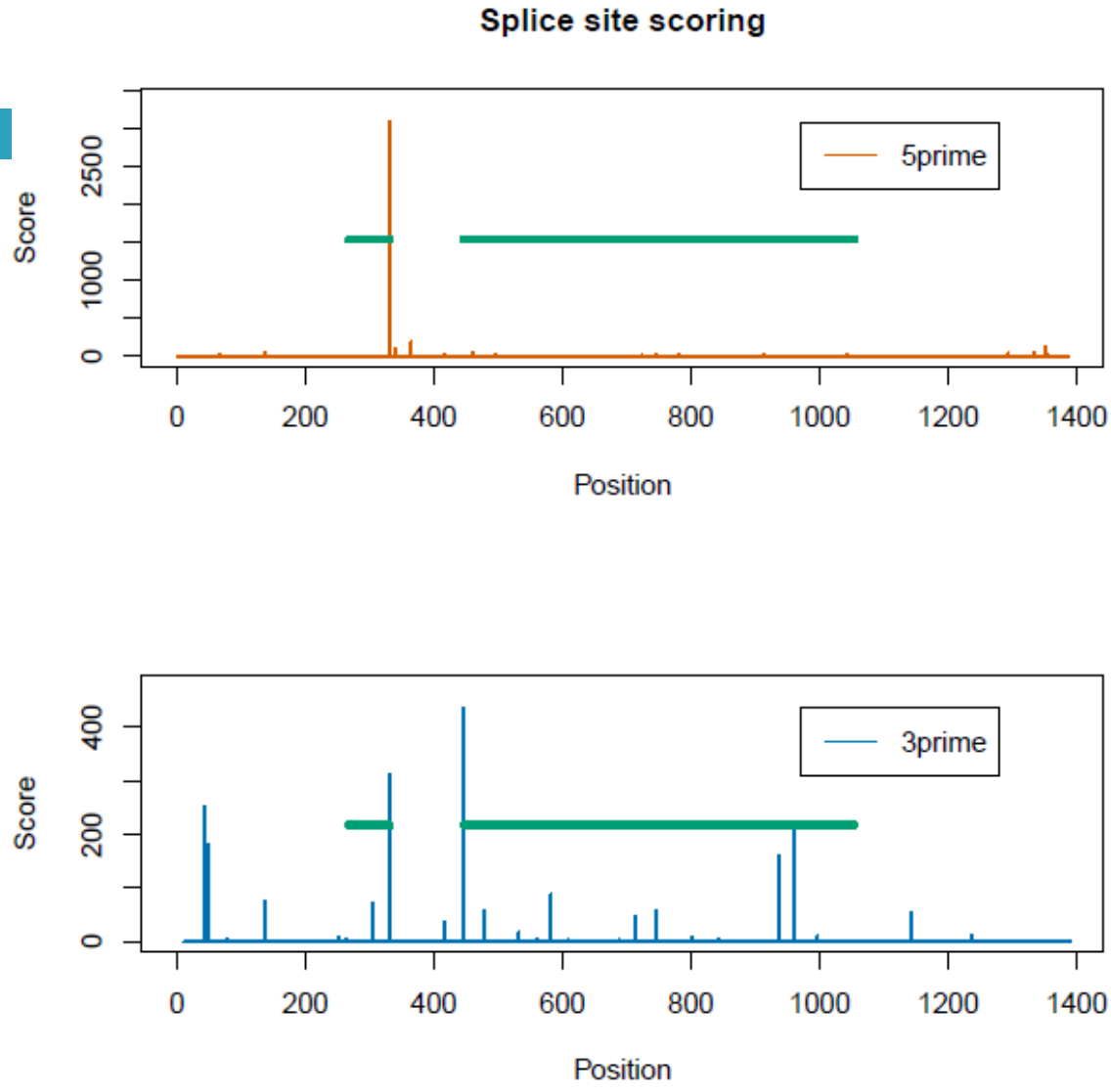
```
862     9.10699121732e-05
863     0.0421877149233
864     3.47543238438e-08
865     8.90678251324e-10
866     7.87686655541e-11
867     1.69153038928
868     0.000283434519102
869     0.000139474591032
870     1.67929615407e-09
871     6.57510980056e-05
872     0.0018565833872
873     1.16798994267e-09
874     3.63513668515e-07
875     4.57861247817e-05
876     0.000274541015957
877     1.2063875477e-09
878     0.000257659537677
879     6.30368794006e-08
880     5.56689797646e-10
881     4.48265957774e-11
882     0.0692197631
883     1.46713237187e-08
884     1.03850108703e-10
885     1.32711032479e-05
886     3.39948840564e-07
887     0.000426052487362
888     2.35157804612e-09
889     2.02971238834
```

```
python score.py matrix5.txt > score5
```

Visualization with 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)
```


Splice site scoring
5prime
Score
0
1000
2500
0
200
400
600
800
1000
1200
1400
Position
3prime
Score
0
200
400
0
200
400
600
800
1000
1200
1400
Position