

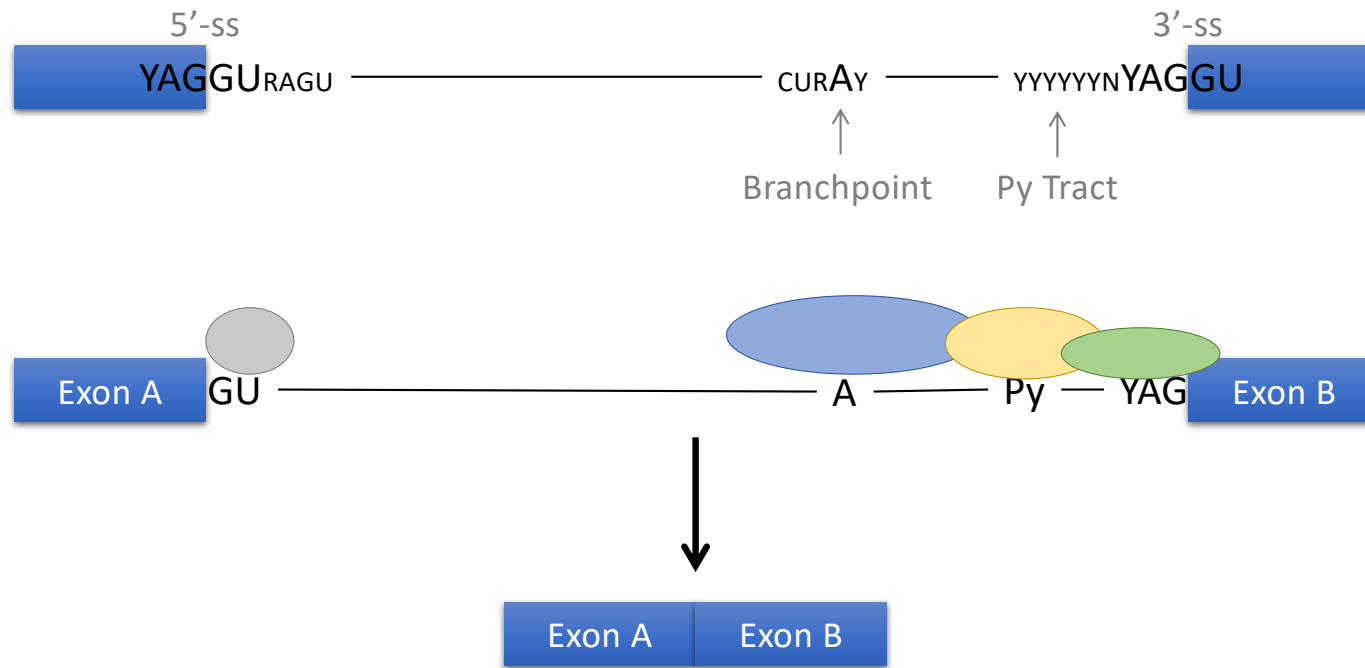
Splicing/Motif Mark

February 13, 2020

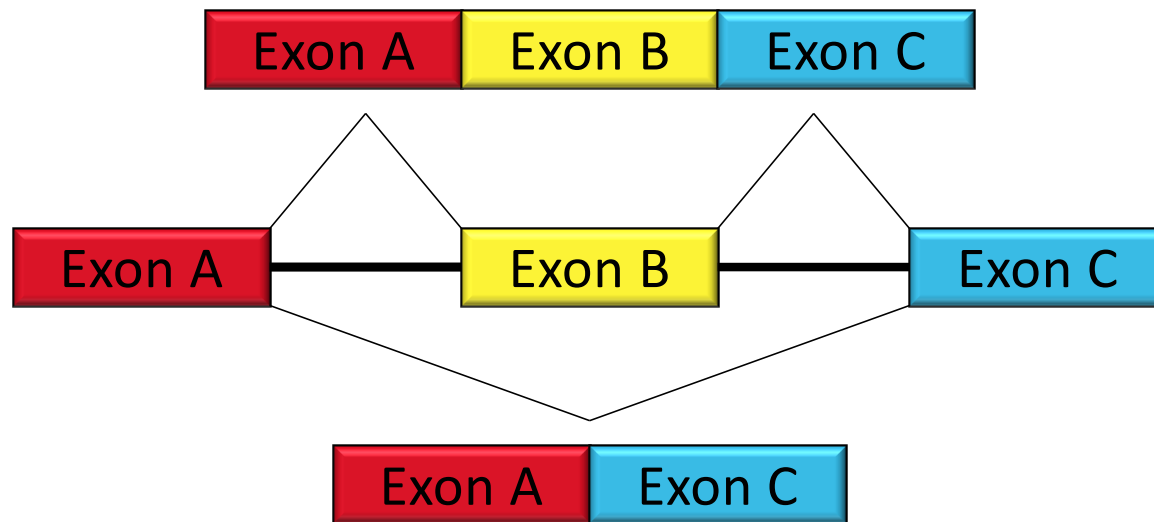
Observations

- MBNL1 and RBFOX1 are both regulators of alternative splicing

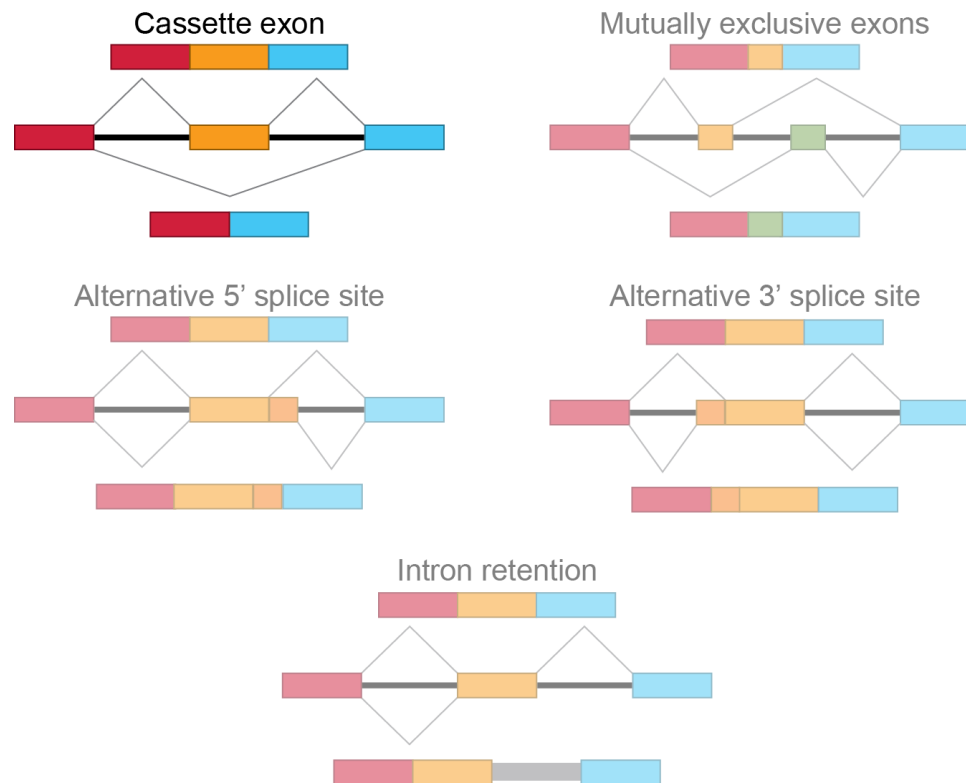
What is splicing?



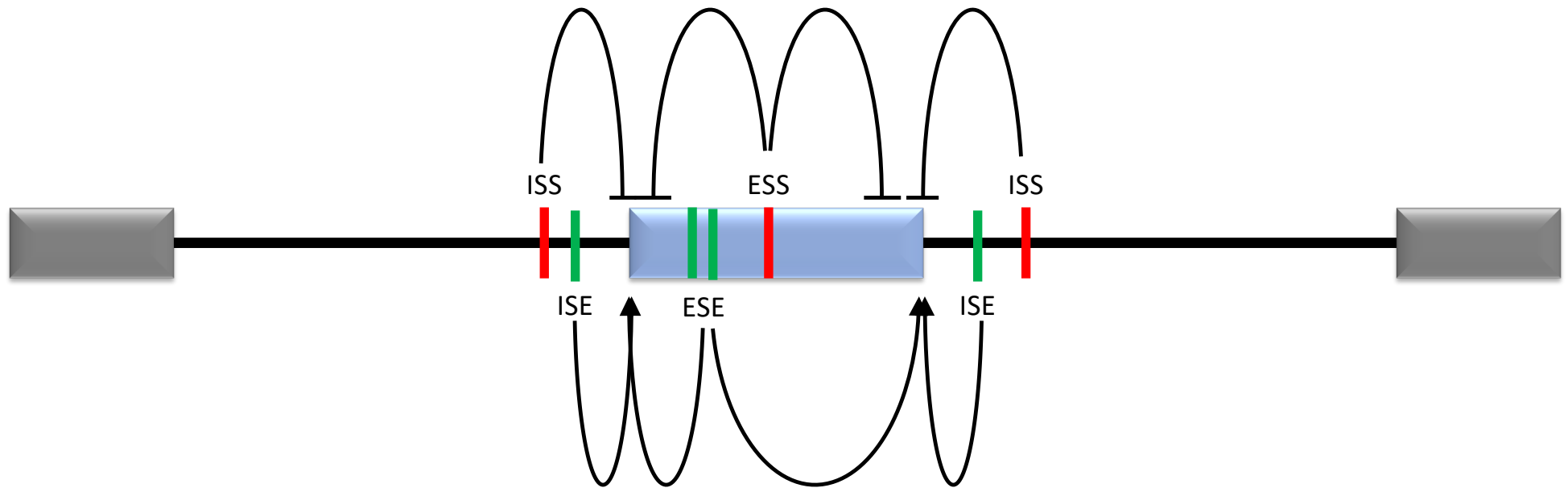
What is alternative splicing?



What is alternative splicing?

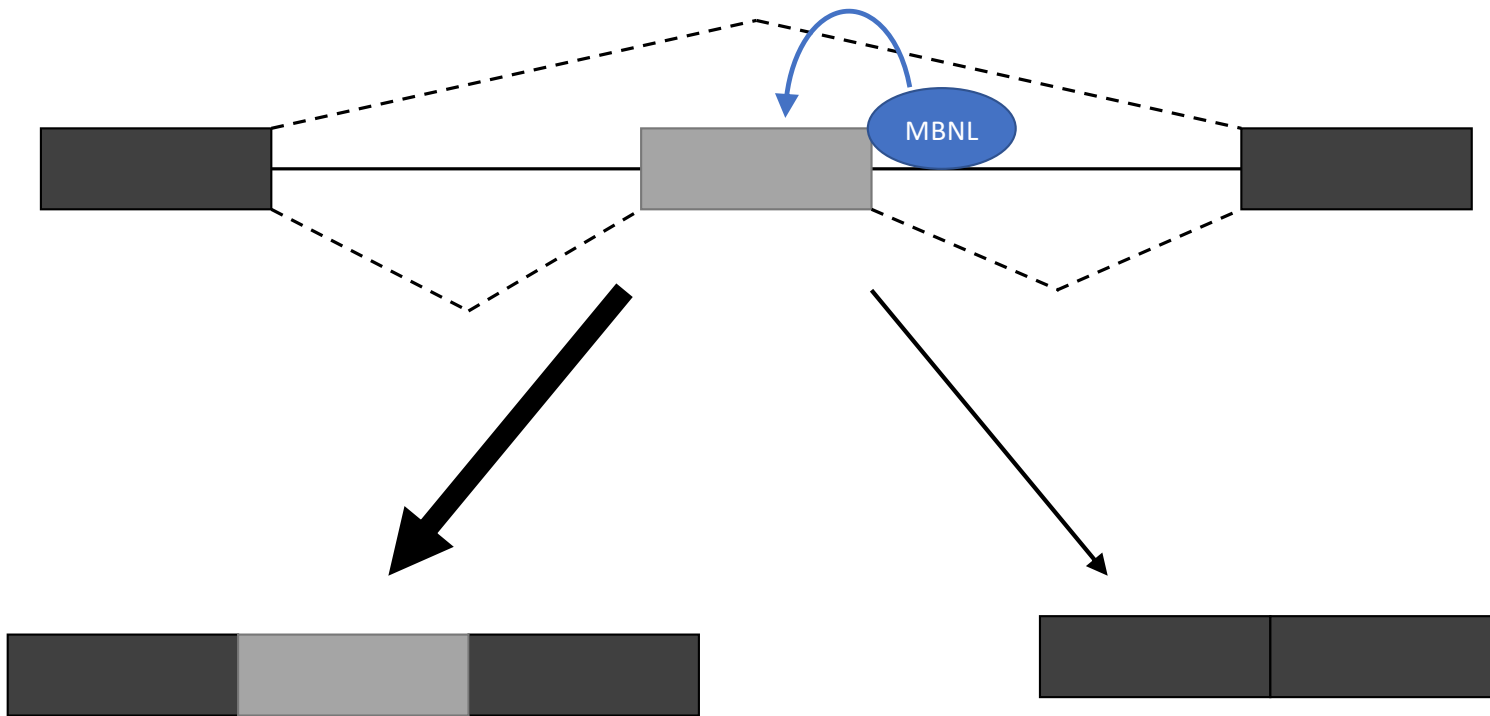


How is the splicing decision made?

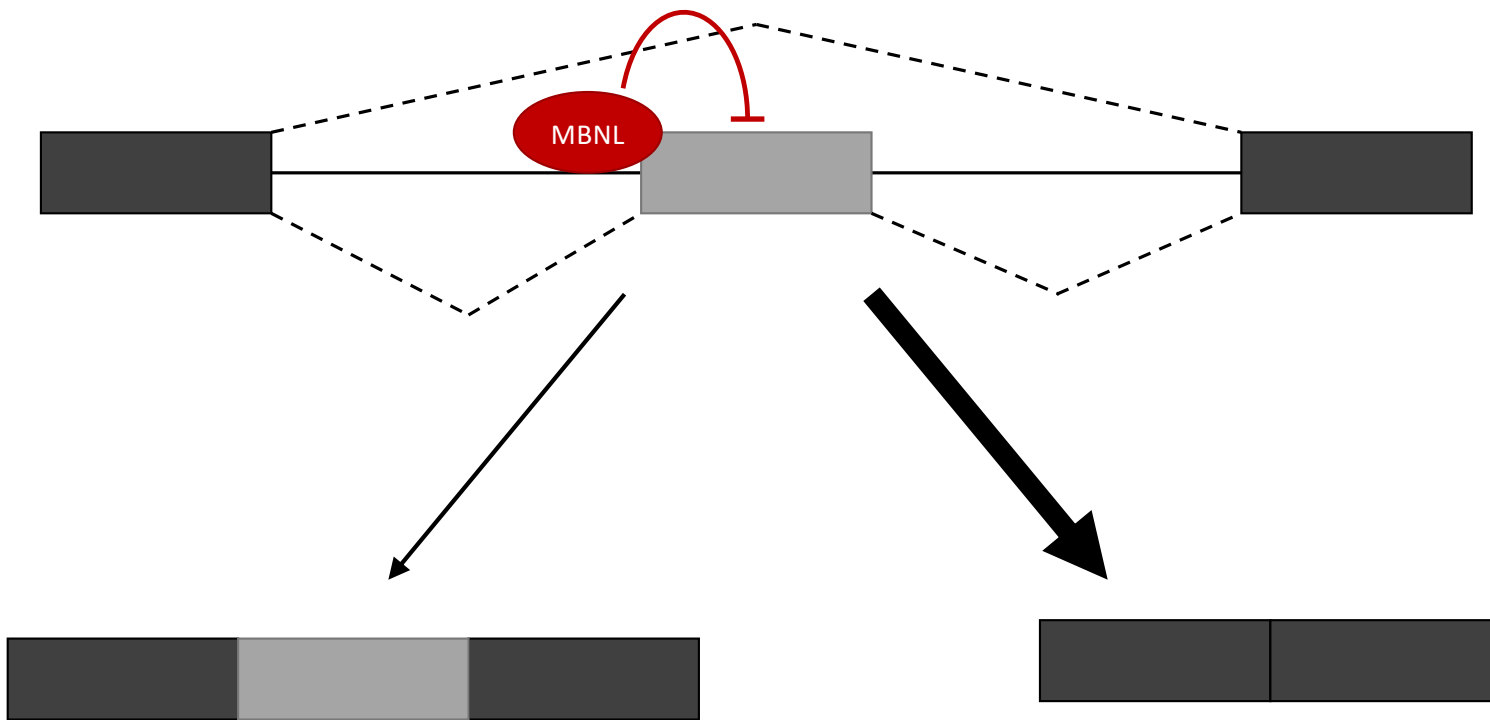


- ISS – intron splicing silencer
- ISE – intron splicing enhancer
- ESS – exon splicing silencer
- ESE – exon splicing enhancer

MBNL is a splicing factor that is both an **activator** and **suppressor** of exon inclusion for pre-mRNA splicing



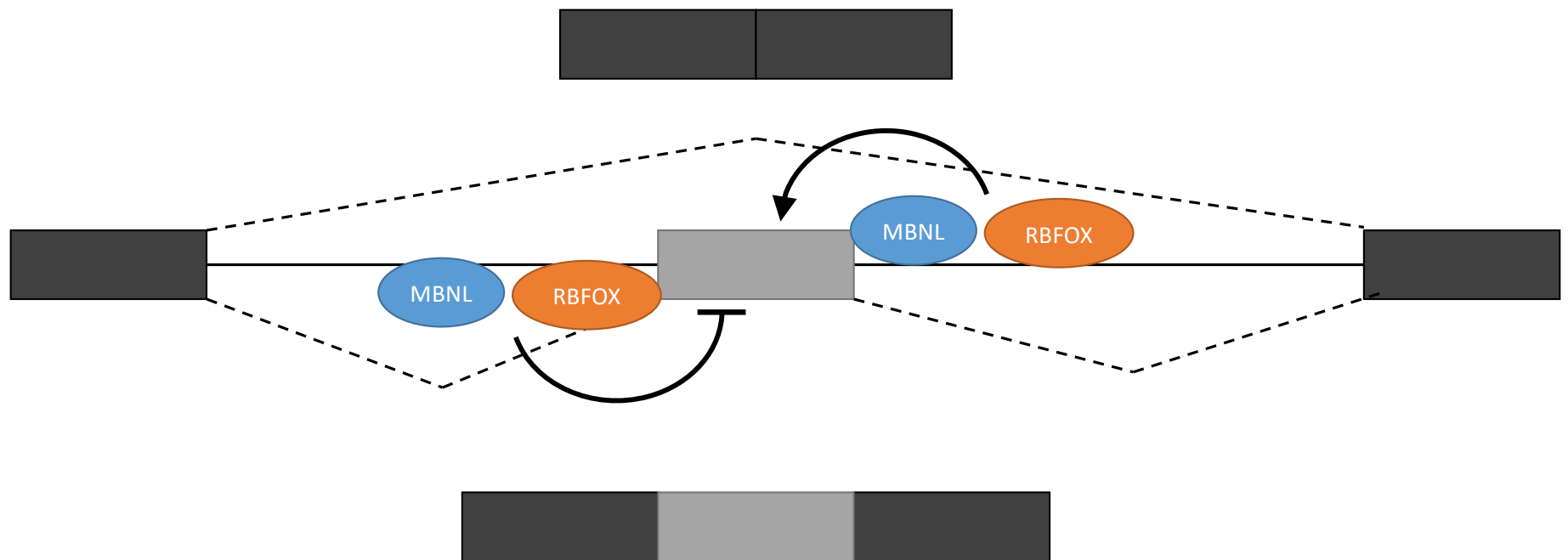
MBNL is a splicing factor that is both an **activator** and **suppressor** of exon inclusion for pre-mRNA splicing



Observations

- MBNL1 and RBFOX1 are both regulators of alternative splicing
- MBNL1 and RBFOX1 regulate splicing in similar manners

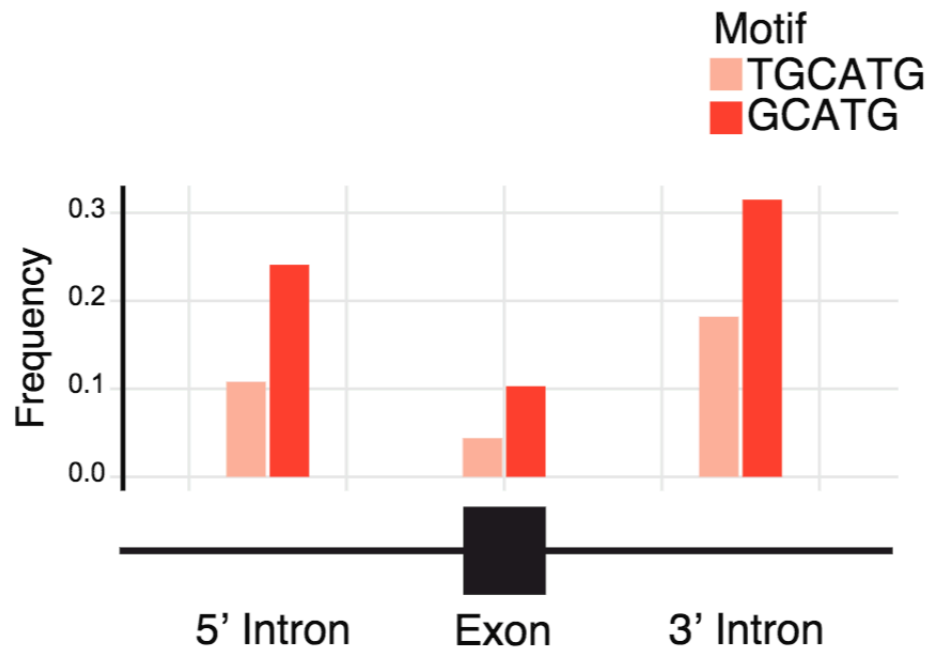
RBFOX1 regulates alternative splicing in a similar manner to MBNL1



Observations

- MBNL1 and RBFOX1 are both regulators of alternative splicing
- MBNL1 and RBFOX1 regulate splicing in similar manners
- MBNL1 and RBFOX1 have similar expression profiles
 - Both expressed in brain, heart, and skeletal muscle
 - Increased expression leads to alternative splicing changes critical for development
 - Depletions in expression have been associated with neurological disorders
- Some MBNL1 regulated splicing events have RBFOX1 binding sites

Analysis of 203 MBNL-regulated exons in mouse revealed the presence of RBFOX binding motifs



66% contain a GCAUG and
33% contain a UGCAUG
RBFOX motif in the exon or
within 250 nucleotides for
the flanking exons

Research question

- How do MBNL1 and RBFOX1 work together to regulate splicing events in transcripts that are regulated by both?

Our goal

- Develop a Python script to plot protein binding motifs on an image of the an exon and flanking introns

```
>INSR chr19:7149896-7151209 (reverse complement)
```

MBNL binding site: YGCT

[illegible]

```
>INSR chr19:7149896-7151209 (reverse complement)
```

MBNL binding site: YGCT

[illegible]

Activity

- Working in groups, discuss general strategy for the algorithm
 - What should the input look like?
 - What should the output look like?
 - What functions are you going to write?
 - How can you handle multiple motifs and multiple genes?
 - How are you going to identify motifs with ambiguity, i.e. YCGY?

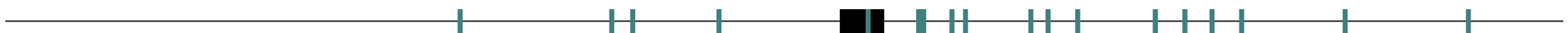
Input

- FASTA file
 - How denote introns vs exons?
- Coordinates
 - UCSC genome browser
 - Need assembly info
 - How interface with UCSC?
- Other?

Output

- Single vector-based image
- To scale!
- Introns vs exons
- Denote multiple motifs
- Key?





```
>INSR chr19:7149896-7151209 (reverse complement)
```

[illegible]

Functions

- Parse FASTA
- Parse file with motifs
- Drawing function?

Activity

- Working in groups, discuss general strategy for the algorithm
 - What should the input look like?
 - What should the output look like?
 - What functions are you going to write?
 - How can you handle multiple motifs and multiple genes?
 - How are you going to identify motifs with “flexibility”, i.e. YCGY?

Drawing with



```
conda install -c conda-forge pycairo
```

```
import cairo
surface = cairo.SVGSurface("plot.svg", width, height)
context = cairo.Context(surface)
context.set_line_width(1)
context.move_to(50,25)
context.line_to(intron1+exon+intron2,25)
context.stroke()
surface.finish()
```

<https://pycairo.readthedocs.io>

<https://cairographics.org/documentation/pycairo/3/>

Your assignment: Motif Mark

- Create a repository in your GitHub profile called motif-mark
- Write a python script to visualize motifs on sequences
 - Minimum requirements:
 - Python3 compatible code
 - Use argparse
 - Input FASTA file and motifs file
 - Output single figure
 - Multiple sequences
 - Multiple motifs
 - Ambiguous motif handling
 - svg output
 - Key/labeling
 - Stretch:
 - Alternate inputs (coordinates)
 - Option of output
 - Offers user customizable colors
 - Color palate options
 - Write object-oriented code
- Turn in your code and a figure for the data Leslie will provide on Canvas