SESSION 5. HUMAN DISEASE

Iron imbalance and the iron-binding responsive element

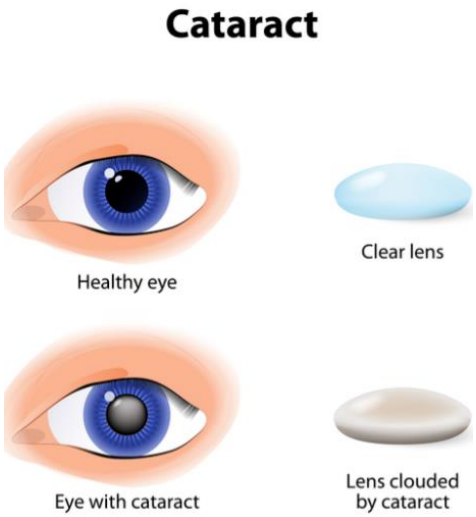
An inherited disease affecting the iron-binding protein function

Hyperferritinaemia cataract syndrom

- → High levels of the protein ferritin in the blood
- → Cataracts: clouding of the lens of the eye.

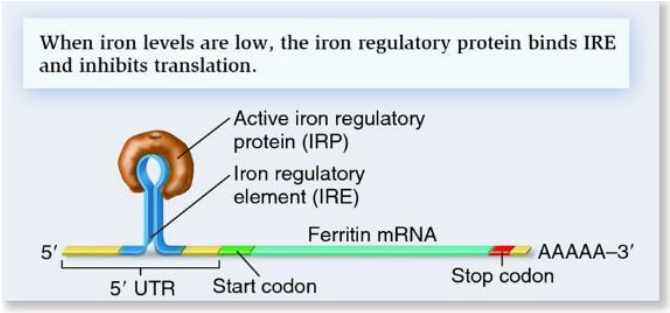
The molecular basis of the disease is a mutation in the ferritin gene.

Mutations on 5'UTRs but not on coding genes → regulatory element on 5'UTR

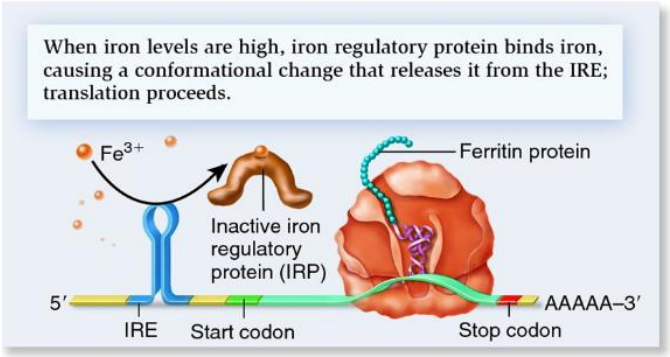


Ferritin is regulated at the level of translation

- **Function of ferritin – free irons are toxic to cells and it has a protective function by binding the irons.**
- **The production of ferritin is carefully regulated in order to maintain a suitable level of irons.**



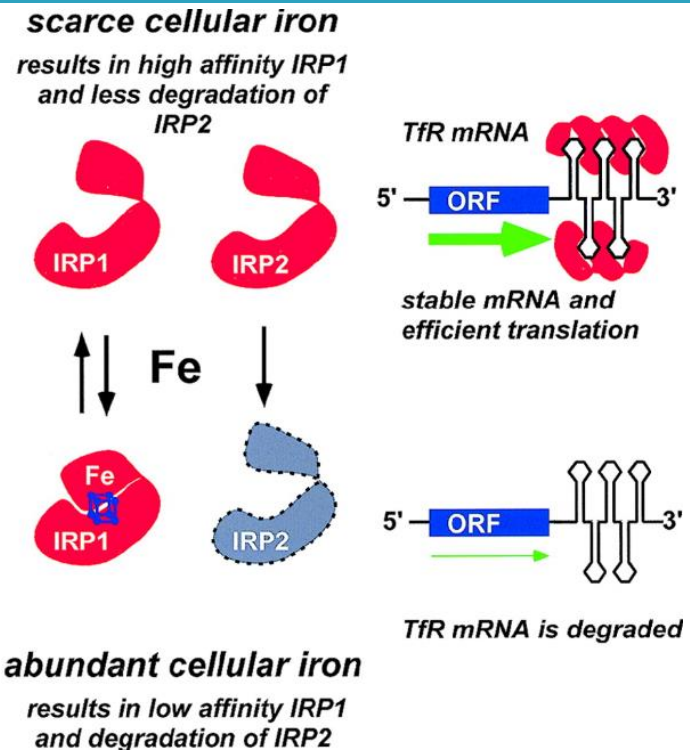
(a) Low iron levels



(b) High iron levels

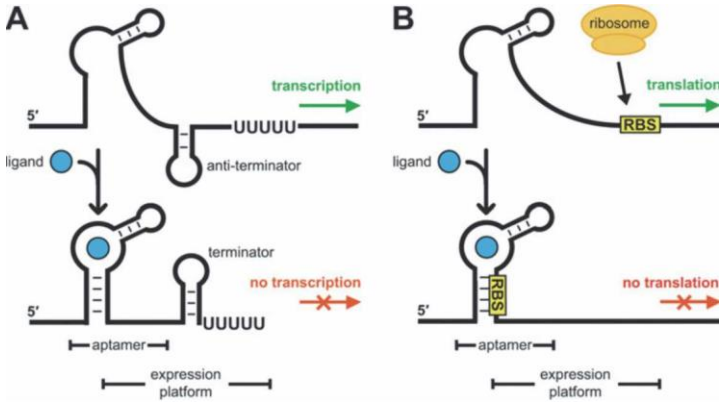
Transferrin

- Function of transferrin is to deliver irons to a cell
- When the irons are abundant, the Tfr mRNAs are suppressed.



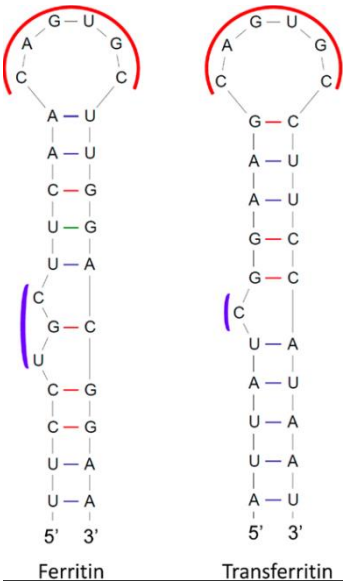
RNA regulatory elements

- IRE
- Riboswitches
- Transcription stop by hairpin structure: replication-dependent histones.
- ~1000 RNA-Binding Proteins (RBPs) recognize specific RNA regulatory elements



RNA regulatory elements

- IRE
- Riboswitches
- Transcription stop by hairpin structure: replication-dependent histones.

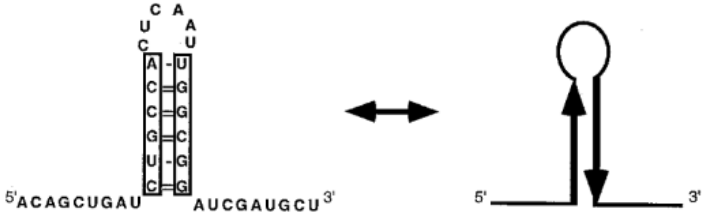


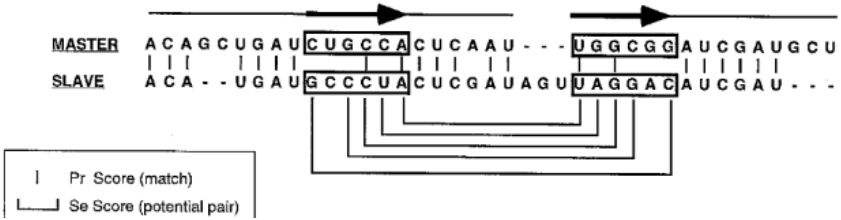
Identifying the iron responsive element

- IRE are also presented in other mRNAs, most of them are related to iron metabolism
- A computational method to identify all IREs over all mRNA sequences.
- This may allow to discover new additional RNAs, regulated in the same way as ferritin and Tfr.

RNA secondary structure

- Alignment-based method (Dynamic programming)





RNA secondary structure

- Energy minimization method
  - A structure with the minimum free energy (kcal/mol)
  - Energy minimization using nearest-neighbor model of nucleotide stacking
  - Nearest-neighbor model: The interaction between bases on different strands depends somewhat on the neighboring bases.

Table 1. Nearest-neighbor parameters for DNA/DNA duplexes in 1 M NaCl.[8]

Nearest-neighbor sequence (5'-3'/3'-5')	ΔH° kJ/mol	ΔS° J/(mol·K)	ΔG°_{37} kJ/mol
AA/TT	−33.1	−92.9	−4.26
AT/TA	−30.1	−85.4	−3.67
TA/AT	−30.1	−89.1	−2.50
CA/GT	−35.6	−95.0	−6.12
GT/CA	−35.1	−93.7	−6.09
CT/GA	−32.6	−87.9	−5.40
GA/CT	−34.3	−92.9	−5.51
CG/GC	−44.4	−113.8	−9.07
GC/CG	−41.0	−102.1	−9.36
GG/CC	−33.5	−83.3	−7.66
Terminal A-T base pair	9.6	17.2	4.31
Terminal G-C base pair	0.4	−11.7	4.05

```
    ↓ ↓ ↓ ↓ ↓
5' C-G-T-T-G-A 3'
3' G-C-A-A-C-T 5'
```

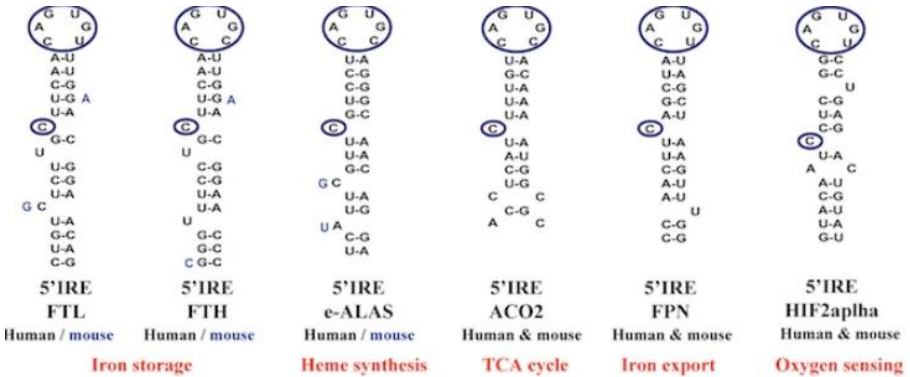
The free energy of forming this DNA from the individual strands, ΔG°, is represented (at 37 °C) as

ΔG°_{37}(predicted) = ΔG°_{37}(CG initiation) + ΔG°_{37}(CG/GC) + ΔG°_{37}(GT/CA) + ΔG°_{37}(TT/AA) + ΔG°_{37}(TG/AC) + ΔG°_{37}(GA/CT) + ΔG°_{37}(AT initiation)

4.05 + (-9.07) + (-6.09) + (-4.26) + (-6.12) + (-5.51) + 4.31 (kJ/mol)

Identifying the iron responsive element

- CAGUGN loop sequence
- The conserved base-pairing pattern



ire.py

```
#!/usr/bin/python


def findstem(strand1, strand2):
    tag = 1
    for j in range(0, 5):
        base1 = strand1[j]
        base2 = strand2[4 - j]
        if not pair(base1, base2):
            tag = 0
    if tag == 1:
        return 1


def pair(base1, base2):
    if    base1 == 'G' and base2 == 'C' \
       or base1 == 'G' and base2 == 'U' \
       or base1 == 'A' and base2 == 'U' \
       or base1 == 'C' and base2 == 'G' \
       or base1 == 'U' and base2 == 'A' \
       or base1 == 'U' and base2 == 'G':
        return 1


seq = 'GAGAGCAGUGGGGGUUUCCUGCUUCAACAGUGCUUGGACGGAACCCGGCGCUCGUUCCCCA'

for i in range(0, len(seq) - 16):

    test = seq[i:i + 16]
    if test[5:10] == 'CAGUG':
        strand1 = test[0:5]
        strand2 = test[11:16]
        if findstem(strand1, strand2):
            pos = i + 1
            print 'match at position', pos, ':'
            print test
            print '<----CAGUGN---->'
```

String comparison

```
if 'a'=='a': print 'True'
else: print 'False'

if 'a'=='A': print 'True'
else: print 'False'
```

```
[jwnam@biglab-master Session5]$ python stringcomp.py
True
False
```

ire.py

```
#!/usr/bin/python


def findstem(strand1, strand2):
    tag = 1
    for j in range(0, 5):
        base1 = strand1[j]
        base2 = strand2[4 - j]
        if not pair(base1, base2):
            tag = 0
    if tag == 1:
        return 1


def pair(base1, base2):
    if    base1 == 'G' and base2 == 'C' \
       or base1 == 'G' and base2 == 'U' \
       or base1 == 'A' and base2 == 'U' \
       or base1 == 'C' and base2 == 'G' \
       or base1 == 'U' and base2 == 'A' \
       or base1 == 'U' and base2 == 'G':
        return 1


seq = 'GAGAGCAGUGGGGGUUUCCUGCUUCAACAGUGCUUGGACGGAACCCGGCGCUCGUUCCCCA'

for i in range(0, len(seq) - 16):

    test = seq[i:i + 16]
    if test[5:10] == 'CAGUG':
        strand1 = test[0:5]
        strand2 = test[11:16]
        if findstem(strand1, strand2):
            pos = i + 1
            print 'match at position', pos, ':'
            print test
            print '<----CAGUGN---->'
```

```
[jwnam@biglab-master Session5]$ python ire.py
match at position 23 :
UUCAACAGUGCUUGGA
<----CAGUGN---->
```

ire2.py

```
#!/usr/bin/python

def findstem(strand1, strand2):
    leftPar = ''; rightPar=''; pairNum=0
    for j in range(0, 5):
        base1 = strand1[j]
        base2 = strand2[4 - j]
        if not pair(base1, base2):
            leftPar+='.'; rightPar='.'+rightPar
        else: leftPar+='('; rightPar=')'+rightPar; pairNum+=1
    return leftPar, rightPar, pairNum


def pair(base1, base2):
    if    base1 == 'G' and base2 == 'C' \
       or base1 == 'G' and base2 == 'U' \
       or base1 == 'A' and base2 == 'U' \
       or base1 == 'C' and base2 == 'G' \
       or base1 == 'U' and base2 == 'A' \
       or base1 == 'U' and base2 == 'G':
        return 1


seq = 'GAGAGCAGUGGGGGUUUCCUGCUUCAACAGUGCAUGGACGGAACCCGGCGCUCGUUCCCCA'

for i in range(0, len(seq) - 16):

    test = seq[i:i + 16]
    if test[5:10] == 'CAGUG':
        strand1 = test[0:5]
        strand2 = test[11:16]
        leftPar, rightPar, pairNum = findstem(strand1, strand2)
        if pairNum>=4:
            pos = i + 1
            print 'match at position', pos, ':'
            print test
            print leftPar+'CAGUGN'+rightPar
```

```
[jwnam@biglab-master Session5]$ python ire2.py
match at position 23 :
UUCAACAGUGCAUGGA
(((((.CAGUGN.)))))
```