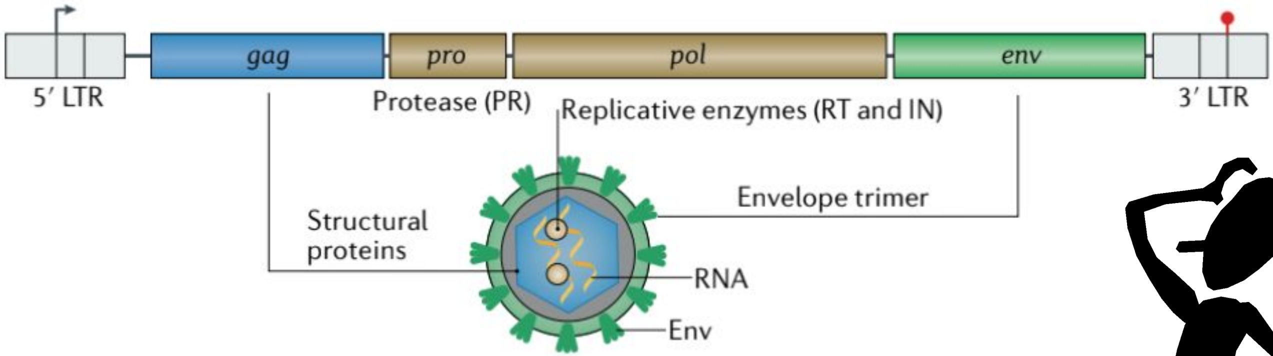
# Comp 555 - BioAlgorithms - Spring 2021



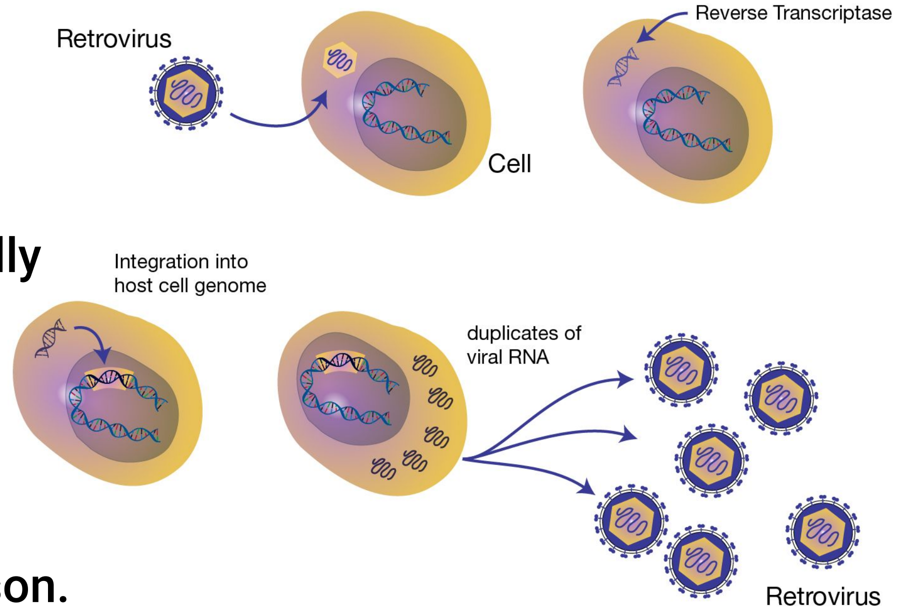
Looking for Fossils

# Endogenous Retroviruses (ERV)

During evolution various Retroviruses have incorporated themselves permanently into vertebrate genomes.

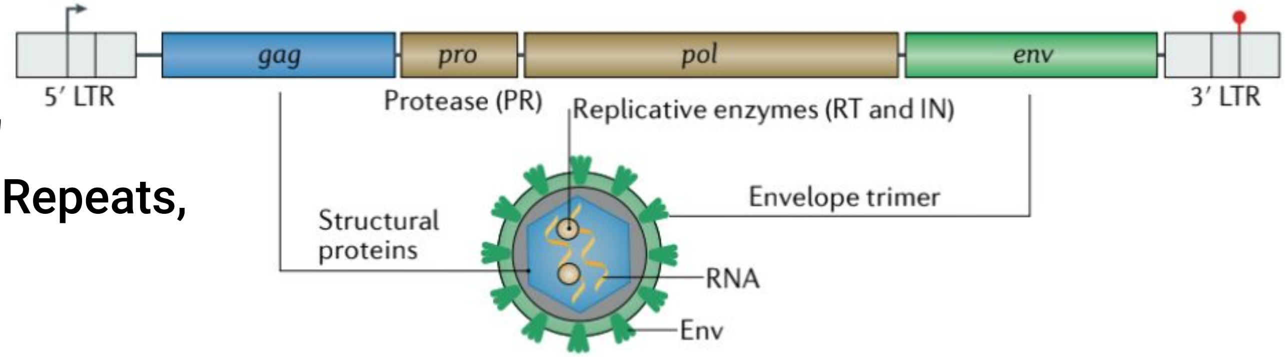
These "Endogenous" Retroviruses are generally dormant, but they occasionally awaken and, rather than leave the cell, they incorporate new copies of themselves back into the host DNA.

Thus, they are a form of Retrotransposon.

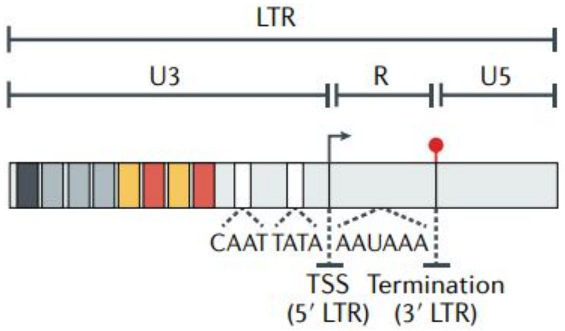


# ERV genome structure

The ERV genome is flanked by two "Identical sequences" called Long Terminal Repeats, LTRs.



These LTRs contain the transcription start and end sites that are used when the ERV is copied (retrotransposed). These are parentheses enclosing the "proviral" sequence.

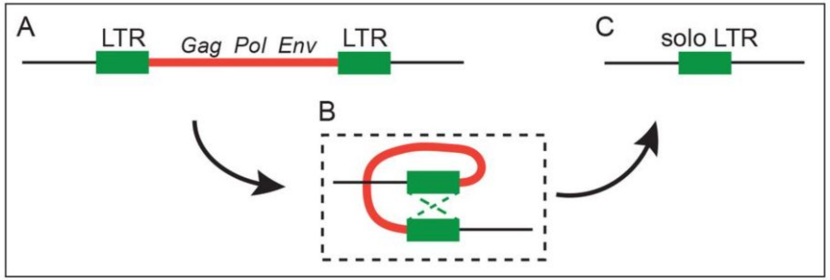


LTRs are required for the ERV to activate.

# LTRs can lose their virus

Active ERVs are bad news.

They tend to insert themselves into genes, and generally reduce the fitness of their host.



One way of cleaning the genome of ERVs is to remove their viral sequence.

The genome has a natural way of doing this, through a process call recombination (the same mechanism that exchanges sequences between the two chromosome copies).

The "identical" sequences of the two LTRs are their Achilles Heel. Sometimes, they pair up and recombine, and as a side-effect the viral sequence is excised.

The genome is full of these vestigial LTRs.

# Let's look for them

There are several ways that we could proceed.

1. We could start by looking at all those 45-mers that are over-represented in the genome. But, not all of these sequences are ERV LTRs
2. We could start with a viral template. Where do we get one?

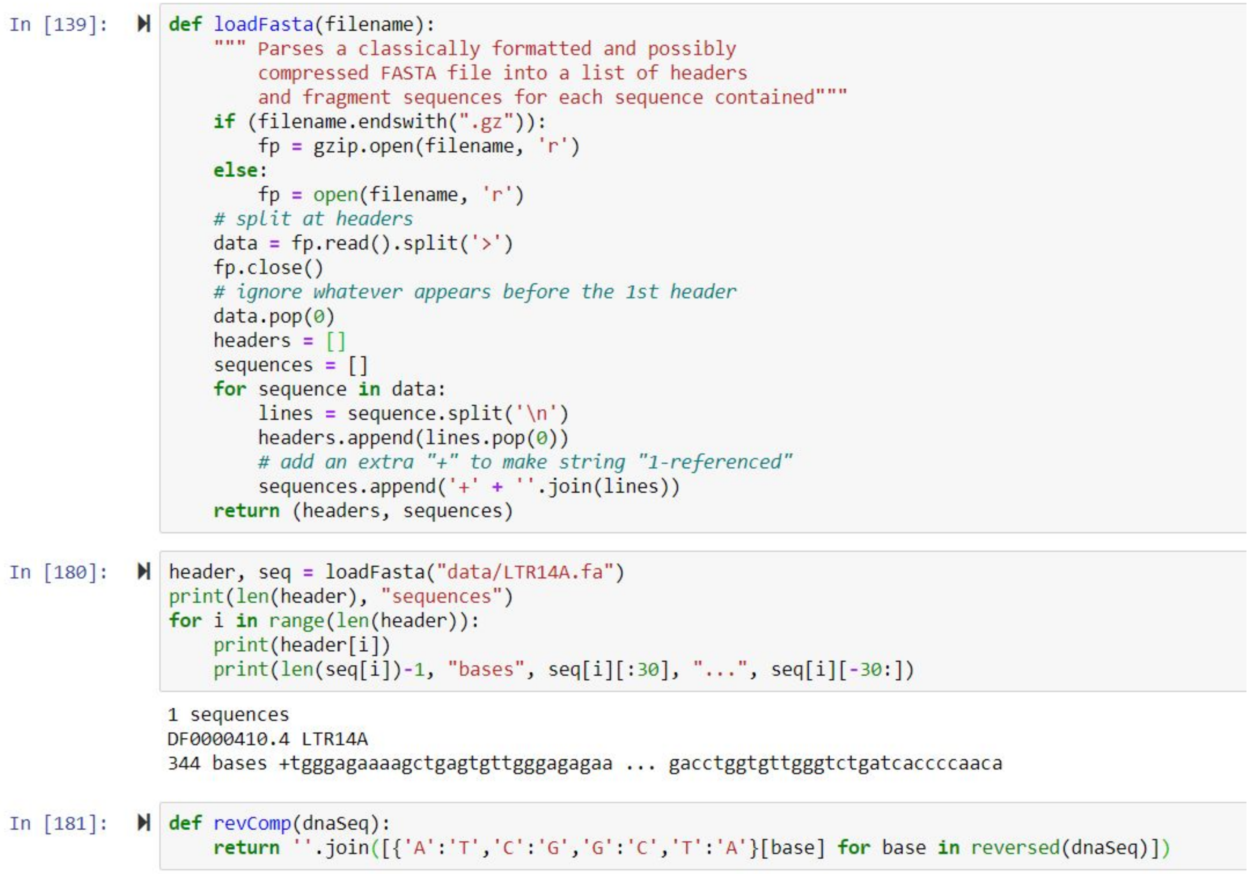
Luckily biologists have used the first method to give us templates that we can use for the second.

There are databases with these "approximate" sequences.

Dfam ISB RepeatMasker

## Getting started

**Same old code...**



```
In [139]:  def loadFasta(filename):
                """ Parses a classically formatted and possibly
                    compressed FASTA file into a list of headers
                    and fragment sequences for each sequence contained"""
                if (filename.endswith(".gz")):
                    fp = gzip.open(filename, 'r')
                else:
                    fp = open(filename, 'r')
                # split at headers
                data = fp.read().split('>')
                fp.close()
                # ignore whatever appears before the 1st header
                data.pop(0)
                headers = []
                sequences = []
                for sequence in data:
                    lines = sequence.split('\n')
                    headers.append(lines.pop(0))
                    # add an extra "+" to make string "1-referenced"
                    sequences.append('+' + ''.join(lines))
                return (headers, sequences)
```

```
In [180]:  header, seq = loadFasta("data/LTR14A.fa")
            print(len(header), "sequences")
            for i in range(len(header)):
                print(header[i])
                print(len(seq[i])-1, "bases", seq[i][:30], "...", seq[i][-30:])
```

```
1 sequences
DF0000410.4 LTR14A
344 bases +tgggagaaaagctgagtgttgggagagaa ... gacctggtgttgggtctgatcaccccaaca
```

```
In [181]:  def revComp(dnaSeq):
                return ''.join([{'A':'T','C':'G','G':'C','T':'A'}[base] for base in reversed(dnaSeq)])
```

# New stuff

## "Signature" k-mers

```
In [185]:  ltr = seq[0].upper()
           K = 19
           forward = dict([(ltr[i:i+K], i) for i in range(1,len(ltr)-K+1)])
           print(len(forward))
           rev = "+" + revComp(ltr[1:])
           reverse = dict([(rev[i:i+K], -i) for i in range(1,len(rev)-K+1)])
           print(len(reverse))

           for key in forward:
               if key in reverse:
                   print(key)
```

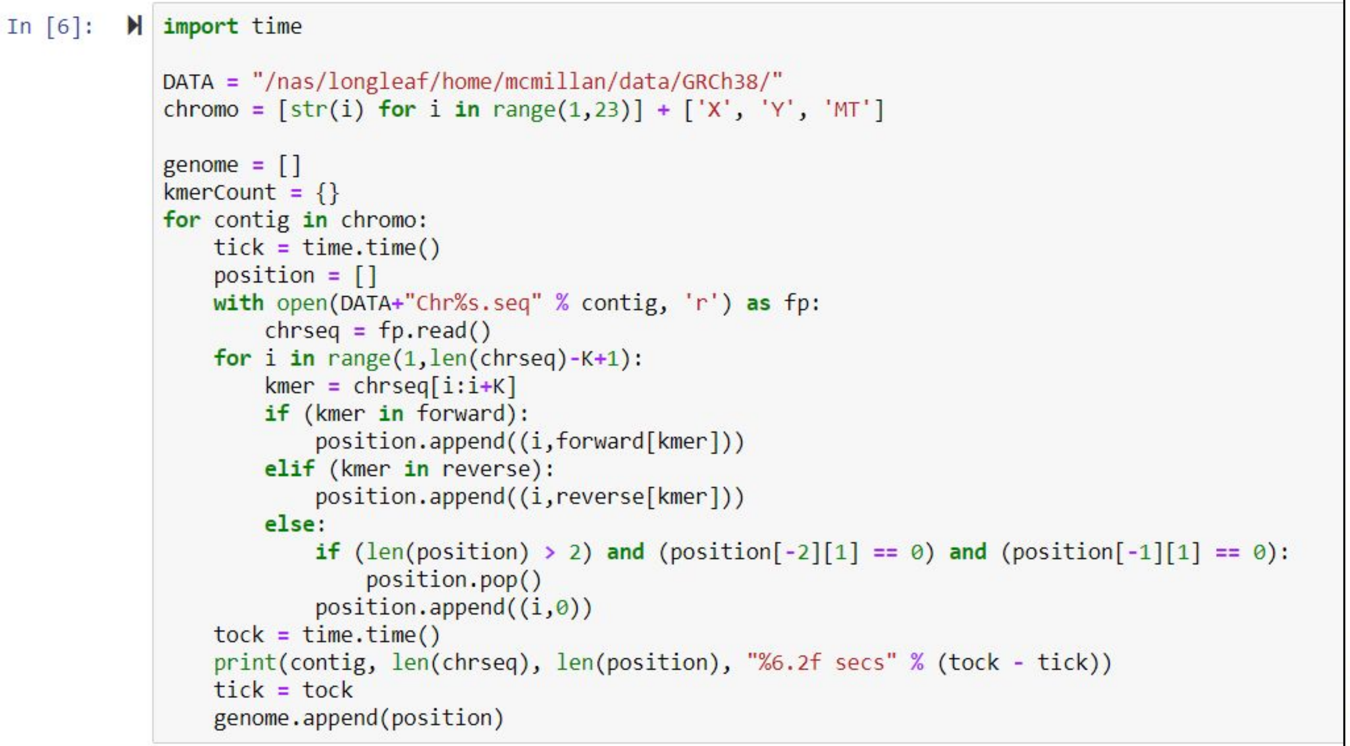
```
326
326
```

```
In [186]:  print(ltr)
```

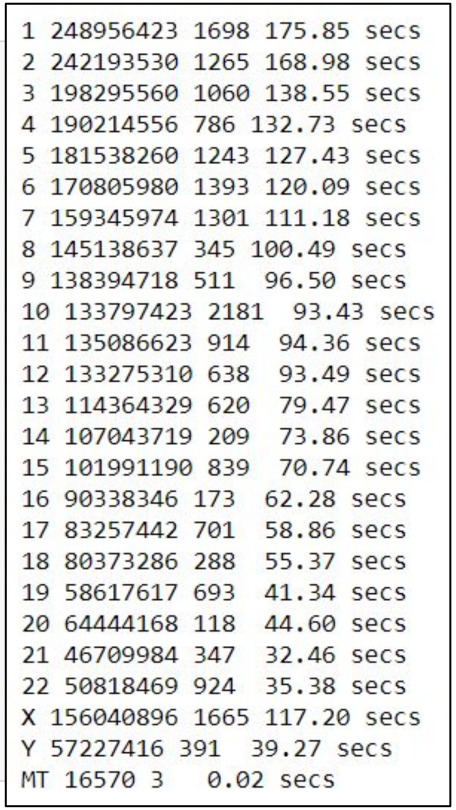
```
+TGGGAGAAAAGCTGAGTGTTGGGAGAGAAGCTGAGGCAGGGCTTGCATGTCTGCTAGACTTGCTGGCTCCTTGCTTCTAGCACTCCCATTATCTCAAGCAGCCATATGTTTCTCATTCACTTG
ATACACCGTTTCCTTTCAACCCCCACATCCTCACCACCTGTTTCTTTGTTTGAGCACCAATAAATAGCGTGGGCTCCCAGAGCTCGGGGCCTTCGCAGCCTCCACACTCGCGATGGCCCCCTGG
TCCCACTTTCTCTCTCAAACTGTCTTTTTCTCATTCCTTTGACTCCGCCGGACTTCGTCGCCCCCACGACCTGGTGTTGGGTCTGATCACCCCAACA
```

## Let's go fishing

### Let's scan the genome looking for LTRs...

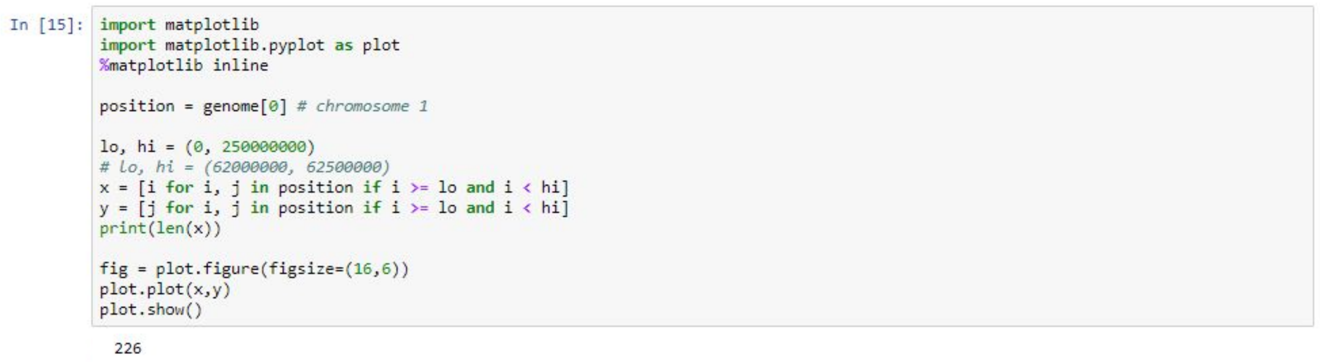


```
In [6]:  import time

         DATA = "/nas/longleaf/home/mcmillan/data/GRCh38/"
         chromo = [str(i) for i in range(1,23)] + ['X', 'Y', 'MT']

         genome = []
         kmerCount = {}
         for contig in chromo:
             tick = time.time()
             position = []
             with open(DATA+"Chr%s.seq" % contig, 'r') as fp:
                 chrseq = fp.read()
             for i in range(1,len(chrseq)-K+1):
                 kmer = chrseq[i:i+K]
                 if (kmer in forward):
                     position.append((i,forward[kmer]))
                 elif (kmer in reverse):
                     position.append((i,reverse[kmer]))
                 else:
                     if (len(position) > 2) and (position[-2][1] == 0) and (position[-1][1] == 0):
                         position.pop()
                     position.append((i,0))
             tock = time.time()
             print(contig, len(chrseq), len(position), "%6.2f secs" % (tock - tick))
             tick = tock
             genome.append(position)
```

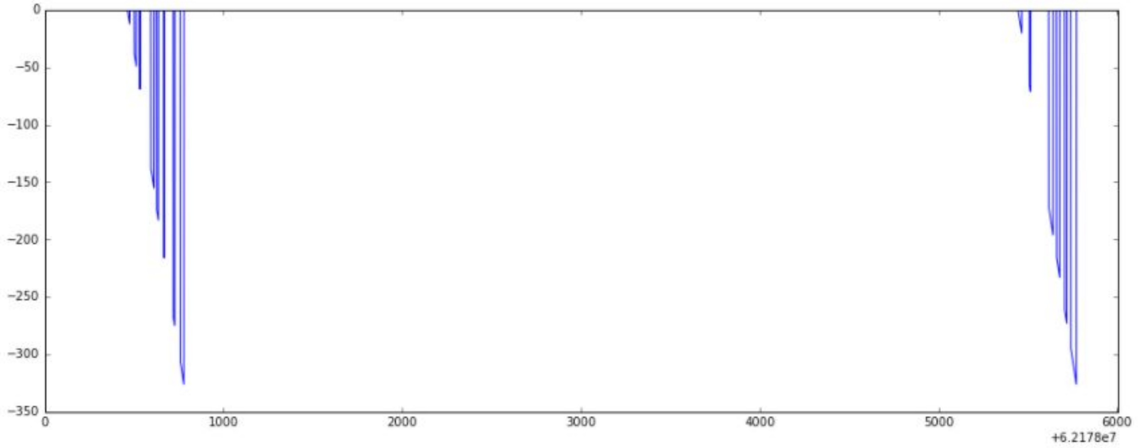


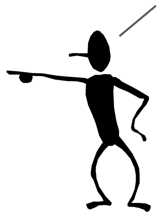
```
1 248956423 1698 175.85 secs
2 242193530 1265 168.98 secs
3 198295560 1060 138.55 secs
4 190214556 786 132.73 secs
5 181538260 1243 127.43 secs
6 170805980 1393 120.09 secs
7 159345974 1301 111.18 secs
8 145138637 345 100.49 secs
9 138394718 511  96.50 secs
10 133797423 2181  93.43 secs
11 135086623 914  94.36 secs
12 133275310 638  93.49 secs
13 114364329 620  79.47 secs
14 107043719 209  73.86 secs
15 101991190 839  70.74 secs
16 90338346 173  62.28 secs
17 83257442 701  58.86 secs
18 80373286 288  55.37 secs
19 58617617 693  41.34 secs
20 64444168 118  44.60 secs
21 46709984 347  32.46 secs
22 50818469 924  35.38 secs
X 156040896 1665 117.20 secs
Y 57227416 391  39.27 secs
MT 16570 3   0.02 secs
```

# Let's take a look



```
In [15]: import matplotlib
         import matplotlib.pyplot as plot
         %matplotlib inline

         position = genome[0] # chromosome 1

         lo, hi = (0, 250000000)
         # lo, hi = (62000000, 62500000)
         x = [i for i, j in position if i >= lo and i < hi]
         y = [j for i, j in position if i >= lo and i < hi]
         print(len(x))

         fig = plot.figure(figsize=(16,6))
         plot.plot(x,y)
         plot.show()

226
```



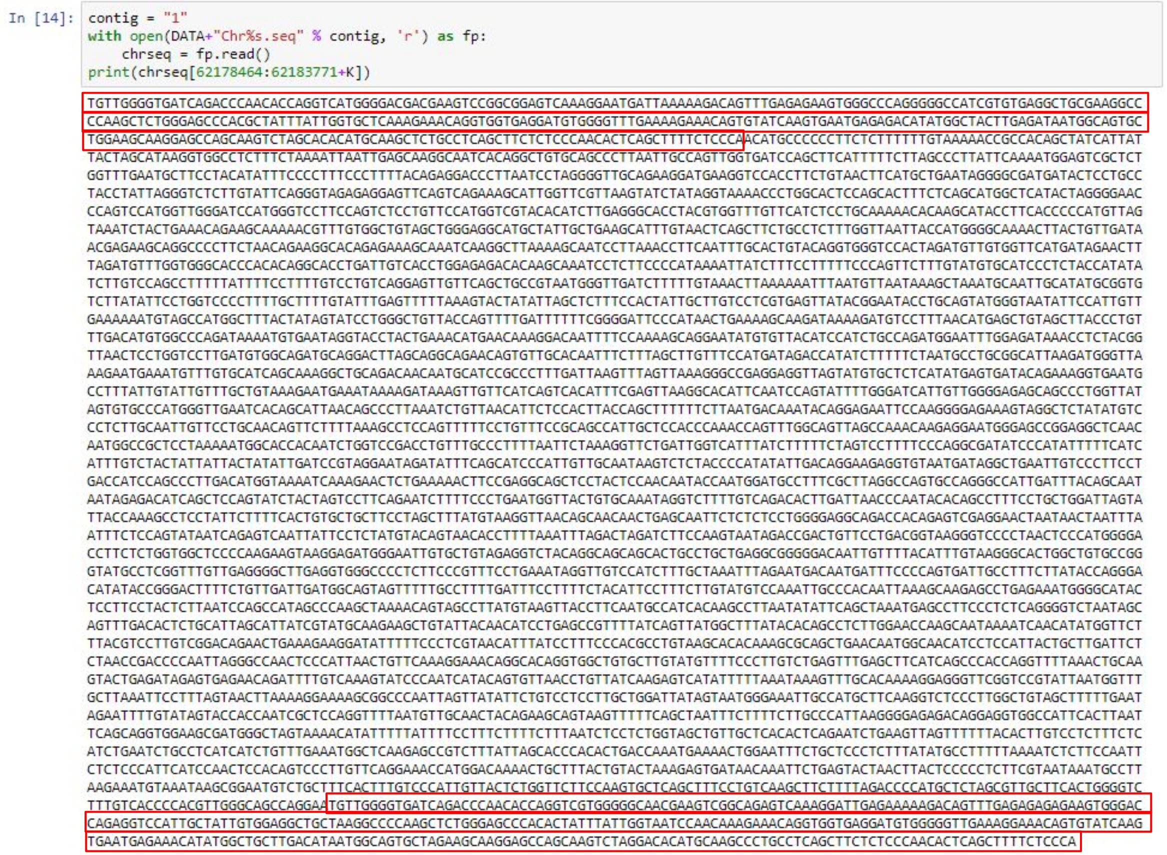


These are potentially LTRs on chromosome 1.

Note, many have accumulated mutations, in fact, none are an exact match for our template.

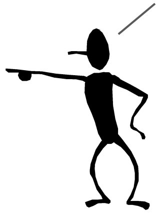
The most likely LTRs are those that share many k-mers in the expected order.

# Taking a look



```
In [14]: contig = "1"
         with open(DATA+"Chr%s.seq" % contig, 'r') as fp:
             chrseq = fp.read()
         print(chrseq[62178464:62183771+K])
```

```
TGTTGGGGTGATCAGACCCAACACCAGGTCATGGGGACGACGAAGTCCGGCGGAGTCAAAGGAATGATTAAAAAGACAGTTTGAGAGAAGTGGGCCCAGGGGGCCATCGTGTGAGGCTGCGAAGGCC
CCAAGCTCTGGGAGCCCACGCTATTTATTGGTGCTCAAAGAAACAGGTGGTGAGGATGTGGGGTTTGAAAAGAAACAGTGTATCAAGTGAATGAGAGACATATGGCTACTTGAGATAATGGCAGTGC
TGGAAGCAAGGAGCCAGCAAGTCTAGCACACATGCAAGCTCTGCCTCAGCTTCTCTCCCAACACTCAGCTTTTCTCCCAACATGCCCCCCTTCTCTTTTTTGTAAAAACCGCCACAGCTATCATTAT
TACTAGCATAAGGTGGCCTCTTTCTAAAATTAATTGAGCAAGGCAATCACAGGCTGTGCAGCCCTTAATTGCCAGTTGGTGATCCAGCTTCATTTTTCTTAGCCCTTATTCAAAATGGAGTCGCTCT
GGTTTGAATGCTTCCTACATATTTCCCCTTTCCCTTTTACAGAGGACCCTTAATCCTAGGGGTTGCAGAAGGATGAAGGTCCACCTTCTGTAACTTCATGCTGAATAGGGGCGATGATACTCCTGCC
TACCTATTAGGGTCTCTTGTATTCAGGGTAGAGAGGAGTTCAGTCAGAAAGCATTGGTTCGTTAAGTATCTATAGGTAAAACCCTGGCACTCCAGCACTTTCTCAGCATGGCTCATACTAGGGGAAC
CCAGTCCATGGTTGGGATCCATGGGTCCTTCCAGTCTCCTGTTCCATGGTCGTACACATCTTGAGGGCACCTACGTGGTTTGTTCATCTCCTGCAAAAACACAAGCATACCTTCACCCCCATGTTAG
TAAATCTACTGAAACAGAAGCAAAAACGTTTGTGGCTGTAGCTGGGAGGCATGCTATTGCTGAAGCATTTGTAACTCAGCTTCTGCCTCTTTGGTTAATTACCATGGGGCAAAACTTACTGTTGATA
ACGAGAAGCAGGCCCCTTCTAACAGAAGGCACAGAGAAAGCAAATCAAGGCTTAAAAGCAATCCTTAAACCTTCAATTTGCACTGTACAGGTGGGTCCACTAGATGTTGTGGTTCATGATAGAACTT
TAGATGTTTGGTGGGCACCCACACAGGCACCTGATTGTCACCTGGAGAGACACAAGCAAATCCTCTTCCCCATAAAATTATCTTTCCTTTTTCCCAGTTCTTTGTATGTGCATCCCTCTACCATATA
TCTTGTCCAGCCTTTTTATTTTCCTTTTGTCCTGTCAGGAGTTGTTCAGCTGCCGTAATGGGTTGATCTTTTTGTAAACTTAAAAAATTTAATGTTAATAAAGCTAAATGCAATTGCATATGCGGTG
TCTTATATTCCTGGTCCCCTTTTGCTTTTGTATTTGAGTTTTTAAAGTACTATATTAGCTCTTTCCACTATTGCTTGTCCTCGTGAGTTATACGGAATACCTGCAGTATGGGTAATATTCCATTGTT
GAAAAAATGTAGCCATGGCTTTACTATAGTATCCTGGGCTGTTACCAGTTTTGATTTTTTCGGGGATTCCCATAACTGAAAAGCAAGATAAAAGATGTCCTTTAACATGAGCTGTAGCTTACCCTGT
TTGACATGTGGCCCAGATAAAATGTGAATAGGTACCTACTGAAACATGAACAAAGGACAATTTTCCAAAAGCAGGAATATGTGTTACATCCATCTGCCAGATGGAATTTGGAGATAAACCTCTACGG
TTAACTCCTGGTCCTTGATGTGGCAGATGCAGGACTTAGCAGGCAGAACAGTGTTGCACAATTTCTTTAGCTTGTTTCCATGATAGACCATATCTTTTTCTAATGCCTGCGGCATTAAGATGGGTTA
AAGAATGAAATGTTTGTGCATCAGCAAAGGCTGCAGACAACAATGCATCCGCCCTTTGATTAAGTTTAGTTAAAGGGCCGAGGAGGTTAGTATGTGCTCTCATATGAGTGATACAGAAAGGTGAATG
CCTTTATTGTATTGTTTGCTGTAAAGAATGAAATAAAAGATAAAGTTGTTCATCAGTCACATTTCGAGTTAAGGCACATTCAATCCAGTATTTTGGGATCATTGTTGGGGAGAGCAGCCCTGGTTAT
AGTGTGCCCATGGGTTGAATCACAGCATTAACAGCCCTTAAATCTGTTAACATTCTCCACTTACCAGCTTTTTTCTTAATGACAAATACAGGAGAATTCCAAGGGGAGAAAGTAGGCTCTATATGTC
CCTCTTGCAATTGTTCCTGCAACAGTTCTTTTAAAGCCTCCAGTTTTTCCTGTTTCCGCAGCCATTGCTCCACCCAAACCAGTTTGGCAGTTAGCCAAACAAGAGGAATGGGAGCCGGAGGCTCAAC
AATGGCCGCTCCTAAAAATGGCACCACAATCTGGTCCGACCTGTTTGCCCTTTTAATTCTAAAGGTTCTGATTGGTCATTTATCTTTTTCTAGTCCTTTTCCCAGGCGATATCCCATATTTTTCATC
ATTTGTCTACTATTATTACTATATTGATCCGTAGGAATAGATATTTCAGCATCCCATTGTTGCAATAAGTCTCTACCCCATATATTGACAGGAAGAGGTGTAATGATAGGCTGAATTGTCCCTTCCT
GACCATCCAGCCCTTGACATGGTAAAATCAAAGAACTCTGAAAAACTTCCGAGGCAGCTCCTACTCCAACAATACCAATGGATGCCTTTCGCTTAGGCCAGTGCCAGGGCCATTGATTTACAGCAAT
AATAGAGACATCAGCTCCAGTATCTACTAGTCCTTCAGAATCTTTTCCCTGAATGGTTACTGTGCAAATAGGTCTTTTGTCAGACACTTGATTAACCCAATACACAGCCTTTCCTGCTGGATTAGTA
TTACCAAAGCCTCCTATTCTTTTCACTGTGCTGCTTCCTAGCTTTATGTAAGGTTAACAGCAACAACTGAGCAATTCTCTCTCCTGGGGAGGCAGACCACAGAGTCGAGGAACTAATAACTAATTTA
ATTTCTCCAGTATAATCAGAGTCAATTATTCCTCTATGTACAGTAACACCTTTTAAATTTAGACTAGATCTTCCAAGTAATAGACCGACTGTTCCTGACGGTAAGGGTCCCCTAACTCCCATGGGGA
CCTTCTCTGGTGGCTCCCCAAGAAGTAAGGAGATGGGAATTGTGCTGTAGAGGTCTACAGGCAGCAGCACTGCCTGCTGAGGCGGGGGACAATTGTTTTACATTTGTAAGGGCACTGGCTGTGCCGG
GTATGCCTCGGTTTGTTGAGGGGCTTGAGGTGGGCCCCTCTTCCCGTTTCCTGAAATAGGTTGTCCATCTTTGCTAAATTTAGAATGACAATGATTTCCCCAGTGATTGCCTTTCTTATACCAGGGA
CATATACCGGGACTTTTCTGTTGATTGATGGCAGTAGTTTTTGCCTTTTGATTTCCTTTTCTACATTCCTTTCTTGTATGTCCAAATTGCCCACAATTAAAGCAAGAGCCTGAGAAATGGGGCATAC
TCCTTCCTACTCTTAATCCAGCCATAGCCCAAGCTAAAACAGTAGCCTTATGTAAGTTACCTTCAATGCCATCACAAGCCTTAATATATTCAGCTAAATGAGCCTTCCCTCTCAGGGGTCTAATAGC
AGTTTGACACTCTGCATTAGCATTATCGTATGCAAGAAGCTGTATTACAACATCCTGAGCCGTTTTATCAGTTATGGCTTTATACACAGCCTCTTGGAACCAAGCAATAAAATCAACATATGGTTCT
TTACGTCCTTGTCGGACAGAACTGAAAGAAGGATATTTTTCCCTCGTAACATTTATCCTTTCCCACGCCTGTAAGCACACAAAGCGCAGCTGAACAATGGCAACATCCTCCATTACTGCTTGATTCT
CTAACCGACCCCAATTAGGGCCAACTCCCATTAACTGTTCAAAGGAAACAGGCACAGGTGGCTGTGCTTGTATGTTTTCCCTTGTCTGAGTTTGAGCTTCATCAGCCCACCAGGTTTTAAACTGCAA
GTACTGAGATAGAGTGAGAACAGATTTTGTCAAAGTATCCCAATCATACAGTGTTAACCTGTTATCAAGAGTCATATTTTTAAATAAAGTTTGCACAAAAGGAGGGTTCGGTCCGTATTAATGGTTT
GCTTAAATTCCTTTAGTAACTTAAAAGGAAAAGCGGCCCAATTAGTTATATTCTGTCCTCCTTGCTGGATTATAGTAATGGGAAATTGCCATGCTTCAAGGTCTCCCTTGGCTGTAGCTTTTTGAAT
AGAATTTTGTATAGTACCACCAATCGCTCCAGGTTTTAATGTTGCAACTACAGAAGCAGTAAGTTTTTCAGCTAATTTCTTTTCTTGCCCATTAAGGGGAGAGACAGGAGGTGGCCATTCACTTAAT
TCAGCAGGTGGAAGCGATGGGCTAGTAAAACATATTTTTATTTTCCTTTCTTTTCTTTAATCTCCTCTGGTAGCTGTTGCTCACACTCAGAATCTGAAGTTAGTTTTTTACACTTGTCCTCTTTCTC
ATCTGAATCTGCCTCATCATCTGTTTGAAATGGCTCAAGAGCCGTCTTTATTAGCACCCACACTGACCAAATGAAAACTGGAATTTCTGCTCCCTCTTTATATGCCTTTTTAAAATCTCTTCCAATT
CTCTCCCATTCATCCAACTCCACAGTCCCTTGTTCAGGAAACCATGGACAAAACTGCTTTACTGTACTAAAGAGTGATAACAAATTCTGAGTACTAACTTACTCCCCCTCTTCGTAATAAATGCCTT
AAGAAATGTAAATAAGCGGAATGTCTGCTTTCACTTTGTCCCATTGTTACTCTGGTTCTTCCAAGTGCTCAGCTTTCCTGTCAAGCTTCTTTTAGACCCCATGCTCTAGCGTTGCTTCACTGGGGTC
TTTGTCACCCCACGTTGGGCAGCCAGGAATGTTGGGGTGATCAGACCCAACACCAGGTCGTGGGGGCAACGAAGTCGGCAGAGTCAAAGGATTGAGAAAAAGACAGTTTGAGAGAGAGAAGTGGGAC
CAGAGGTCCATTGCTATTGTGGAGGCTGCTAAGGCCCCAAGCTCTGGGAGCCCACACTATTTATTGGTAATCCAACAAAGAAACAGGTGGTGAGGATGTGGGGGTTGAAAGGAAACAGTGTATCAAG
TGAATGAGAAACATATGGCTGCTTGACATAATGGCAGTGCTAGAAGCAAGGAGCCAGCAAGTCTAGGACACATGCAAGCCCTGCCTCAGCTTCTCTCCCAACACTCAGCTTTTCTCCCA
```



These is an ERV that includes an ancient version of the viral sequence. Note: it is on the reverse DNA strand.....

```
+TGGGAGAAAAGCTGAGTGTTGGGAGAGAAGCTGAGGCAGGG
CTTGCATGTCTGCTAGACTTGCTGGCTCCTTGCTTCTAGCAC
TCCCATTATCTCAAGCAGCCATATGTTTCTCATTCACTTGAT
ACACCGTTTCCTTTCAACCCCCACATCCTCACCACCTGTTTC
TTTGTTTGAGCACCAATAAATAGCGTGGGCTCCCAGAGCTCG
GGGCCTTCGCAGCCTCCACACTCGCGATGGCCCCCTGGTCCC
ACTTTCTCTCTCAAACTGTCTTTTTCTCATTCCTTTGACTCC
GCCGGACTTCGTCGCCCCCACGACCTGGTGTTGGGTCTGATC
ACCCCAACA
```

## Next Time

**Looking for hidden patterns in DNA without a template**