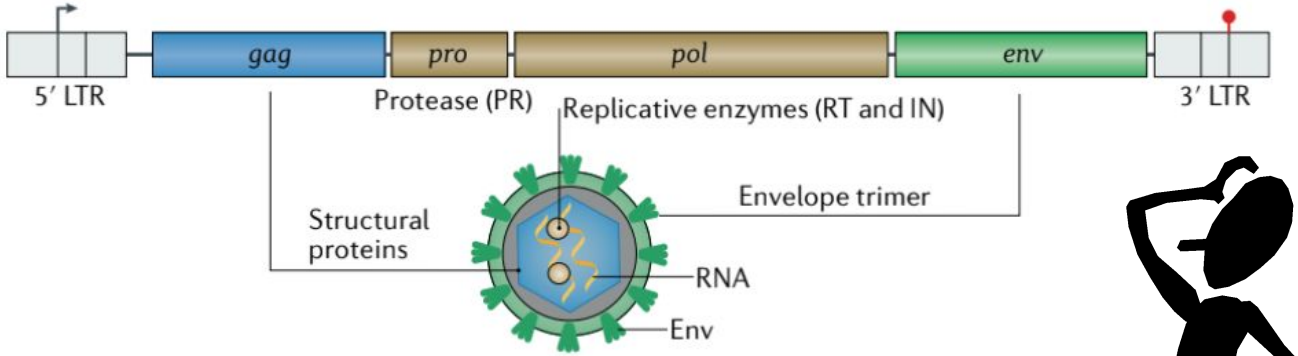
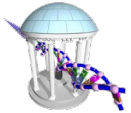
Comp 555 - BioAlgorithms - Spring 2022



PROBLEM SET #1 IS NOW POSTED AND IS DUE ON 2/3/2022.

Looking for Fossils

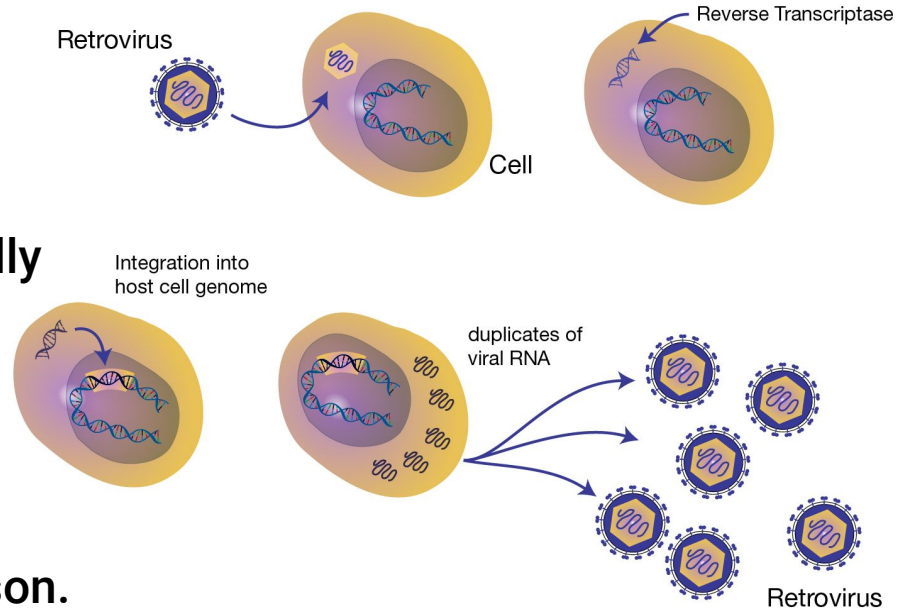
One class of TEs: Endogenous Retroviruses (ERV)



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

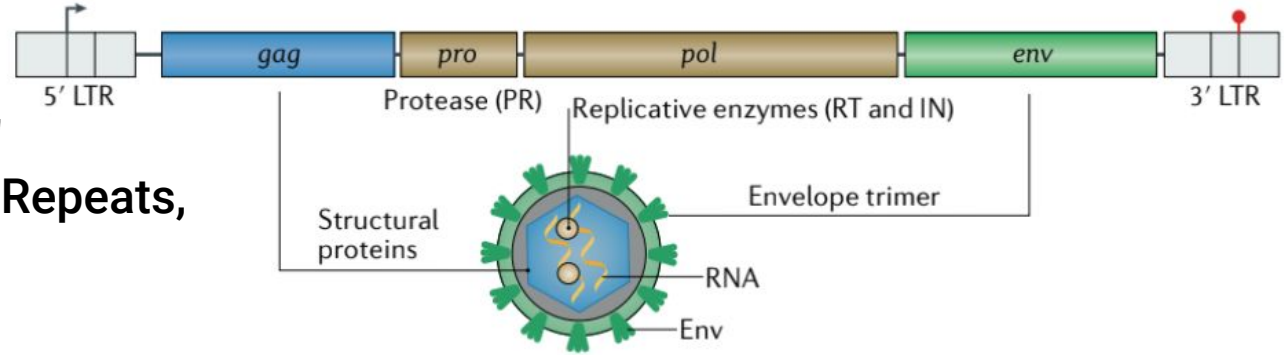
These "Endogenous" Retroviruses are generally dormant, but they occasionally awaken and, rather than leave the cell, they incorporate new copies of themselves back into the host DNA.

Thus, they are a form of Retrotransposon.

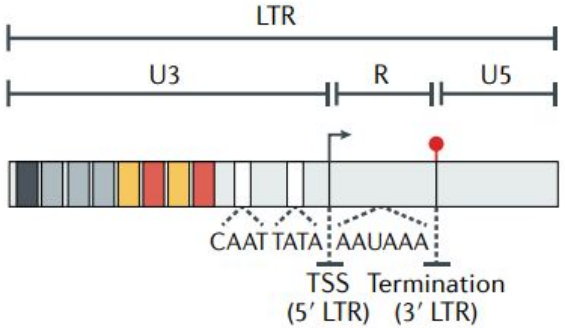


ERV genome structure

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

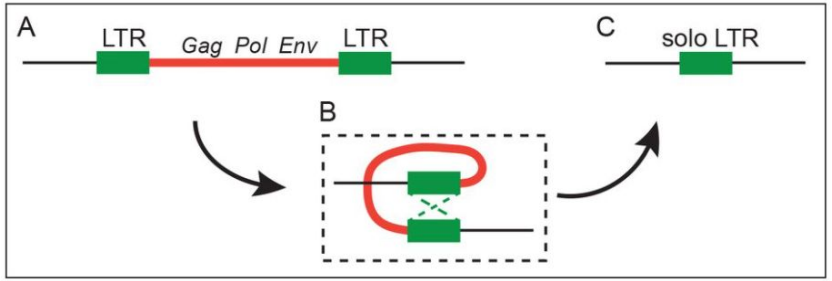


These LTRs contain the transcription start and end sites that are used when the ERV is copied (retrotransposed). These are parentheses enclosing the "proviral" sequence.



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.

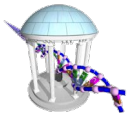
One way of cleaning the genome of ERVs is to remove their viral sequence.

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.

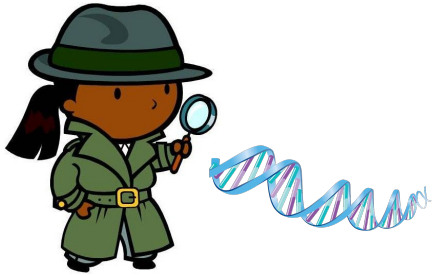
Let's look for some ERVs



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?

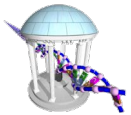
Luckily, biologists have used the first method to give us templates that we can use for the second approach.



There are databases with these "approximate" sequences.

Dfam ISB RepeatMasker

Getting started



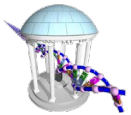
Some old code...

In [3]:
```
import gzip

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 [4]:
```
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
```

Looking at an LTR sequence



Let's get some sense of what we're looking for.

- **344 base pair consensus LTR sequence for an ERV class**
- **What is a consensus? (We'll quantify this concept next lecture)**
- **There are 0 exact matches in the genome**
- **Tyranny of consensus-- being close to many, but having no perfect match**

In [5]:
```
def revComp(dnaSeq):
    return ''.join([{'A':'T','C':'G','G':'C','T':'A'}[base] for base in reversed(dnaSeq)])
```

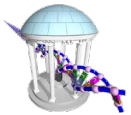
In [7]:
```
print(ltr)
```

```
+TGGGAGAAAAGCTGAGTGTTGGGAGAGAAGCTGAGGCAGGGCTTGCATGTCTGCTAGACTTGCTGGCTCCTTGCTTCTAGCACTCCCATTATCTCAAGCAGCCATATGTTTCTCATTCACTTGATA
CACCGTTTCCTTTCAACCCCCACATCCTCACCACCTGTTTCTTTGTTTGAGCACCAATAAATAGCGTGGGCTCCCAGAGCTCGGGGCCTTCGCAGCCTCCACACTCGCGATGGCCCCCTGGTCCCAC
TTTCTCTCTCAAACTGTCTTTTTCTCATTCCTTTGACTCCGCCGGACTTCGTCGCCCCCACGACCTGGTGTTGGGTCTGATCACCCCAACA
```

In [12]:
```
print("-"+revComp(ltr[1:]))
```

```
-TGTTGGGGTGATCAGACCCAACACCAGGTCGTGGGGGCGACGAAGTCCGGCGGAGTCAAAGGAATGAGAAAAAGACAGTTTGAGAGAGAAAGTGGGACCAGGGGGCCATCGCGAGTGTGGAGGCTG
CGAAGGCCCCGAGCTCTGGGAGCCCACGCTATTTATTGGTGCTCAAACAAAGAAACAGGTGGTGAGGATGTGGGGGTTGAAAGGAAACGGTGTATCAAGTGAATGAGAAACATATGGCTGCTTGAGA
TAATGGGAGTGCTAGAAGCAAGGAGCCAGCAAGTCTAGCAGACATGCAAGCCCTGCCTCAGCTTCTCTCCCAACACTCAGCTTTTCTCCCA
```

Then what do we search for?

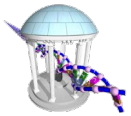


- **Find clusters of genome subsequences that are shared with the LTR**
- **In the same relative order**
- **"Signature" k-mers**
- **LTRs can be on either strand**

In [6]:
```
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))

# Check if any k-mer is in both lists
for key in forward:
    if key in reverse:
        print(key)
```

```
326
326
```

Let's go fishing

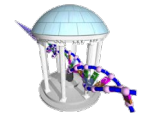


Now scan the genome looking for LTRs...

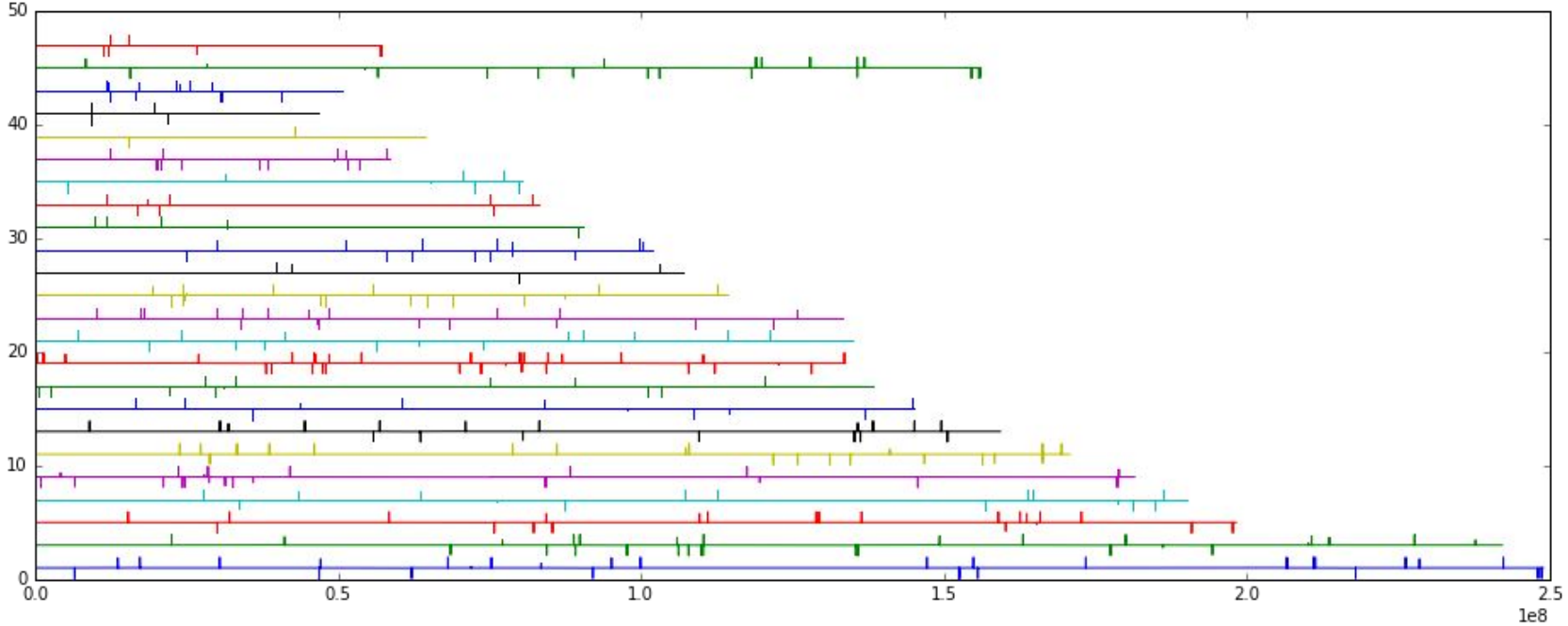
In [*]:
```
import time

DATA = "../HumanGenome/"
chromo = [str(i) for i in range(1,23)] + ['X', 'Y', 'MT']

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

```
1 248956423 1698 197.12 secs
2 242193530 1265 185.52 secs
3 198295560 1060 157.76 secs
4 190214556 786 163.40 secs
5 181538260 1243 146.19 secs
6 170805980 1393 148.28 secs
7 159345974 1301 138.78 secs
8 145138637 345 135.24 secs
9 138394718 511 124.19 secs
10 133797423 2181 171.12 secs
11 135086623 914 246.94 secs
12 133275310 638 265.58 secs
13 114364329 620 156.64 secs
14 107043719 209 139.83 secs
15 101991190 839 189.80 secs
16 90338346 173 128.09 secs
17 83257442 701 104.92 secs
18 80373286 288  63.82 secs
19 58617617 693  73.21 secs
20 64444168 118  58.78 secs
21 46709984 347  41.72 secs
22 50818469 924  46.05 secs
X 156040896 1665 137.28 secs
Y 57227416 391  48.07 secs
MT 16570 3   0.04 secs
```

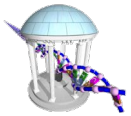
Big Picture



The LTRs of ERV14's are everywhere

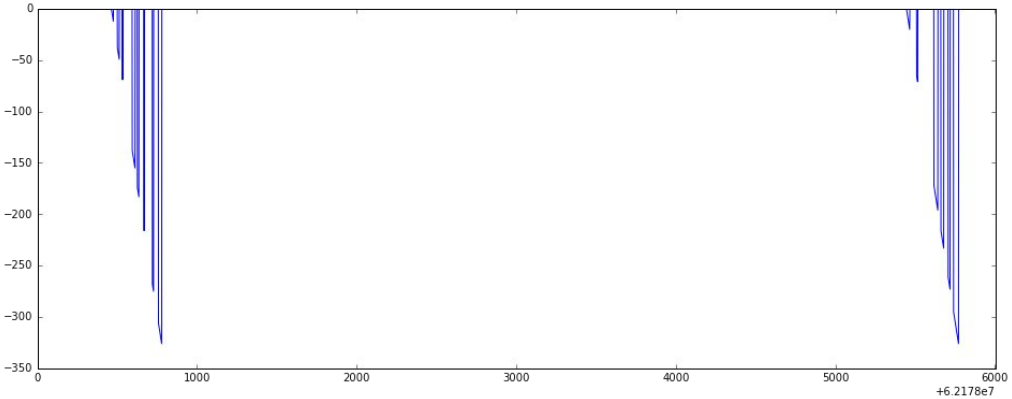


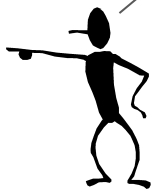
Let's take a Zoomed-in look



In [39]:
```
import matplotlib
import matplotlib.pyplot as plot
%matplotlib inline


j = 0      # chromosome 1

#lo, hi = (0, 250000000)
lo, hi = (62000000, 62500000)
#lo, hi = (99000000, 110000000)

x = [position for contig, position, offset in ltrFind[j] if position >= lo and position < hi]
y = [offset for contig, position, offset in ltrFind[j] if position >= lo and position < hi]
print(len(x))

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

226



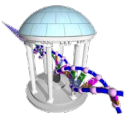


These are potential LTRs on chromosome 1.

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

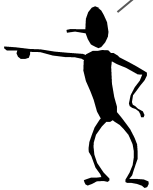
The most likely LTRs are those that share many k-mers in the expected order.

Going directly to the sequence



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

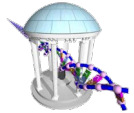
```
TGTTGGGGTGATCAGACCCAACACCAGGTCATGGGGACGACGAAGTCCGGCGGAGTCAAAGGAATGATTAAAAAGACAGTTTGAGAGAAGTGGGCCCAGGGGGCCATCGTGTGAGGCTGCGAAGGCC
CCAAGCTCTGGGAGCCCACGCTATTTATTGGTGCTCAAAGAAACAGGTGGTGAGGATGTGGGGTTTGAAAAGAAACAGTGTATCAAGTGAATGAGAGACATATGGCTACTTGAGATAATGGCAGTGC
TGGAAGCAAGGAGCCAGCAAGTCTAGCACACATGCAAGCTCTGCCTCAGCTTCTCTCCCAACACTCAGCTTTTCTCCCAACATGCCCCCCTTCTCTTTTTTGTA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