26-bioinformatics

April 24, 2016

1 BioPython

- large module for bioinformatics
- docs

In [ ]:
```
from Bio.Seq import *
s = Seq("AGTACACTGGT")
s
```

In [ ]:
```
[s.complement(), s.reverse_complement()]
```

2 Read Fasta format

- lady slipper orchid pics

In [ ]:
```
# need data in files

import urllib.request

def copyToFile(path, url):
    with urllib.request.urlopen(url) as nt:
        lines = nt.readlines()
        with open(path, "bw") as f:
            for line in lines:
                f.write(line)
```

In [ ]:
```
# places

url = 'https://raw.githubusercontent.com/biopython/biopython/master/Doc/examples/ls_orchid.fasta
url2 = 'https://raw.githubusercontent.com/biopython/biopython/master/Doc/examples/ls_orchid.gbk

path = '/tmp/orch.fasta'
path2 = '/tmp/orch.gbk'

bioin = '/Users/lstead/bioin/'
bioout = '/Users/lstead/bioout/'
```

In [71]:
```
from Bio import SeqIO

copyToFile(path, url)

l = list(SeqIO.parse(path, 'fasta'))
for seq_record in l[:10]:
```

```
            print(seq_record.id)
            print(repr(seq_record.seq))
            print(len(seq_record))
```

```
gi|2765658|emb|Z78533.1|CIZ78533
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGG...CGC', SingleLetterAlphabet())
740
gi|2765657|emb|Z78532.1|CCZ78532
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAG...GGC', SingleLetterAlphabet())
753
gi|2765656|emb|Z78531.1|CFZ78531
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGCAG...TAA', SingleLetterAlphabet())
748
gi|2765655|emb|Z78530.1|CMZ78530
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAAACAACAT...CAT', SingleLetterAlphabet())
744
gi|2765654|emb|Z78529.1|CLZ78529
Seq('ACGGCGAGCTGCCGAAGGACATTGTTGAGACAGCAGAATATACGATTGAGTGAA...AAA', SingleLetterAlphabet())
733
gi|2765652|emb|Z78527.1|CYZ78527
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGTAG...CCC', SingleLetterAlphabet())
718
gi|2765651|emb|Z78526.1|CGZ78526
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGTAG...TGT', SingleLetterAlphabet())
730
gi|2765650|emb|Z78525.1|CAZ78525
Seq('TGTTGAGATAGCAGAATATACATCGAGTGAATCCGGAGGACCTGTGGTTATTCG...GCA', SingleLetterAlphabet())
704
gi|2765649|emb|Z78524.1|CFZ78524
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATAGTAG...AGC', SingleLetterAlphabet())
740
gi|2765648|emb|Z78523.1|CHZ78523
Seq('CGTAACCAGGTTTCCGTAGGTGAACCTGCGGCAGGATCATTGTTGAGACAGCAG...AAG', SingleLetterAlphabet())
709
```

In [72]:
```
copyToFile(path2, url2)
```

```
        for seq_record in list(SeqIO.parse(path2, 'genbank'))[:10]:
            print(seq_record.id)
            print(repr(seq_record.seq))
            print(len(seq_record))
```

```
Z78533.1
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGG...CGC', IUPACAmbiguousDNA())
740
Z78532.1
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAG...GGC', IUPACAmbiguousDNA())
753
Z78531.1
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGCAG...TAA', IUPACAmbiguousDNA())
748
Z78530.1
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAAACAACAT...CAT', IUPACAmbiguousDNA())
744
Z78529.1
```

```
Seq('ACGGCGAGCTGCCGAAGGACATTGTTGAGACAGCAGAATATACGATTGAGTGAA...AAA', IUPACAmbiguousDNA())
733
Z78527.1
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGTAG...CCC', IUPACAmbiguousDNA())
718
Z78526.1
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGTAG...TGT', IUPACAmbiguousDNA())
730
Z78525.1
Seq('TGTTGAGATAGCAGAATATACATCGAGTGAATCCGGAGGACCTGTGGTTATTCG...GCA', IUPACAmbiguousDNA())
704
Z78524.1
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATAGTAG...AGC', IUPACAmbiguousDNA())
740
Z78523.1
Seq('CGTAACCAGGTTTCCGTAGGTGAACCTGCGGCAGGATCATTGTTGAGACAGCAG...AAG', IUPACAmbiguousDNA())
709
```

In [73]:
```
# different alphabets for different purposes
# DNA sequence

s2 = Seq("AGTACACTGGT", IUPAC.unambiguous_dna)
print(s2, s2.alphabet)

# amino acids
s3 = Seq('AGTACACTGGT', IUPAC.protein)
print(s3, s3.alphabet)
```

```
AGTACACTGGT IUPACUnambiguousDNA()
AGTACACTGGT IUPACProtein()
```

In [74]:
```
# Seq is not a string, but has string like features

[s, s[1], len(s), s.count('A'), s[3:8]]
```

Out[74]:
```
[Seq('AGTACACTGGT', Alphabet()), 'G', 11, 3, Seq('ACACT', Alphabet())]
```

In [75]:
```
# compute % of G and C

100 * float(s.count("G") + s.count("C")) / len(s)
```

Out[75]:
```
45.45454545454545
```

In [76]:
```
# builtin function

from Bio.SeqUtils import GC
GC(s)
```

Out[76]:
```
45.45454545454545
```

In [77]:
```
from reportlab.lib import colors
from reportlab.lib.units import cm
from Bio.Graphics import GenomeDiagram
from Bio import SeqIO

url='https://raw.githubusercontent.com/biopython/biopython/dbb0de6337d5604ea5a5b2276dfb0ae0d2d
```

```
path = '/tmp/NC_005816.gb'

copyToFile(path, url)

record = SeqIO.read(path, "genbank")

gd_diagram = GenomeDiagram.Diagram("Yersinia pestis biovar Microtus plasmid pPCP1")
gd_track_for_features = gd_diagram.new_track(1, name="Annotated Features")
gd_feature_set = gd_track_for_features.new_set()
```

In [78]:
```
for feature in record.features:
    if feature.type != "gene":
        #Exclude this feature
        continue
    if len(gd_feature_set) % 2 == 0:
        color = colors.blue
    else:
        color = colors.lightblue
    gd_feature_set.add_feature(feature, color=color, label=True)

gd_diagram.draw(format="linear", orientation="landscape", pagesize='A4',
                fragments=4, start=0, end=len(record))
gd_diagram.write(bioout+"plasmid_linear.pdf", "PDF")
gd_diagram.write(bioout+"plasmid_linear.eps", "EPS")
gd_diagram.write(bioout+"plasmid_linear.svg", "SVG")
gd_diagram.draw(format="circular", circular=True, pagesize=(20*cm,20*cm),
                start=0, end=len(record), circle_core=0.7)
gd_diagram.write(bioout+"plasmid_circular.pdf", "PDF")
```

In [79]:
```
from reportlab.lib import colors
from reportlab.lib.units import cm
from Bio.Graphics import GenomeDiagram
from Bio import SeqIO
from Bio.SeqFeature import SeqFeature, FeatureLocation

record = SeqIO.read(path, "genbank")

gd_diagram = GenomeDiagram.Diagram(record.id)
gd_track_for_features = gd_diagram.new_track(1, name="Annotated Features")
gd_feature_set = gd_track_for_features.new_set()

for feature in record.features:
    if feature.type != "gene":
        #Exclude this feature
        continue
    if len(gd_feature_set) % 2 == 0:
        color = colors.blue
    else:
        color = colors.lightblue
    gd_feature_set.add_feature(feature, sigil="ARROW",
                               color=color, label=True,
                               label_size = 14, label_angle=0)

#I want to include some strandless features, so for an example
#will use EcoRI recognition sites etc.
```

```
        for site, name, color in [("GAATTC","EcoRI",colors.green),
                                  ("CCCGGG","SmaI",colors.orange),
                                  ("AAGCTT","HindIII",colors.red),
                                  ("GGATCC","BamHI",colors.purple)]:
            index = 0
            while True:
                index  = record.seq.find(site, start=index)
                if index == -1 : break
                feature = SeqFeature(FeatureLocation(index, index+len(site)))
                gd_feature_set.add_feature(feature, color=color, name=name,
                                           label=True, label_size = 10,
                                           label_color=color)
                index += len(site)

        gd_diagram.draw(format="linear", pagesize='A4', fragments=4,
                        start=0, end=len(record))
        gd_diagram.write(bioout+"plasmid_linear_nice.pdf", "PDF")
        gd_diagram.write(bioout+"plasmid_linear_nice.eps", "EPS")
        gd_diagram.write(bioout+"plasmid_linear_nice.svg", "SVG")

        gd_diagram.draw(format="circular", circular=True, pagesize=(20*cm,20*cm),
                        start=0, end=len(record), circle_core = 0.5)
        gd_diagram.write(bioout+"plasmid_circular_nice.pdf", "PDF")
        gd_diagram.write(bioout+"plasmid_circular_nice.eps", "EPS")
        gd_diagram.write(bioout+"plasmid_circular_nice.svg", "SVG")
```

In [80]:
```
from Bio import SeqIO
entries = [("Chr I", "NC_003070.fna"),
           ("Chr II", "NC_003071.fna"),
           ("Chr III", "NC_003074.fna"),
           ("Chr IV", "NC_003075.fna"),
           ("Chr V", "NC_003076.fna")]
for (name, filename) in entries:
   record = SeqIO.read(bioin+filename,"fasta")
   print(name, len(record))
```

```
Chr I 30494425
Chr II 19705359
Chr III 23470805
Chr IV 18585042
Chr V 26992728
```

In [ ]:
```
from reportlab.lib.units import cm
from Bio.Graphics import BasicChromosome

entries = [("Chr I", 30432563),
           ("Chr II", 19705359),
           ("Chr III", 23470805),
           ("Chr IV", 18585042),
           ("Chr V", 26992728)]

max_len = 30432563 #Could compute this
telomere_length = 1000000 #For illustration

chr_diagram = BasicChromosome.Organism()
```

```
    chr_diagram.page_size = (29.7*cm, 21*cm) #A4 landscape

    for name, length in entries:
        cur_chromosome = BasicChromosome.Chromosome(name)
        #Set the scale to the MAXIMUM length plus the two telomeres in bp,
        #want the same scale used on all five chromosomes so they can be
        #compared to each other
        cur_chromosome.scale_num = max_len + 2 * telomere_length

        #Add an opening telomere
        start = BasicChromosome.TelomereSegment()
        start.scale = telomere_length
        cur_chromosome.add(start)

        #Add a body - using bp as the scale length here.
        body = BasicChromosome.ChromosomeSegment()
        body.scale = length
        cur_chromosome.add(body)

        #Add a closing telomere
        end = BasicChromosome.TelomereSegment(inverted=True)
        end.scale = telomere_length
        cur_chromosome.add(end)

        #This chromosome is done
        chr_diagram.add(cur_chromosome)

    chr_diagram.draw(bioout+"simple_chrom.pdf", "Arabidopsis thaliana")
```

In [ ]:
```
    from reportlab.lib.units import cm
    from Bio import SeqIO
    from Bio.Graphics import BasicChromosome

    entries = [("Chr I", "NC_003070.gbk"),
               #("Chr II", "NC_003071.gbk"),
               ("Chr III", "NC_003074.gbk"),
               ("Chr IV", "NC_003075.gbk"),
               ("Chr V", "NC_003076.gbk")]

    max_len = 30432563 #Could compute this
    telomere_length = 1000000 #For illustration

    chr_diagram = BasicChromosome.Organism()
    chr_diagram.page_size = (29.7*cm, 21*cm) #A4 landscape

    for index, (name, filename) in enumerate(entries):
        print(bioin+filename)
        record = SeqIO.read(bioin+filename,"genbank")
        length = len(record)
        print(length)
        features = [f for f in record.features if f.type=="tRNA"]
        #Record an Artemis style integer color in the feature's qualifiers,
        #1 = Black, 2 = Red, 3 = Green, 4 = blue, 5 =cyan, 6 = purple
        for f in features: f.qualifiers["color"] = [index+2]
```

```
            cur_chromosome = BasicChromosome.Chromosome(name)
            #Set the scale to the MAXIMUM length plus the two telomeres in bp,
            #want the same scale used on all five chromosomes so they can be
            #compared to each other
            cur_chromosome.scale_num = max_len + 2 * telomere_length

            #Add an opening telomere
            start = BasicChromosome.TelomereSegment()
            start.scale = telomere_length
            cur_chromosome.add(start)

            #Add a body - again using bp as the scale length here.
            body = BasicChromosome.AnnotatedChromosomeSegment(length, features)
            body.scale = length
            cur_chromosome.add(body)

            #Add a closing telomere
            end = BasicChromosome.TelomereSegment(inverted=True)
            end.scale = telomere_length
            cur_chromosome.add(end)

            #This chromosome is done
            chr_diagram.add(cur_chromosome)

        chr_diagram.draw(bioout+"tRNA_chrom.pdf", "Arabidopsis thaliana")
```

In [ ]: