Bioinformática com Biopython

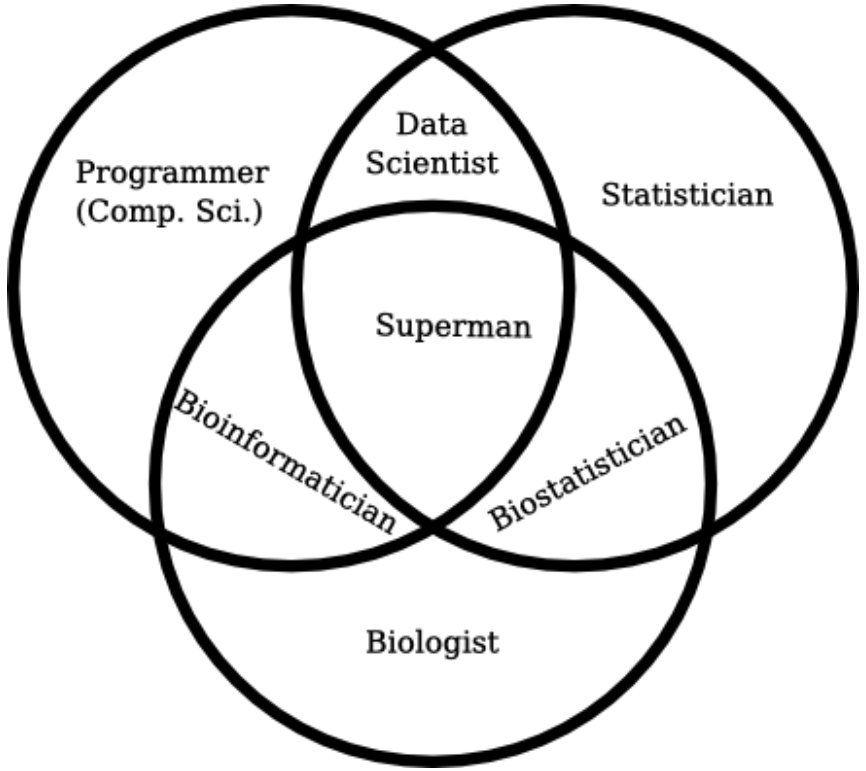
Python Brasil 11

Marcos Castro

Apresentação

- Nome: Marcos Castro
  - Formação: Ciência da Computação.
  - Obs.: espero não assassinar a Biologia...
- Grupo de Bioinformática Unifesp-SJC
  - Apresentações toda sexta.
  - O café do laboratório é famoso!
- Objetivos da palestra:
  - Incentivar o uso de Python em Bioinformática.
  - Uso de Biopython para otimizar o seu tempo.

Bioinformática



O que é Biopython?

- Conjuntos de ferramentas gratuitas para bioinformática.
- Site oficial: http://biopython.org/
- Código Biopython: https://github.com/biopython/biopython
- Biopython Tutorial: http://biopython.org/DIST/docs/tutorial/Tutorial.pdf
- Códigos da apresentação: https://github.com/marcoscastro/pybr11
- Slides: https://speakerdeck.com/marcoscastro/bioinformatica-com-biopython

Instalação

- Suporte para diversos sistemas operacionais:
  - http://biopython.org/wiki/Download
- Versões:
  - http://biopython.org/DIST/
- Pelo pip:
  - pip install biopython

Hello Biopython

objeto Seq

```
from Bio.Seq import Seq

s = Seq('ACTG')
print s
```

Complementar e reverso complementar

- Complemento:

```
sequencia = Seq('ACTG')
print sequencia.complement() # imprime TGAC
```

- Reverso complementar:

```
sequencia = Seq('ACTG')
print sequencia.reverse_complement() # imprime CAGT
```

Transcrição

- Transcrição: onde tem “T” troca por “U” (recordando o ensino médio).
- IUPAC: define padrões de alfabetos para nucleotídeos/proteínas.

```
1  from Bio.Seq import Seq
2  from Bio.Alphabet import IUPAC
3
4  dna = Seq('ACTG', IUPAC.unambiguous_dna)
5  rna = dna.transcribe()
6  print rna # imprime ACUG
7  print rna.back_transcribe() # imprime ACTG
```

Tradução

- De RNA para proteína:

```
rna = Seq('GUGCCC', IUPAC.unambiguous_rna)
proteina = rna.translate()
print proteina
```

- Tradução direta (DNA para proteína):

```
dna = Seq('ATGCTA', IUPAC.unambiguous_dna)
prot_dna = dna.translate()
print prot_dna
```

Parsers – Bio.SeqIO

- O pacote BioSeqIO fornece suporte a vários formatos tais como fasta, fastq etc.
- FASTA
  - Formato muito utilizado para armazenamento de sequências biológicas.

```
>PYTHON_BRASIL
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCT
CCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATG
CTCCGGGGCCACGGCCACCGCTGCCCTGCCCCGGCCGAGCG
```

Parsers – Bio.SeqIO

- Lendo arquivo FASTA:

```
from Bio import SeqIO

for i in SeqIO.parse('exemplo_fasta', 'fasta'):
    print i.id # imprime o cabeçalho
    print i.seq # imprime a sequência
```

Parsers – Bio.SeqIO

- Gerando arquivo FASTA:

```
1  from Bio import SeqIO
2  from Bio.Seq import Seq
3  from Bio.SeqRecord import SeqRecord
4
5  registro = SeqRecord(Seq('ACTG'), id='pybr11')
6  arq = open('arquivo.fasta', 'w')
7  SeqIO.write(registro, arq, 'fasta')
8  arq.close()
```

Parsers – Bio.SeqIO

- Lendo arquivo genbank:

```
1  from Bio import SeqIO
2
3  for i in SeqIO.parse('exemplo_genbank.gbk', 'genbank'):
4      print i.id
5      print i.seq
```

Parsers – Bio.SeqIO

- Conversões FASTQ -> FASTA e FASTQ -> QUAL

```
from Bio import SeqIO

SeqIO.convert('exemplo.fastq', 'fastq',\
              'exemplo.fasta', 'fasta')

SeqIO.convert('exemplo.fastq', 'fastq',\
              'exemplo.qual', 'qual')
```

Parsers – Bio.SeqIO

- Conversão genbank -> FASTA

```
from Bio import SeqIO

SeqIO.convert('exemplo_genbank.gbk', 'genbank', \
              'exemplo_fasta.fasta', 'fasta')
```

Parsers – Bio.SeqIO

- Ordenando arquivos Multi-FASTA pelo tamanho:

```
from Bio import SeqIO

seqs = list(SeqIO.parse('multifasta.txt', 'fasta'))
seqs.sort(cmp=lambda x,y: cmp(len(x), len(y)))
SeqIO.write(seqs, 'multifasta_sorted.txt', 'fasta')
```

Alinhamentos

- Módulo pairwise2: from Bio.pairwise2 import *

```
alignments = align.globalxx('ACCGT', 'ACG')
for a in alignments:
    print format_alignment(*a)

alignments = align.localxx('ACCGT', 'CCG')
for a in alignments:
    print format_alignment(*a)
```

BLAST remoto

- BLAST: Basic Local Alignment Search Tool
- Vários ferramentas: blastn (nucleotídeos), blatsp (proteínas) etc.

```
1  from Bio import SeqIO
2  from Bio.Blast import NCBIWWW
3
4  arq = SeqIO.read('exemplo_fasta', format='fasta')
5  print 'Buscando...'
6  result = NCBIWWW.qblast('blastn', 'nt', \
7                          arq.seq, format_type='Text')
8  print result.read()
9  print 'Fim!'
```

BLAST local

- Requer que a suíte de aplicativos esteja instalada:
  - ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST

```
1  from Bio.Blast.Applications import *
2
3  comando = NcbiblastnCommandline(
4                      query='seq1.fasta',
5                      subject='seq2.fasta',
6                      outfmt=0, out='saida.txt')
7  stdout, stderr = comando()
8  result = open('saida.txt', 'r')
9  print result.read()
```

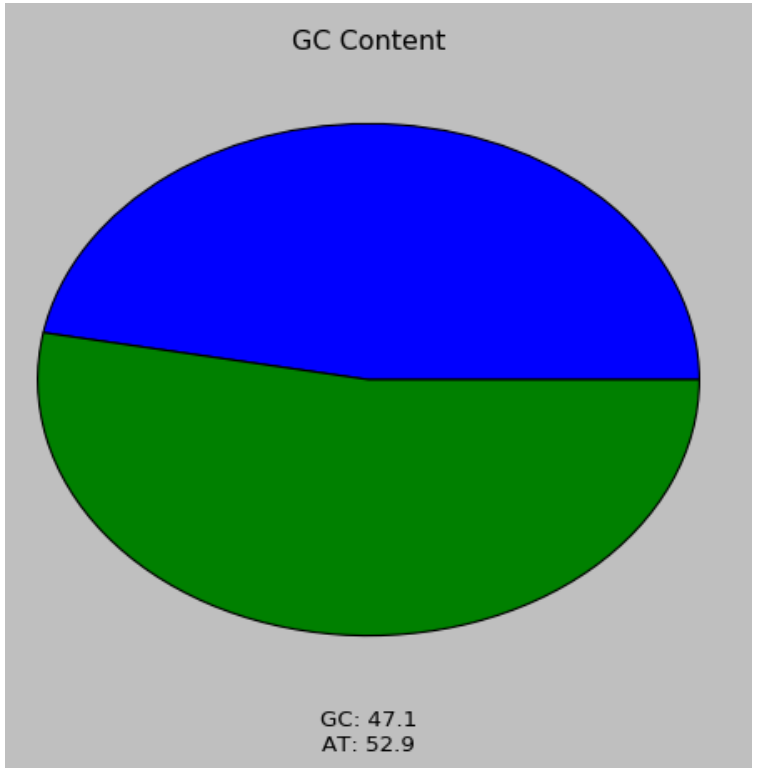
Visualização de dados

- Necessita do módulo pylab (pip install matplotlib)
- Conteúdo GC (GC content):

```
1  from Bio.Seq import Seq
2  from Bio.SeqUtils import GC
3  import pylab
4
5  seq = Seq('CTTGCACTGACATCGAT')
6  gc = GC(seq)
7  at = 100 - gc
8
9  pylab.pie([gc, at])
10 pylab.title('GC Content')
11 pylab.xlabel('GC: %0.1f\nAT: %0.1f' % (gc, at))
12 pylab.show()
```

Visualização de dados

- Conteúdo GC (GC content):

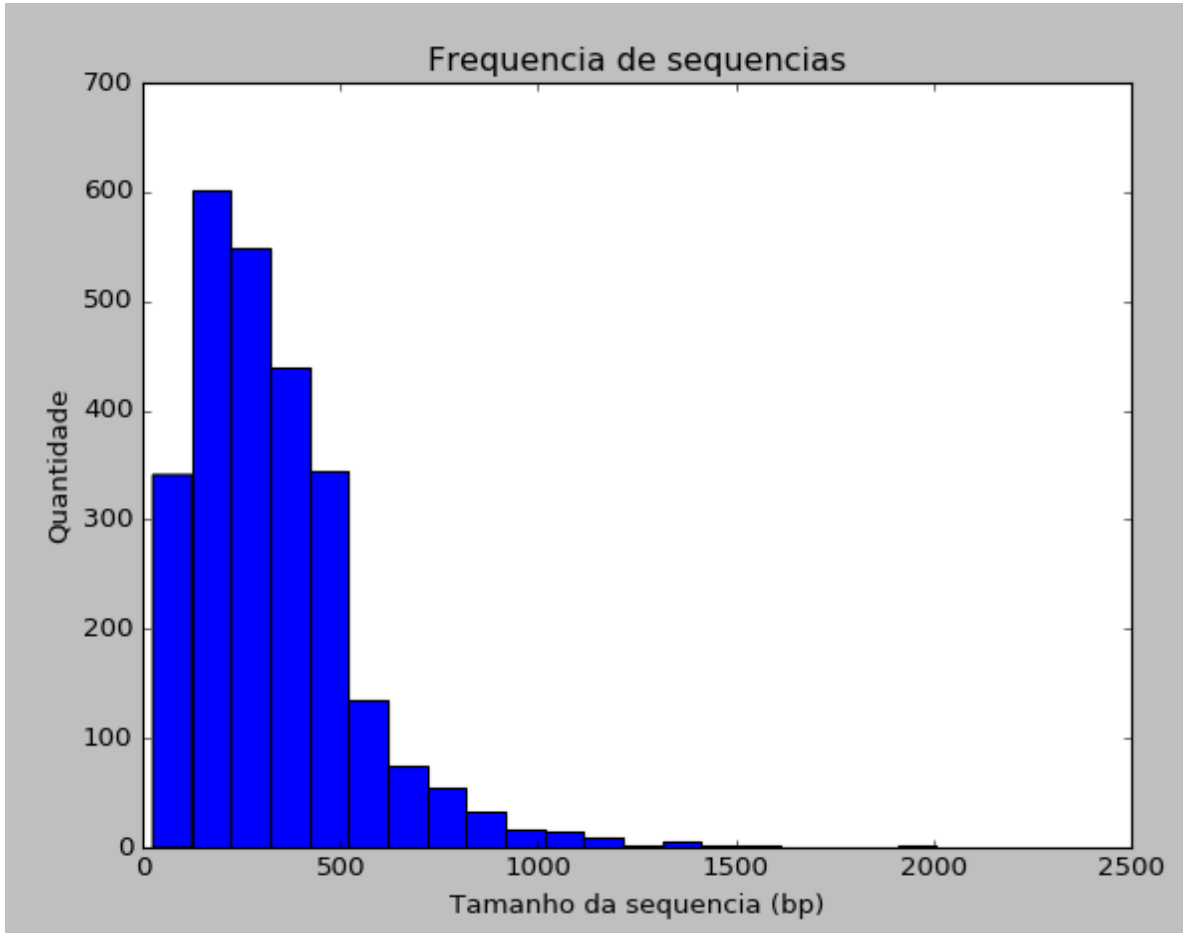


Visualização de dados

- Quantidade de genes pelo tamanho da sequência codificadora.

```
1  from Bio import SeqIO
2  import pylab
3
4  arq = SeqIO.read('NC_017108.gbk', 'genbank')
5
6  tamanhos = [len(i.qualifiers['translation'][0]) \
7              for i in arq.features if i.type == 'CDS']
8
9  pylab.hist(tamanhos, bins=20)
10 pylab.title('Frequencia de sequencias')
11 pylab.xlabel('Tamanho da sequencia (bp)')
12 pylab.ylabel('Quantidade')
13 pylab.show()
```

Visualização de dados



Cálculo do N50

```
1  from Bio import SeqIO
2
3  seqs = list(SeqIO.parse('multifasta.txt', 'fasta'))
4  tamanhos = [len(seq) for seq in seqs]
5
6  tamanhos.sort()
7  metade = sum(tamanhos) / 2
8  soma_tamanhos = 0
9
10 for tam in tamanhos:
11     soma_tamanhos += tam
12     if soma_tamanhos > metade:
13         print 'N50: %s' % tam
14         break
```

Árvore Trie

- http://bioinformatics.cvr.ac.uk/blog/trie-data-structure/

```
from Bio.trie import trie
from Bio import triefind

t = trie()
genome = 'ACGTA'
print triefind.find(genome, t)
```

Algoritmos Genéticos

- Exemplo:

```
for i in range(pop_size):
    seq = MutableSeq(''.join(\
            random_base() for j in range(seq_size)),\
            IUPAC.unambiguous_dna)
    pop.append(Organism(seq, fitness))

cross = SinglePointCrossover(cross_rate)
mut = SinglePositionMutation(mut_cross)
roulette = RouletteWheelSelection(mut, cross)
```

Algoritmos Genéticos

- Exemplo:

```
for i in range(max_gen):
    pop = roulette.select(pop)
    for j in range(pop_size):
        pop[j].fitness = fitness(pop[j].genome)
```

- http://biopython.org/DIST/docs/api/Bio.GA-module.html

kNN (k-nearest-neighbors classification)

- Necessita do pacote numpy.

```
1  from Bio import kNN
2
3  xs = [
4          [10, 10, 10, 10], [9, 11, 10, 10],
5          [10, 10, 11, 9], [25, 5, 5, 5],
6          [9, 9, 11, 11], [15, 15, 5, 5],
7          [5, 10, 5, 20], [9, 9, 9, 13],
8          [9, 11, 11, 9], [20, 10, 5, 5],
9          [9, 10, 11, 10], [5, 15, 15, 5]
10     ]
11
12 k = 3
13 ys = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0]
14 knn = kNN.train(xs, ys, k)
15
16 new_example = [5, 25, 5, 5]
17 print kNN.classify(knn, new_example)
```

Aprendendo Bioinformática

- Introdução à Programação para Bioinformática com Biopython:
  - http://www.amazon.com/dp/B015IK1C4O/
- Documentação Biopython:
  - http://pydoc.net/Python/biopython/1.63
- Artigo Biopython:
  - http://goo.gl/yYSyLs
- Rosalind:
  - Aprenda Bioinformática resolvendo problemas.
  - http://rosalind.info/
- Blog Bioinformática:
  - http://bioinformatica.blog.br/

Aprendendo Bioinformática

- An Introduction to Bioinformatics Algorithms (Pavel Pevzner)
- Curso de Bioinformática com Python:
  - https://goo.gl/kwiZf7
- Várias apresentações sobre temas relacionados à Bioinformática:
  - https://speakerdeck.com/marcoscastro
  - http://slideshare.net/mcastrosouza

Contato

mcastrosouza@live.com

https://twitter.com/mcastrosouza

http://bioinformatica.blog.br/

Dúvidas?

404 — Sequence Not Found

GACCCAGCAATGACGTATACATGGCTTAAT
GAA