

# Simplified Bioinformatics Module

## 简明生物信息学模块

## Biopython, Bioperl...

这些模块使用频率较低的原因：

1. 使用起来太复杂，学习成本高
2. 包含了很多不常用的功能
  - biopython的使用手册长达343页

使用基础分析模块的好处：

1. 不需要重复的编写代码
2. 更快速、准确地解决问题
3. 更加方便的程序开发

这样，我们就可以把时间和精力放到更加重要的分析工作上，而不是纠结编程相关的问题。

# Simplified Bioinformatics Module(SBM)

## 简明生物信息学模块

- 集合最常用的功能
- 操作简单、易上手、人性化
- 既能直接使用，又能为二次开发提供便利

# SBM中的对象

- **Fasta**对象：处理Fasta文件
  - **Length**对象：处理序列的长度信息
- **Sequence**对象：处理DNA序列
- **Table**对象：处理表格格式的文件
  - 列与列之间以tab等分割符号分开，例如BLAST的输出结果。  
Excel可以处理的，它都能够处理
  - 基于Table对象-> **Gff3**对象，**Blast**对象
- **Cluster**对象：处理聚类工具的输出结果

# SBM安装

GitHub - zhengshuai0928/ x +

← → ↻ 🔒 GitHub, Inc. [US] | https://github.com/zhengshuai0928/SBM

## Simplified Bioinformatics Module


bioinformatics-tool python-module python bioinformatics-module python-script python3

🕒 14 commits

🌿 1 branch

📦 0 releases

Branch: master ▾ New pull request

 **zhengshuai0928** minor modification

📁 SBM	minor modification
📁 __pycache__	when input id not exist in fasta; remove the prompts of fuzzy match
📁 example	example/fasta
📄 LICENSE	init
📄 README.md	readme
📄 sbm.py	minor modification

# SBM安装

- `git clone https://github.com/zhengshuai0928/SBM.git`
- 将`sbm.py`文件和SBM文件夹放到python的搜索路径下
- python版本: `python3+`
- 操作系统: `Linux; Mac Linux terminal`
- 依赖模块: `numpy`

# SBM使用

首先，导入模块

```
>>> from sbm import *  
### Simplified Bioinformatics Module (SBM) 1.0  
### You gonna love using this module!^_^  
>>>
```

# SBM基本使用方法

1. 创建对象
2. 调用方法（函数）



# SBM使用（Fasta对象）

## 1. 创建对象

```
>>> fa = Fasta('example/fastas.fas')
Fasta reading in...
[#####] 100.00%
>>>
```

- 读入速度与Fasta文件的行数成正比

## 2. 调用函数（方法），实现功能

```
>>>
>>> fa.rmDups()    #去重
>scaffold135_size1862_120290_126299
  duplicates >scaffold16_size95689_3849877_3855886
```

## SBM使用（Fasta对象）

```
>>>  
>>> fa.toFile('new.fasta')      #将处理后的文件输出  
Writing to new.fasta ...  
[#####]100.00%  
>>>
```

```
>>>  
>>> fa.getseqs_gff('example/gff3.gff3', 'gene', 'gene.fas')  
>>>
```

#根据Gff3文件提取特征序列

## SBM使用（Fasta对象/Length对象）

```
>>>
>>> fa.len().n50() 等价于-> >>> leng = fa.len()
>>> leng.n50()
```

```
-----
Total number:      3948
Total length:      27294857
  Max length:      9369690
  Min length:       106
      N50:          10474
      N90:          2445
    >10M:           0
    >1M:             1
   >100K:           1
   >10K:            487
   >1K:             3947
```

```
>>>
```

#计算N50

## SBM使用（Fasta对象）

```
>>>
>>> dir(Fasta)
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__',
 '__eq__', '__format__', '__ge__', '__getattr__', '__getitem__',
 '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__',
 '__lt__', '__module__', '__ne__', '__new__', '__reduce__',
 '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__',
 '__subclasshook__', '__weakref__', 'gcContent', 'getseqs_gff',
 'getseqs_id', 'items', 'keys', 'len', 'rmDups', 'setAttrs',
 'toFile', 'values']
>>>
```

#查看Fasta对象的所有方法

## SBM使用（Fasta对象）

```
>>>  
>>> help(Fasta.getseqs_gff)  
  
Help on function getseqs_gff in module SBM.fasta:  
  
getseqs_gff(self, gff3_path, feature, out_path, mode='w')  
    Extract feature sequences according to Gff3 file.  
    Note: extracted sequences will be connected together in order.  
(END)
```

#使用help查看函数的功能和用法



## SBM使用（Fasta对象）

键和值一一对应，可根据键查值

- Fasta对象一个类字典的对象，可以方便地进行查找（支持模糊查询）和遍历

比如要查看ID为‘>scaffold16\_size95689\_859\_6011’的序列

```
>>>
>>> fa['>scaffold16_size95689_859_6011']
'CAGCCTTATTTTAGGAGGCATAACTTTTGAAGCTTCCCCCCACATATTAGGTGGC
ATTACTTCCAGAGCAATCCCCCCTTATTAGGAGGCATTGCTTTCGGAGCA'
>>>
```

```
>>>
>>> fa['scaffold16_size95689_859']
'CAGCCTTATTTTAGGAGGCATAACTTTTGAAGCTTCCCCCCACATATTAGGTGGC
ATTACTTCCAGAGCAATCCCCCCTTATTAGGAGGCATTGCTTTCGGAGCA'
>>>
```

模糊查询

## SBM使用（Fasta对象）

遍历：

for key in fa:

*do something with key*

*or do something with fa[key]*

```
>>> for key in fa:  
...     print(key)  
...     print(fa[key])
```

- 方便的模糊查询和遍历
- 简化了处理fasta文件的流程，便于基于Fasta对象开发新的脚本

# SBM使用

其他对象使用方法类似

1. 创建对象
2. 调用方法

Sequence对象

Table对象

Gff3对象

Blast对象

Cluster对象

...



## SBM使用(Table对象)

- 处理表格格式的文件，例如

element	start	element	end	element	length	sequence
859	6011	5153	scaffold16_size95689	859	1192	
105971	108252	2282	scaffold16_size95689	105971	106347	
123502	132063	8562	scaffold16_size95689	123502	123708	
247967	253271	5305	scaffold16_size95689	247967	248212	
709247	711263	2017	scaffold16_size95689	709247	710085	
874060	876552	2493	scaffold16_size95689	874060	874915	
892206	899604	7399	scaffold16_size95689	892206	892480	
1145437	1147788	2352	scaffold16_size95689	1145437	1145779	
1904919	1907558	2640	scaffold16_size95689	1904919	1905780	
1953385	1955089	1705	scaffold16_size95689	1953385	1953864	

# SBM使用(Table对象)

## 1. 创建对象

tb = Table(文件路径, 分割符号, 是否有标题)

```
tb = Table('example/table.csv', delimiter='\t', header=True)
```

- 文件被解析为行和列
- 支持按列名操作
- 类似Excel

```
>>>  
>>> tb.col_names #查看列名  
['start', 'end', 'length', 'sequence']  
>>>
```

## SBM使用(Table对象)

方便的查询操作:

列名: 'start', 'end', 'length', 'sequence'

第1行第2列: `tb[0][1]` 或 `tb[0]['end']`

第1行所有列: `tb[0][:]`

第3列: `tb[:,2]` 或 `tb[:, 'length']`

- 行和列编号从0开始


## SBM使用(Table对象)

### 2. 调用方法

```
>>> tb.sort(2, desc=True)
```

||

```
>>> tb.sort('length', desc=True)
```



#按照第三列排序，desc=True表示倒序排列

- 使用列号和列名都可以，
- *注意列号从0开始*

# SBM使用(Table对象)

## 2. 调用方法

```
>>>  
>>> tb.filter('length > 5000')    #按条件进行筛选  
Filter: column length > 5000  
[#####]100.00%  
>>>
```

筛选表达式格式:

列名 比较运算符 值

例如: length > 5000



也可以使用列编号

例如 2 > 5000 #筛选出第3列的值大于5000的所有行

## SBM使用(Table对象)

遍历:

```
for row in tb:
    do something with row[:] (列表 (list) 型变量)
    for col in row:
        do something with col (字符串 (str) 变量)
```

```
>>>
>>> for row in tb:
...     print(row)
...     for col in row:
...         print(col)
... 
```

#打印每一行，再打印该行的每一列

## SBM使用(Table对象)

时刻不要忘了

`dir()`

`help()`

## SBM使用(Table对象)

- 很多软件的输出结果都是Table格式
- 基于Table对象，可以进行很多扩展（设计新的对象），比如Gff3, Blast
- SBM中除Table外，内置了Gff3 和 Blast对象



# SBM使用(Blast对象)

## 1. 创建对象

```
>>> bl = Blast('example/blast.csv')
```

#处理标准格式的Blast输出结果，无需额外参数，列名会自动加上，可通过`bl.col_names`查看

#非标准格式，可通过`fmt=''`指定列名，例如

```
bl = Blast('blast_out', fmt='seqid e-value bit_score')
```

列名间以空格分开

## 2. 调用方法

```
>>> bl.filter('evalue<1e-3') #筛选出e值小于 $10^{-3}$ 的所有行
Filter: column evalue < 1e-3
[#####]100.00%
```

## SBM使用(Blast对象)

- Blast对象继承了Table对象
- 所以，Table对象中的所有方法在Blast对象都可以使用
- Gff3对象同理

# SBM使用

时刻不要忘了

`dir()`

`help()`

# SBM

```
>>> from sbm import *  
### Simplified Bioinformatics Module (SBM) 1.0  
### You gonna love using this module!^_^
```

# 知识分享

- 新浪博客：  
<http://blog.sina.com.cn/maxtoefl>

## 生物信息(2)

- 生物信息学在线作图工具收集帖（通... 2018-11-21 10:02 [编辑] 更多▼
- BLAST中，与HSP和query\_coverage(q...  2018-11-17 13:40 [编辑] 更多▼

## python编程(1)

- python编程技巧帖（通过评论，持续... 2018-11-28 09:34 [编辑] 更多▼