**实践任务**

lesson 1

test.fq.gz这个文件是Illumina测序数据的一个例子，以fastq格式存储。

**任务是：**1）网上查找资料了解fastq格式；

2）统计一下这批数据产量（即总共产生了多少reads和多少碱基）；

3）统计一下这批数据中A、T、C、G的总个数分别是多少。

echo -ne "A\nT\nG\nC\nN\n"|while read p;do zcat ../00.data/test.fq.gz |sed -n '2~4p'|grep -o ${p}|wc -l|while read s;do echo -ne "${p}:\t${s}\n";done;done

Fastq format:  ＃学习一下fastq格式，这是测序数据最常用的存储格式。

<http://en.wikipedia.org/wiki/FASTQ_format>

lesson 2

cm.fa.gz这个文件一种学名叫*Camponotus floridanus*的蚂蚁的基因组序列，以fasta格式存储。

**任务是：**1）网上查找资料了解fasta格式；

2）计算基因组中每条scaffold的长度，输出格式为第一列是scaffold ID，第二列为该scaffold长度；

3)统计该种蚂蚁的基因组大小是多少Mb；

4）计算该基因组的CG含量、5)统计该基因组中酶切位点CCGGTCGACCGG的数量。

lesson 3

cflo\_v3.3.gff.gz这个文件*Camponotus floridanus蚂蚁*的蛋白编码基因的注释文件，以Gff3格式存储。

**任务是：**1）网上查找资料了解Gff3格式；

2）统计该蚂蚁的基因总个数；

3）计算每一个基因的CDS的总长，输出格式为第一列是基因ID，第二列是该基因的CDS总长度。

lesson 4

**背景：**真核生物的基因包括了外显子和内含子，但转录成RNA后，内含子往往会被切除，所有外显子的序列首尾相接，形成能够编码蛋白质的成熟mRNA序列。然而，一条完整的mRNA序列，并不是全部序列都能翻译成蛋白质，只有中间的CDS区(也叫Open reading frame、Protein coding region)才能翻译，CDS区两侧分别是5’UTR和3’UTR。真核基因的结构以及对应的mRNA结构，可以参考以下两个链接中的示意图：

<https://en.wikipedia.org/wiki/Gene>    ＃Structure and function部分的第一个图。

**思考题：**5’UTR、3’UTR、CDS跟exon的关系是什么。

<https://en.wikipedia.org/wiki/Messenger_RNA>    ＃Structure部分的图。

**思考题：**5’Cap和Poly-A tail是不是由基因组编码的。

而我们平常做基因注释，往往只能把每个基因的CDS区给注释出来(见之前给大家的GFF文件出现的大量CDS标签)。所以，只要把每个基因的CDS序列提取出来，然后首尾相接，即可用于翻译成蛋白质的序列。

**编程任务：**根据*Camponotus floridanus*蚂蚁的基因组fasta文件（cm.fa.gz）以及配套的基因注释gff文件（cflo\_v3.3.gff.gz）。

1)获取每个基因去除内含子后的CDS序列，以fasta格式输出；

2)根据已知的密码子表，把CDS翻译成蛋白质序列，以fasta格式输出。

密码子表：直接把以下密码子表(以二维哈希存储)拷贝到你的脚本中使用即可，例如 $CODE{standard}{GCA}的value是A；$CODE{standard}{TGC}的value是C。

这里展示的密码子表的脚本是perl语言的，需要你自己转换成python语言的脚本

## the standard codon table.

my %CODE = (

        "standard" =>

        {

        'GCA' => 'A', 'GCC' => 'A', 'GCG' => 'A', 'GCT' => 'A',                               # Alanine

        'TGC' => 'C', 'TGT' => 'C',                                                           # Cysteine

        'GAC' => 'D', 'GAT' => 'D',                                                           # Aspartic Acid

        'GAA' => 'E', 'GAG' => 'E',                                                           # Glutamic Acid

        'TTC' => 'F', 'TTT' => 'F',                                                           # Phenylalanine

        'GGA' => 'G', 'GGC' => 'G', 'GGG' => 'G', 'GGT' => 'G',                               # Glycine

        'CAC' => 'H', 'CAT' => 'H',                                                           # Histidine

        'ATA' => 'I', 'ATC' => 'I', 'ATT' => 'I',                                             # Isoleucine

        'AAA' => 'K', 'AAG' => 'K',                                                           # Lysine

        'CTA' => 'L', 'CTC' => 'L', 'CTG' => 'L', 'CTT' => 'L', 'TTA' => 'L', 'TTG' => 'L',   # Leucine

        'ATG' => 'M',                                                                         # Methionine

        'AAC' => 'N', 'AAT' => 'N',                                                           # Asparagine

        'CCA' => 'P', 'CCC' => 'P', 'CCG' => 'P', 'CCT' => 'P',                               # Proline

        'CAA' => 'Q', 'CAG' => 'Q',                                                           # Glutamine

        'CGA' => 'R', 'CGC' => 'R', 'CGG' => 'R', 'CGT' => 'R', 'AGA' => 'R', 'AGG' => 'R',   # Arginine

        'TCA' => 'S', 'TCC' => 'S', 'TCG' => 'S', 'TCT' => 'S', 'AGC' => 'S', 'AGT' => 'S',   # Serine

        'ACA' => 'T', 'ACC' => 'T', 'ACG' => 'T', 'ACT' => 'T',                               # Threonine

        'GTA' => 'V', 'GTC' => 'V', 'GTG' => 'V', 'GTT' => 'V',                               # Valine

        'TGG' => 'W',                                                                         # Tryptophan

        'TAC' => 'Y', 'TAT' => 'Y',                                                           # Tyrosine

        'TAA' => 'U', 'TAG' => 'U', 'TGA' => 'U'                                              # Stop

        }

);

编程过程中，可以人工核对以下两个基因，看是否得到了正确的结果：

>Cflo\_00164--NA

MNVHAPAEMVFRPTYWIRSARYFISGPGPDSSRTTHFYFDLIDAGRIGHA

PVTLLRTSLGDYTKYEQNNVVQQTKPPFFPNQRDPWAFVAEEERKSQESL

SRRNGLNKAPTGWHSATVPVTNDGVCAFYAMSGEQRAPNEGSTTVSAYGA

NWIEAVYDCDEGEQRRRTEEERDREIADRSEETSGDVNRYEPIAFITSVI

IEKSAGAISGHGKRKDNRRKLRSGIDRPDRLVDLSANVRFIDRSLVGLLK

RMLARGYTPGYVTN

>Cflo\_00187--FBpp0110217\_DROME

MSQPRRGPIHGSSPVPGHPSTTVSASSSSSSSHEIQTRGIPLPGMVPSRS

IHASPMHGNPIHGFVNPLGPGSLVHPPPLTHAAAASGISHHPPNVHQGTE

KILPPGSPYSFHGSPLRTPMSSGHGSPQDSPHVIRHRRDSEHSSGAVSRR

SSSEVVSPLPEGHPLGSPDSRGRRRSDYSPHQAFHKFLARVVNRKLVVGI

DGNPTDTPLKIGTSLTAAREKNRMTFLLERDKRPPNFDARYPYPSTPLRI

VSSAIRKRTEIRRRRRGHIVTQRSTGFFIKSDDAALISFKLFATSTVRTF

AVTESRNSARTLLSDSGGKTASEWAIQHFSSFEMGGTGGLKHPVQAVYYA

LCVPVHLTVTICSQRESLDPCGALVVMSYIMHEKSEHLNELRLLYISVCD

SSIIWGNFCNVFNKISSEVIYLAHWFRVTVILSTPFSILVSVLFDDMRSR

NITHIRCILKVTKLREHWDETNSKVMQRKTQLDMMLGDSQRYEAKRNEVE

VWLARMETRLERMRAVGHTADVLEAQLREQKYGLFDGICLSAQALNIYPK

FAGANIFRGRELFPFVGRIKESERFGIYLPGRFLSLGALCNFDWDQSVPI

GISEEKLRTIYRRKTLVEDLTLPTYQGSSAKEARLLLTRYEAGSFHAELH

QYKHHIELFNQLTQKLIAVYQQDDTTRVKKMTETINQRYNNLNTRQLCIS

RILAFASTPSRETLPRIKLQITDVILYRDRSVSSIINRGKLLHSAMNSLH

NFDRSLDKFLAWLSEAESSMEGLEAEADRLGGRRDQGALRRPQHQLKKSE

ITNVRLSSIDHLQSCDINKQPPERITRLSDLTLMGNFIQIRVIVKVRWKS

VKAFTDEVAIGECFVEVSKVTETRKSGKPSQPLSLSRFIGAERGRTIDKT

PPPLEYE

lesson 5(这一题需要到华大的集群上进行)

1、把上一lesson中得到的蛋白序列fasta文件(如名字叫Cflo.pep)，作为输入文件，编写一个脚本，把Cflo.pep切割成100个子文件，每个文件的序列条数尽量相等，然后输出到一个叫split的目录下，子文件名可以是Cflo.pep.1、Cflo.pep.2、……、Cflo.pep.100。该python脚本需要会自己判断split/目录是否已经存在，如果不存在则自动创建。

2、编写一个脚本(script.py)，用于判断一条蛋白序列的第一个氨基酸是否是M，如果一条蛋白序列的第一个氨基酸是M，就输出这条蛋白。编写第二个脚本(script2.py)，用于自动获取split/目录下的100个子文件，然后对每一个子文件生成一条执行第一个脚本的命令行，并把命令行(如python script1.py Cflo.pep.1 >Cflo.pep.1.out)输出到一个shell文件中(如Cflo.pep.1.sh)，从而得到100个子文件对应的100个shell文件。最后把这100个shell文件用qsub提交到计算节点中并行执行。

lesson 6(这一题需要到华大的集群上进行)

大家在lesson 4中已经生成得到了Camponotus floridanus蚂蚁全基因组的蛋白序列fasta文件，总共有17064条蛋白序列。现在，有另一种蚂蚁Acromyrmex echinatior全基因组的蛋白序列，文件如下：Aech\_v3.10.geneWithUTR.pep.gz

本次的任务是通过双向比对最好的方法(Reciprocal Best Hit, RBH)鉴定这两种蚂蚁的直系同源基因(ortholog)。RBH是鉴定物种间直系同源基因最传统也是最简单的方法，在进化分析中经常用到。（**思考**：什么是直系同源基因？什么是旁系同源基因？两者的差别是什么？）

**步骤：**

1、用Linux的cat命令把两种蚂蚁的蛋白序列fasta文件合并成一个fasta文件(如名字叫merge.pep)；

2、用blast软件做蛋白对蛋白的序列比对，query文件和subject文件都是merge.pep，相当于merge.pep自身比自身。输出格式设置为m8格式。为了提高比对的效率，要求把query文件切成100个子文件，每个子文件同时对subject文件做blast，100个比对任务提交到计算集群并行跑，都完成后把100个m8文件直接cat起来，作为总的m8文件。(**思考：**这跟直接用query文件比subject文件，结果有什么不一样？)

服务器上blast软件的路径在：/share/app/blast-2.2.26/bin/blastall

开始分析前，需要先了解一下blast比对的基本原理以及用法。

3、写一个python脚本，以m8文件作为输入，通过双向比对最好的原则鉴定两个物种的ortholog pairs。判定一个物种的某基因在另一个物种中的best hit，可以根据blast score值的高低，score越高越好。例如Cflo蚂蚁的A gene，在Aech蚂蚁中比上了10个hit (分别是A、B、C、…...)，通过score判断，best hit是Aech的B基因；Aech的B gene，在Cflo中比上了5个hit，通过score值判断，best hit是Cflo的A基因。则Cflo的A gene和Aech的B gene是一对直系同源基因。要求：尝试在本脚本中使用哈希数组。（Python中尝试使用字典完成）