pep_pattern.pl,搜索蛋白质序列模体的 Perl 脚本

李旭凯 彭良才 王令强

华中农业大学作物遗传改良国家重点实验室/华中农业大学生物质与生物能源中心/
华中农业大学植物科学技术学院,武汉 430070

摘要 用 Perl 语言编写了一个 pep_pattern. pl 脚本,可从一组相关序列中搜索非常相似的序列片段,通过匹配所有可能的氨基酸片段的排列,统计每个匹配模体在序列中的出现频率和位置,搜索蛋白质序列中的 2～4 个多肽的模体。

关键词 pep_pattern. pl; Perl 脚本; 蛋白质序列; 模体; 排列

中图分类号 Q 811.4; R 857.3 **文献标识码** A **文章编号** 1000-2421(2014)04-0001-06

模体(motif)是 DNA 或蛋白质序列中局部的保守多肽区域,或者是反复发生在一组相关的 DNA 或蛋白质序列中共有的一小段序列模式,也译为基序(pattern)[1]。蛋白质具有结构域和生物功能位点。功能相近的蛋白质或同类蛋白质家族成员表现出该功能所必需的模体,这个模体不仅反映蛋白质的功能位点,而且也作为蛋白质家族的识别信号。一个蛋白质家族绝大多数成员共同拥有的模体很可能是该家族组成结构和执行功能的重要部分,这些模体位点的探测和定位是蛋白质研究的重要方面。从蛋白序列中识别出某个蛋白质家族共同的模体就能够描述这个蛋白质家族的特征,可以利用这些模体特征来进行搜索和发掘蛋白家族有生物学意义的新成员[2],所以,模体的识别方法就显得很重要。对于一个给定的某个蛋白质家族,用一些模体识别方法和软件就能识别出这些序列保守的模体,但是在识别结果中可能存在因随机匹配而产生的假模体。一般从统计学意义上对模体的真假进行判断以确定其是否具有生物意义。虽然统计学意义并不完全与生物学意义等同,但是在识别的模体结果中,具有统计学意义的模体其具有生物学意义的可能性越大[2]。目前模体的识别方法分为两类:统计学方法,如 Gibbs Sampling[3]、MEME[4] 和 MMER[5] 等;确定性(deterministic)方法,如 Pratt[6]、TEIRESIAS[7]、SPLASH[8]、SPAT[9] 等。这些方法中,统计学的模体识别方法应用更加广泛。

但是对于氨基酸数目少(2～4)的多肽,通过各种生物信息学方法识别出的模体,目前没有很好的办法得到预期的结果;由于序列模式复杂,很难预测氨基酸序列的模式。pep_pattern. pl 提供了一种方便的 Perl 脚本处理生物蛋白序列,正好解决了这个问题。笔者用 Perl 语言编写了 1 个脚本——pep_pattern. pl,用于从一组相关序列中搜索其中非常相似的序列片段,通过匹配所有可能的氨基酸片段的排列,统计每个匹配模体在序列中出现的频率和位置(如果在同一序列中检索到同一模体多次出现,则以空格间隔开),搜索蛋白质序列中的 2～4 个氨基酸多肽的模体。

1 材料与方法

1.1 在 Windows 操作系统下安装 Perl

Perl 是“practical extraction and report language”的缩写,它是由 Larry Wall 设计编写的,并由他不断更新和维护,用于在 UNIX 环境下编程。Perl 可以跨系统平台运行,可用于 UNIX、Linux、MAC 和 Windows 等系统环境下编程和运行,并由

收稿日期: 2013-05-27
基金项目: 国家自然科学基金项目(30900890,31171524)、全国博士后基金项目(20070420917)、湖北省自然科学基金项目(2009CDB324)和中央高校基本科研业务费专项(2011PY047)
李旭凯,博士研究生. 研究方向: 生物信息学. E-mail: specterae@163.com
通信作者: 王令强,博士,副教授. 研究方向: 能源作物分子生物学和遗传育种. E-mail: lqwang@mail.hzau.edu.cn

CPAN不断更新和维护[10]。CPAN[11](comprehensive perl archive network)中译为“Perl综合典藏网”。它包含了极多用Perl写成的软件和其文件。Perl借鉴了C、sed、awk、shell脚本语言以及很多其他编程语言的特性和优点。其中最重要的特性就是它内部集成了正则表达式(regular expression)的功能。

pep_pattern.pl是一个Perl脚本程序,运行Perl脚本时要求在电脑上预先安装有编译并运行的Perl解释器。在绝大部分类UNIX系统(包括Linux和Mac OS)中,Perl解释器是随系统安装的,可在终端的命令行输入命令“perl-v”来查看Perl的版本。而对于大多数人使用的Windows系统则有“Strawberry Perl”和“ActivePerl”2种版本可用。对于熟悉Linux系统环境或者服务器下Perl编程的人来说,使用Strawberry Perl软件会更加习惯。Strawberry Perl[12]是Windows下的“the core Windows distribution of Perl”的一个版本,它尽可能的在Windows系统平台上保持了Perl在Unix上的风格,最大程度地保证了Perl的可移植性。因此,在CPAN上安装时,在Strawberry Perl下能很容易编译通过。Strawberry Perl目前最新版本为Strawberry Perl 5.18.2.2,32位系统的下载地址为“http://strawberryperl.com/download/5.18.2.2/strawberry-perl-5.18.2.2-32bit.msi”。双击“msi”文件并按照其安装提示步骤即可完成Perl在Windows系统中的安装[13]。

1.2 数据的格式

pep_pattern.pl必须输入1个FASTA格式的序列文件。序列文件的第1行是由“>”开头来进行注释,其后第1个单词为序列名,随后是一些说明性文字。从第2行开始为序列本身,只允许使用既定的氨基酸编码符号,用单字符号大写字母表示(表1),直到下一个注释符号截止。然后将这个FASTA格式的序列文件另存为“文本文件(*.txt)”,将这个序列文件与pep_pattern.pl脚本文件存放在同一个目录下(C:\strawberry\perl\bin\)。

1.3 pep_pattern.pl的运行

将pep_pattern.pl文件存放到“C:\strawberry\perl\bin\”目录,然后点击任务栏上的“开始—运行”在运行框输入“cmd”命令即可打开“cmd命令提示符”,在cmd窗口中键入“cdC:\strawberry\perl\bin”后回车,使当前目录转到含Perl命令执行程序的目录下。在“C:\strawberry\perl\bin>”的提示符下键入“perl pep_pattern.pl Protein_Seq.txt”,回车即运行了。其中“Protein_Seq.txt”是蛋白质FASTA格式序列文件名,输出结果pep_pattern.pl会自动生成文件“Result.txt”,并存放入其中。键入“perl pep_pattern.pl -h”可获得帮助信息。

表1 FASTA格式支持的氨基酸代码

Table 1 The accepted amino acid codes

单字符号 One-letter symbol	意义 Meaning	三字符号 Three-letter symbol	中文名称 Name
A	Alanine	Ala	丙氨酸
C	Cysteine	Cys	半胱氨酸
D	Aspartic acid	Asp	天冬氨酸
E	Glutamic acid	Glu	谷氨酸
F	Phenylalanine	Phe	苯丙氨酸
G	Glycine	Gly	甘氨酸
H	Histidine	His	组氨酸
I	Isoleucine	Ile	异亮氨酸
K	Lysine	Lys	赖氨酸
L	Leucine	Leu	亮氨酸
M	Methionine	Met	甲硫氨酸
N	Asparagine	Asn	天冬酰胺
P	Proline	Pro	脯氨酸
Q	Glutamine	Gln	谷氨酰胺
R	Arginine	Arg	精氨酸
S	Serine	Ser	丝氨酸
T	Threonine	Thr	苏氨酸
V	Valine	Val	缬氨酸
W	Tryptophan	Trp	色氨酸
Y	Tyrosine	Tyr	酪氨酸

运行“perl pep_pattern.pl Protein_Seq.txt”后,程序会提示“Please input the optimum width of motif with the limit:”即输入设定功能域长度值的最大值,然后回车,如果没有输入程序默认输入“4”,如图1。pep_pattern.pl会搜寻功能域长度从2开始到输入的功能域长度数字之间所有长度的模体,比如输入“4”,pep_pattern.pl会搜寻“2、3、4”这些功能域长度的模体。示例中表示要从数据“Protein_Seq.txt”中搜寻2~3个氨基酸多肽的模体,结果输出到“Result.txt”文件。如果输入大于“4”的参数,程序也可以继续运行出对应功能域长度模体的结果,但是运行时间将会很长,这时候建议使用MEME(http://meme.nbcr.net/meme/cgi-bin/meme.cgi),能够更快得到结果。

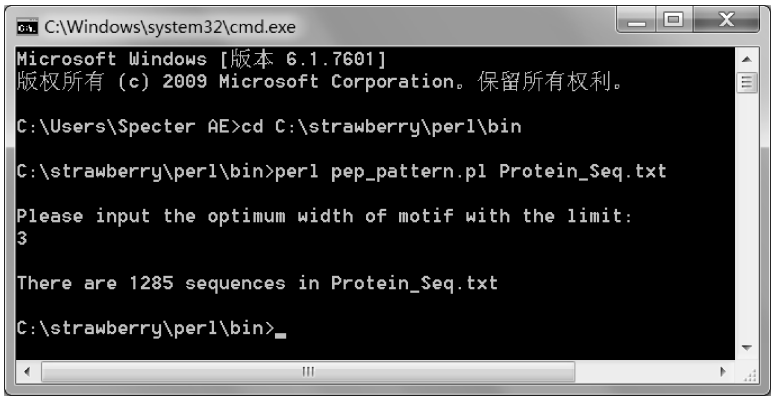

图 1　pep_pattern.pl 的运行界面示意图

Fig. 1　Running window of pep_pattern.pl

脚本程序运行结束后，在“C:\strawberry\perl\bin”目录下可找到输出的结果文件“Result.txt”。结果文件分 5 列，依次表示：序列名、序列氨基酸总数、模体位置、模体多肽、出现次数。

2　结果与分析

2.1　pep_pattern.pl 的原理和特点

pep_pattern.pl 提供了一种方便的 Perl 脚本处理生物蛋白序列。pep_pattern.pl 通过枚举匹配所有可能氨基酸多肽的排列，统计每个匹配到模体在序列中出现的频率和位置（如果在同一序列中检索到同一模体多次出现，则以空格间隔开），搜索蛋白质序列中的 2～4 个多肽的模体。pep_pattern.pl 必须输入一个 FASTA 格式的序列文件，序列文件的第 1 行是由“>”开头来进行注释，其后第 1 个单词为序列名，然后是一些说明性的文字。从第 2 行开始为序列本身，只允许使用既定的氨基酸编码符号，用单字符号大写字母表示。其结果输出文档为文本文档“Result.txt”。

运行 pep_pattern.pl 操作简单、快捷、参数设置少。可以全面地搜寻一组蛋白质序列中 2～4 个氨基酸多肽的模体。

2.2　pep_pattern.pl 的源代码

pep_pattern.pl 的源代码如下，可以将以下代码复制到记事本中，保存一个文件名（如：pep_pattern.txt），然后重命名为 pep_pattern.pl，之后就可以按照本文“1.3”的运行方法使用。

代码如下：

```
#! /usr/bin/perl
### Print help ###
my $PrintHelp=qq(
USAGE:
    perl pep_pattern.pl <Infile>
AUTHOR:
    XukaiLi (specterae\@163.com) 2014/04
DESCRIPTION:
    Thisperl script was written for manipulating regular expressions
    describing amino acid sequence pattern or motif. Patterns can
    be quite complex and it is often difficult to generate amino
    acid sequence pattern. The pep_pattern.pl addresses this
    problem, providing a convenient set of tools for working
    with biological sequence motif.
    Written by Xukai Li and test under the perl enviroment 5.12.4.
DATA STRUCTURES:
    The sequence format for pep_pattern.pl must be a protein
    sequence and must in Fasta format. Sequences start with a
    header line followed by sequence lines. A header line has
    the character ">" in position one, followed by a unique name.
    After the header line come the actual sequence lines
```

(in capital). Spaces and blank lines are ignored.

The input file should use only plain, unformatted text.

EXAMPLE:

perl pep_pattern. pl Protein_Seq. fasta

```
\n);
die( $PrintHelp) if( $ARGV[0] = ~/-[hH]
+/);
### Main ###
    die " Please input command line arguments of protein sequence file. \nUsage:perl pep_pattern. pl Protein_Seq. fasta\n\nTo show brief help usage, do \"pep_pattern. pl -h\"\n" unless $ARGV[0];
    print"\nPlease input the optimum width of motif with the limit:\n";
my $width = <STDIN>;
$width =~/\d+/? ($width):($width=4);
$amino_acid =['A','C','D','E','F','G','H','I','K','L','M','N','P','Q','R','S','T','V','W','Y'];
&create( $amino_acid);
sub create {
    my ( $list) = @_;
    my $str_list = [''];
    for ( $x=0; $x< $width; $x++) {
    $str_list = create_list( $str_list, $list);
    }
    return $str_list;
    }
    sub create_list {
    my ( $ref_str_list, $ref_array) = @_;
    my @return_array;
    foreach my $str (@{ $ref_str_list}) {
    foreach my $element (@{ $ref_array}) {
    push @return_array,"${str}$element";
    }
}
    if ( $return_array[20]=~/\w+/) {
    open(IN, $ARGV[0])||die "$!";
    open(OUT,">Result. txt")||die "$!";
    local $/ = ">";
    my %hash =();
    $no_heads =0;
    while(<IN>){
    chomp;
    my ( $head, $seq) = split(/\n/, $_,2);
    next unless( $head && $seq);
    $no_heads ++;
    $seq=~s/\s+//g;
    $seq=~s/[*-]//g;
    $head=~s/\s+$//;
    foreach $motif (@return_array) {
        if( $seq =~ /$motif/){
          $hash{ $motif}++;
          push (@Motif, $motif);
          }
        }
      }
print "\nThere are $no_heads sequences in $ARGV[0]\n";
open(DATA, $ARGV[0]);
while(<DATA>){
chomp;
    my ( $head, $seq) = split(/\n/, $_,2);
    next unless( $head && $seq);
    $seq=~s/\s+//g;
    $seq=~s/[*-]//g;
    $head=~s/\s+$//;
    foreach my $key ( sort { $hash{ $b} <=> $hash{ $a} } keys % hash ) {
          my $value = $hash{ $key};
            if( $seq =~ /$key/ and $value/$no_heads >0.5){
            my ( $position, $now ) = ( 0,-1 );
            until ( $position == -1 ) {
          $position = index( $seq, $key, $now + 1 );
          $now = $position;
           push (@ position, $position) unless $position < 0;
          $position{ $head. $key}. = " $position " unless $position < 0;
}
    my $seq_length = length ( $seq);
    print OUT ">$head\t$seq_length\t$position{ $head. $key}\t$key\t$value\n";
```

```
                }
            }
        }
    }
return \@return_array;
}
close(IN);
close(OUT);
```

3 讨　论

在准备数据表时，pep_pattern.pl 必须输入一个 FASTA 格式的序列文件，不支持其他序列格式，请使用前先把格式转换为 FASTA 格式。

这里提供一个把任意格式（AB1、ABI、ALF、CTF、EMBL、EXP、Fasta、Fastq、GCG、GenBank、PIR、PLN、SCF、ZTR、ace、game、locuslink、phd、qual、raw、swiss）转换为任意格式的 perl 程序——Reformat.pl。读者可以将下文代码其拷贝到记事本中，保存为一个文件名（如：Reformat.txt），然后重命名为 Reformat.pl，之后按照类似的运行方法使用。值得注意的是有一句代码“use Bio :: SeqIO;”，要求读者在“cmd”命令行输入“cpan”，随后在“cpan>”提示符后输入“install Bio :: SeqIO”回车，见图 2。之后就可以顺利运行 Reformat.pl 了。

Reformat.pl 代码如下：

```
#! /usr/bin/perl -w
#用法:"perl Reformat.pl format1 format2< 要转的文件名>output"
use Bio :: SeqIO;
$format1 = shift;
$format2 = shift || die "Usage: reformat format1 format2 < input >output";
```

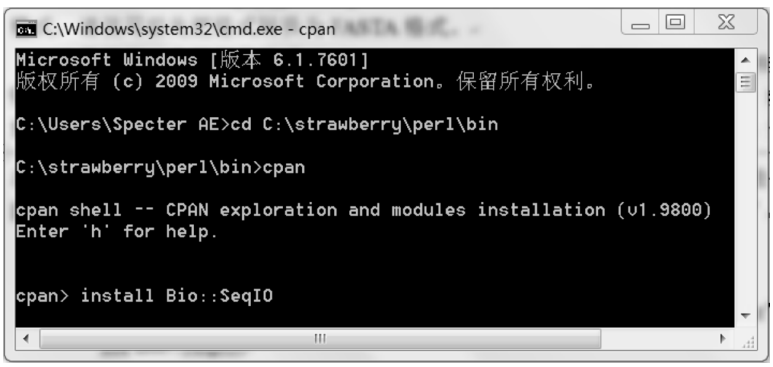

图 2　cpan 的运行界面示意图

Fig. 2　Running window of cpan

```
$in = Bio :: SeqIO->newFh(-format =>$format1,-fh =>\*ARGV );
$out = Bio :: SeqIO->newFh(-format =>$format2 );
$out $_ while <$in>;
```

pep_pattern.pl 程序的不足是设置的参数只有 1 个，即输入设定功能域长度值的最大值，如果没有输入程序默认输入“4”。pep_pattern.pl 会搜寻功能域长度从 2 开始到输入的功能域长度数字之间所有长度的模体，搜寻尽可能多的信息。用户还可以对该脚本进行修改，对编程有困难的用户可以和笔者联系。本研究没有对得到的结果进行评估，只是对每个模体出现的次数做了计数，不能确定其是否具有生物学意义。但是在识别结果中具有统计学意义的模体将更可能具有生物学意义。

参　考　文　献

[1] GIBAS C, JAMBECK P. Developing bioinformatics computer skills[M]. Sebastopol: O'Reilly Media, 2001: 180-188.

[2] 杜春娟，朱云平，贺福初，等. 蛋白质家族模体(motif)的评价策略[J]. 北京生物医学工程，2005，24(2)：97-102.

[3] LAWRENCE C E, ALTSCHUL S F, BOGUSKI M S, et al. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment [J]. Science, 1993, 226: 208-214.

[4] BAILEY T L, ELKAN C. Fitting a mixture model by expectation maximization to discover motifs in biopolymers: proceedings of the Second International Conference on Intelligent Systems for Molecular Biology [C]. Menlo Park: AAAI Press, 1994.

[5] EDDY S R. Profile hidden Markov models [J]. Bioinformatics, 1998, 14(9): 755-763.

[6] INGE J, COLLINS J E, HIGGINS D G. Finding flexible patterns in unaligned protein sequences [J]. Protein Science,

1995,4(8):1587-1595.
[7] LSIDORE R,ARIS F. Combinatorial pattern discovery in biological sequences:the TEIRESIAS algorithm [J]. Bioinformatics,1998,14 (1):55-67.
[8] ANDREA C. SPLASH:structural pattern localization analysis by sequential histograms[J]. Bioinformatics,2000,16(4):341-357.
[9] REECE K H,AJAY K R,STOLOVITZK G,et al. Systematic and fully automated identification of protein sequence patterns [J]. Journal of Computational Biology,2000,7(3/4):585-600.
[10] SCHWARTZ R L,TOM P,BRIAN D F. Learning Perl[M]. 5th ed. Sebastopol:O'Reilly Media,2008:1-17.
[11] The Comprehensive Perl archive network (CPAN) [CP/OL]. [2012-10-09]. http://www.cpan.org.
[12] KENNEDY A. Strawberry Perl for Windows [CP/OL]. [2013-03-12]. http://strawberryperl.com.
[13] 李旭凯,郭凯,彭良才,等. ChooseMaterials.pl,控制变量挑选实验材料的Perl脚本[J]. 生物信息学,2013,11(3):186-191.

pep_pattern.pl,a Perl script for searching motifs in a group of related DNA/protein sequences

LI Xu-kai PENG Liang-cai WANG Ling-qiang

National Key Laboratory of Crop Genetic Improvement/Biomass and Bioenergy Research Center/College of Plant Sciences and Technology,Huazhong Agricultural University,Wuhan 430070,*China*

Abstract A motif is a sequence pattern occurring repeatedly in a group of related DNA or protein sequences,and is an important concept for describing the common structure and function shared by the members of a protein family. However,the motif can be quite complex and is often difficult to predict the pattern of amino acid sequence. To get the desired results of the short motifs (2-4 polypeptides) derived from various bioinformatics is still a difficult task. The pep_pattern.pl can be used to solve this problem and provide a convenient set of Perl script for working with biological sequence motif. A Perl script pep_pattern.pl was written for searching very similar amino acid sequence pattern or motif in a group of related protein sequences by matching all the possible amino acids fragments permutation and counting frequency and position of each motif matched in sequence.

Key words pep_pattern.pl; Perl script; protein sequences; motif; permutation

(责任编辑:张志钰)