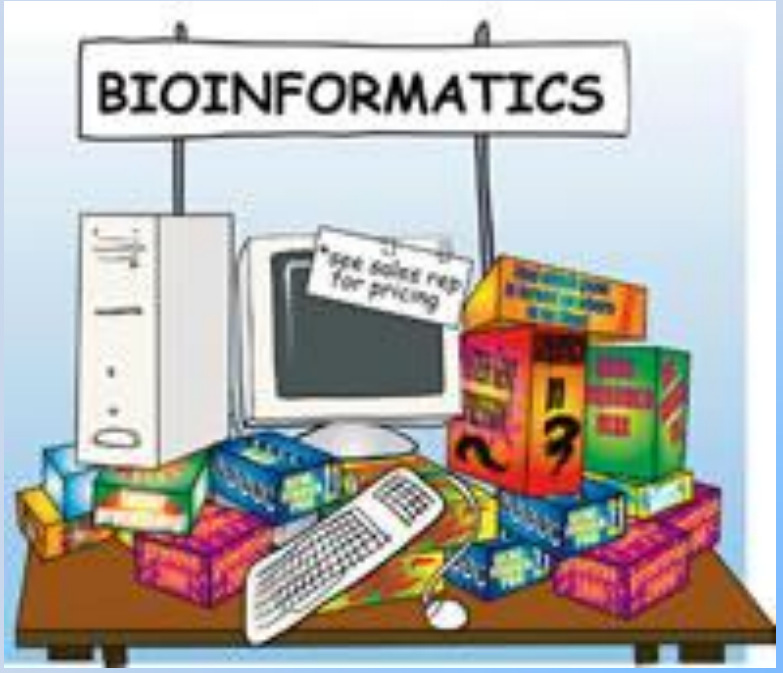
INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior Agrária

CURSO DE BIOINFORMÁTICA

INSTALAÇÃO DO BIOPERL E EXECUÇÃO DE SCRIPTS

©2009 Sérgio Deusdado

A bioinformática e a programação

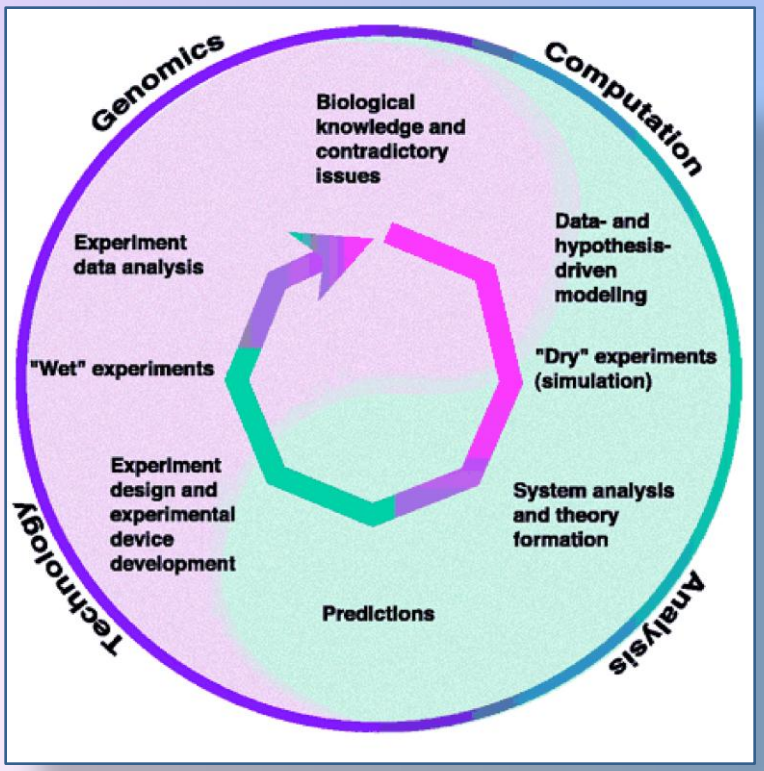


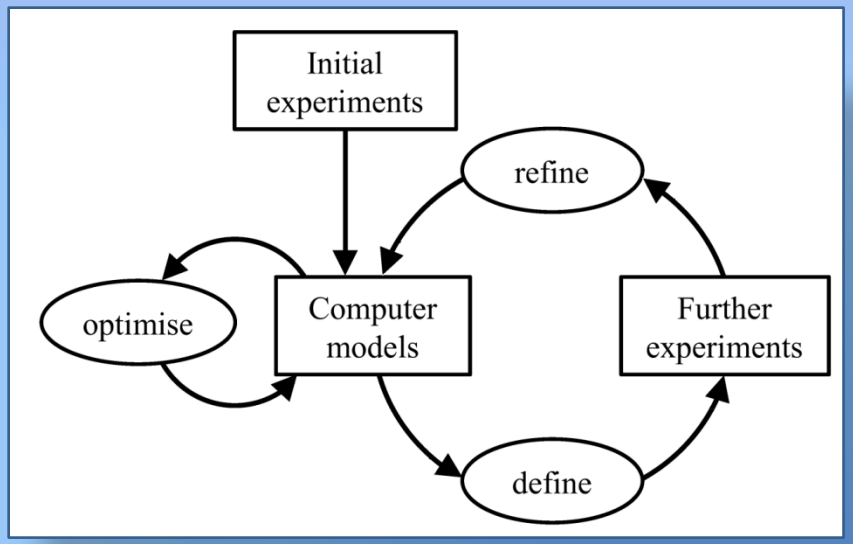
Programar ou não programar, eis a questão?

© Sérgio Deusdado

Desenvolvimento de software bioinformático

Etapas de desenvolvimento e refinamento





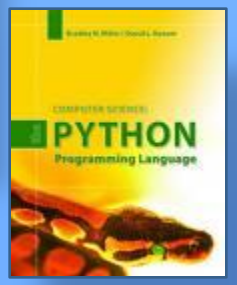
© Sérgio Deusdado

Quais as linguagens de programação mais usadas em bioinformática?

São adaptações de linguagens de programação de uso geral, que receberam um conjunto de novas funcionalidades para tratamento de dados biológicos. As mais usadas exibem propriedades adequadas aos requisitos da bioinformática, como sejam: prototipagem rápida, a facilidade de manipulação de *strings*, a gestão da memória transparente, a portabilidade, etc.

Exemplos (domínio público):

- **BioPerl** (desde 1995)
http://www.bioperl.org/
- **BioJava**
http://biojava.org
- **BioPhyton**
http://biophyton.org

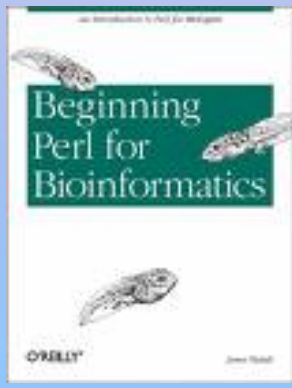


© Sérgio Deusdado

Quais as razões para escolher a linguagem Perl - BioPerl?

Vantagens gerais:

-Velocidade de execução dos programas implementados;
-Prototipagem rápida;
-Poucas linhas de código para grandes resultados;
-É uma linguagem de domínio público (gratuita) e existe código-fonte disponível;
- Orientada a objectos.



Não é necessário(em oposição a outras linguagens):

-Declarar variáveis antes de as usar;
-Definir previamente o tipo de dados a incluir na variável;
-Reservar espaço em memória para alocar os conteúdos da variáveis.

© Sérgio Deusdado

Variáveis em Perl: Tópicos

Uma variável é basicamente um contentor de dados, simples ou estruturado.

Em Perl existem três tipos básicos de variáveis:

- Escalares(inteiros, decimais, texto, etc.);
- Listas;
- Tabelas (arrays) associativas (tabelas de hash).

Prefixos para distinção de variáveis:

($)- Escalares;

(@)- Listas;

(%)- Tabelas (arrays) associativas.

© Sérgio Deusdado

Variáveis escalares em Perl

As referências às variáveis iniciam-se sempre com "$", quer para acessos quer para atribuições

Para escalares:

```
$x = 1;
$x = "Hello World!";
$x = $y;
```

Para tabelas (arrays) de escalares:

```
$a[1] = 0;
$a[1] = $b[1];
```

© Sérgio Deusdado

Variáveis de tipo lista em Perl

As listas são precedidas pelo símbolo arroba "@":

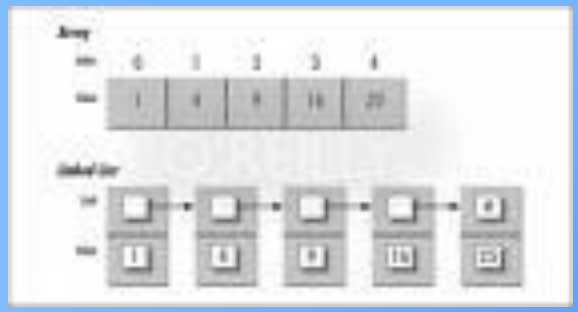
```
@sequencia = (1, 2, 3, 4, 5);
@frutos = ("morango", "banana", "laranja");
@copia_sequencia = @sequencia;
```

Uma lista pode ser entendida como um vector (tabela unidimensional)

Índices inteiros podem ser utilizados para referenciar os elementos que integram as listas

Para imprimir um elemento de uma lista:

```
print $frutos[2];
```



© Sérgio Deusdado

Variáveis de tipo arrays associativos em Perl

Estas variáveis são diferenciadas pela utilização do símbolo percentagem "%"

São destinadas a conter associações entre arrays de escalares, pares chave/valor constituindo tabelas de hash

Exemplo:

```
$fred{"a"} = "aaa";
$fred{"b"} = "bbb";
$fred{6} = "cc";
$fred{1} = 2;
```

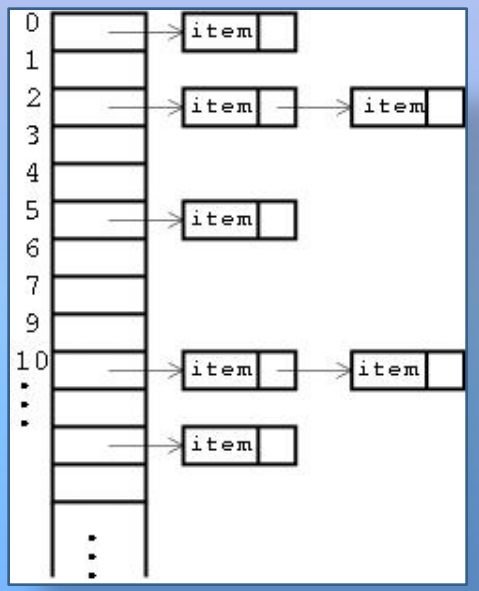


Tabela de hash

É equivalente, em Perl, a escrever o seguinte código:

```
%fred = ("a", "aaa", "b", "bbb", 6, "cc", 1, 2);
```

© Sérgio Deusdado

Programação modular em Perl

Para utilização de programação modular recorre-se à codificação dos sub-programas usando a palavra reservada "sub", seguindo a delimitação do código do sub-programa entre chavetas.

Exemplo:

```
Sub meusubprograma1
{
    # Código Perl do módulo.
}
```

Antecede os comentários

Os módulos são reutilizáveis e, se necessário, de invocação recursiva. A invocação é precedida do símbolo "&"

Exemplo:

```
&meusubprograma1;
```

© Sérgio Deusdado

Módulos existentes no Perl e seus componentes

O Perl está estruturado em módulos de funções pré-programadas (primitivas). O programador normalmente reutiliza essas primitivas para a composição e implementação dos seus programas.

Exemplos de módulos:
Math; String; Bio

Na descriminação dos componentes , estes separam-se do módulo (ou sub-módulo) pelos símbolos"::"

Exemplos de módulos e do componente específico:

Math::Complex
Math::Approx
String::BitCount
String::Approx
Bio::Tools::pSW

Efectua o alinhamento de seqs (PD) pelo algoritmo de Smith-Waterman

© Sérgio Deusdado

O que é o BioPerl?

O **BioPerl** não é uma nova linguagem, é apenas uma colecção de módulos que implementam funções necessários para o tratamentos de dados bioinformáticos

Facilita o **desenvolvimento de aplicações bioinformáticas ou simplesmente de *scripts*** (trechos que implementam acções simples e recorrentes) para utilização em meios de investigação bioinformática

O **módulo a adicionar ao Perl chama-se Bio**, e incluí, em número crescente, componentes para as mais diversas necessidades de análise e tratamento de dados biológicos

© Sérgio Deusdado

Porquê ensinar BioPerl a alunos de Bioinformática?

O (**Bio)Perl é uma linguagem vocacionada para a manipulação e processamento de cadeias de símbolos**, revelando utilidade em quase todas as operações básicas de análise de sequências biológicas

Usando os módulos disponíveis (gratuitamente) do BioPerl, **qualquer pessoa com um mínimo de formação pode construir pequenos programas (e *scripts*) em BioPerl**, simplesmente pela invocação dessas funções pré-concebidas

No dia a dia do laboratório de bioinformática as necessidades pontuais nem sempre têm resposta nas aplicações bioinformáticas mais implantadas, pelo que **a produtividade do investigador pode ser melhorada se detiver competências em BioPerl**. O diálogo com os programadores fica mais facilitado.

© Sérgio Deusdado

Instalação do BioPerl

(*O software usado é na totalidade de domínio público e gratuito*)

1 º Fazer o download do Perl através do site do Perl ActiveState em: http://activestate.com/Products/ActivePerl
2º Fazer a instalação do Perl no sistema
3º Abrir o PPM (Perl Package Manager), que está no grupo de programas instalados do Perl acessível via menu Iniciar/Programas/ActivePerl
4º No PPM, seleccionar no menu Edit a opção Preferences… e aceder ao separador Repositories
- Um por um, adicionar os seguintes módulos (para a versão 5.10):
    BioPerl-Regular Releases http://bioperl.org/DIST
    Kobes http://cpan.uwinnipeg.ca/PPMPackages/10xx/
    Bribes http://www.Bribes.org/perl/ppm

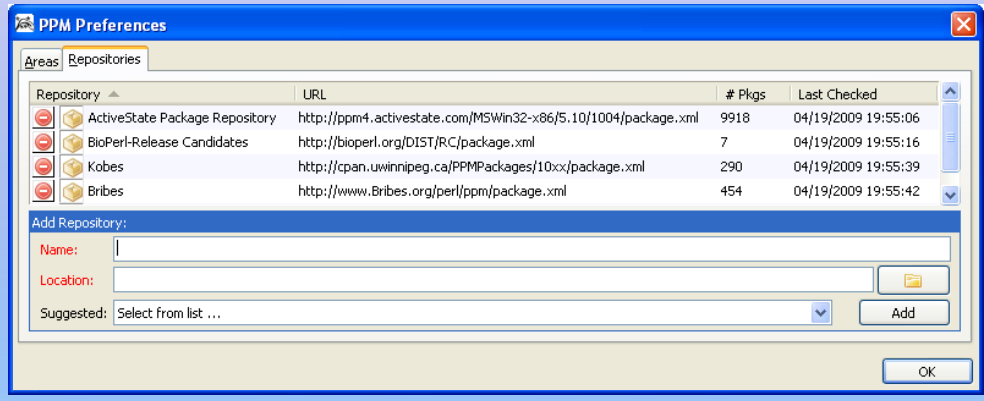
© Sérgio Deusdado

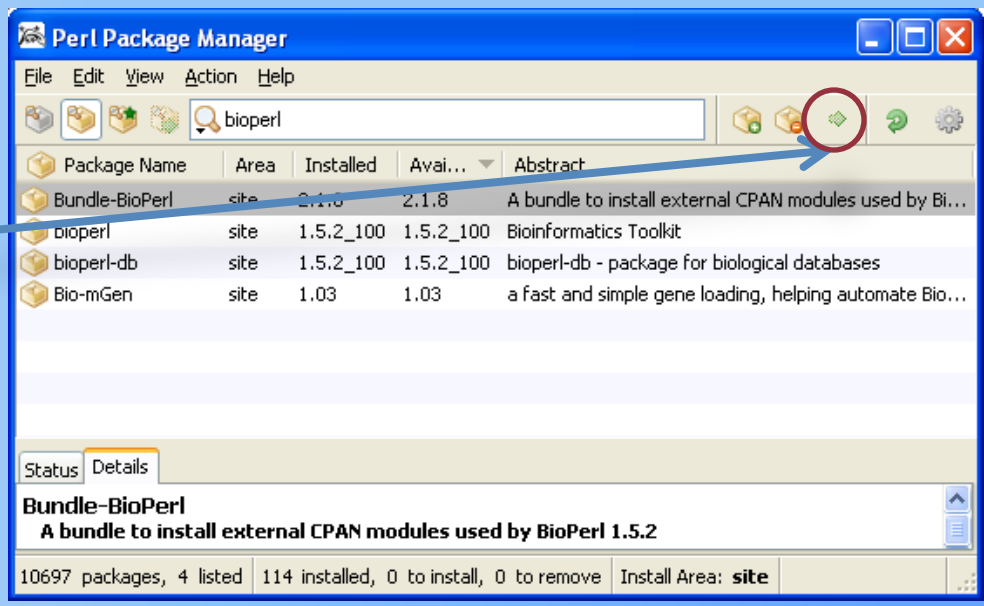
Instalação do BioPerl (cont.)

5º Clicar em OK na janela PPM Preferences, verificando se os módulos foram adicionados (ver imagem à direita)

6º No PPM pesquisar por "bioperl" e marcar o módulo para instalação (botão dir. do rato no módulo e seleccionar a opção)

7º Carregar no botão ➡ para executar as operações de instalação marcadas e aguardar que finalize a instalação (resultado igual à imagem)





© Sérgio Deusdado

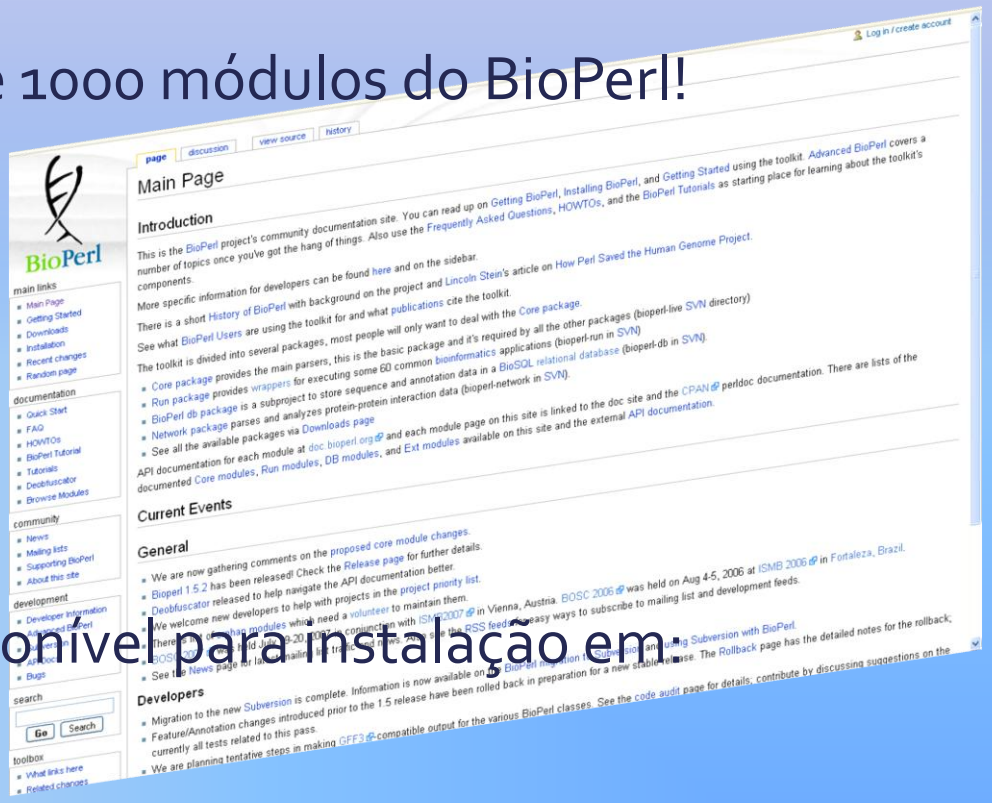
Utilizar os módulos existentes no BioPerl

Os módulos do BioPerl invocam-se por Bio::XXX...

Actualmente existem mais de 1000 módulos do BioPerl!

Exemplos:

- **Bio::Align**
- **Bio::SeqIO**
- **Bio::Annotation**
- **Bio::Assembly**
- **Bio::Biblio**



O BioPerl (módulos) está disponível para instalação em:

http://www.bioperl.org/

com documentação e exemplos, para facilitar a aprendizagem
(Clicar em BioPerl 1.4 Module Documentation)

© Sérgio Deusdado

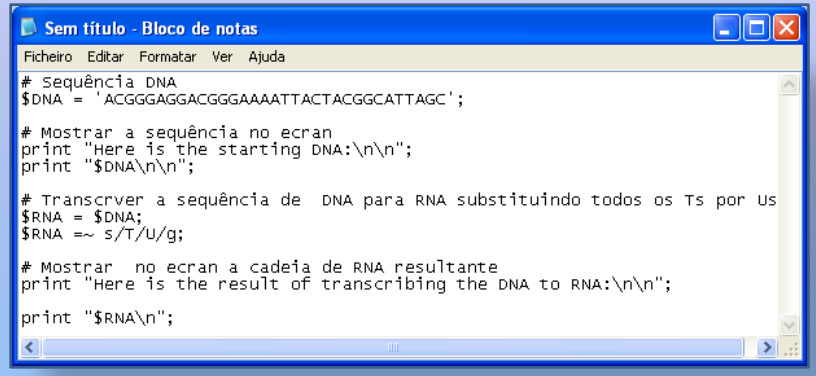
Começar a programar em BioPerl

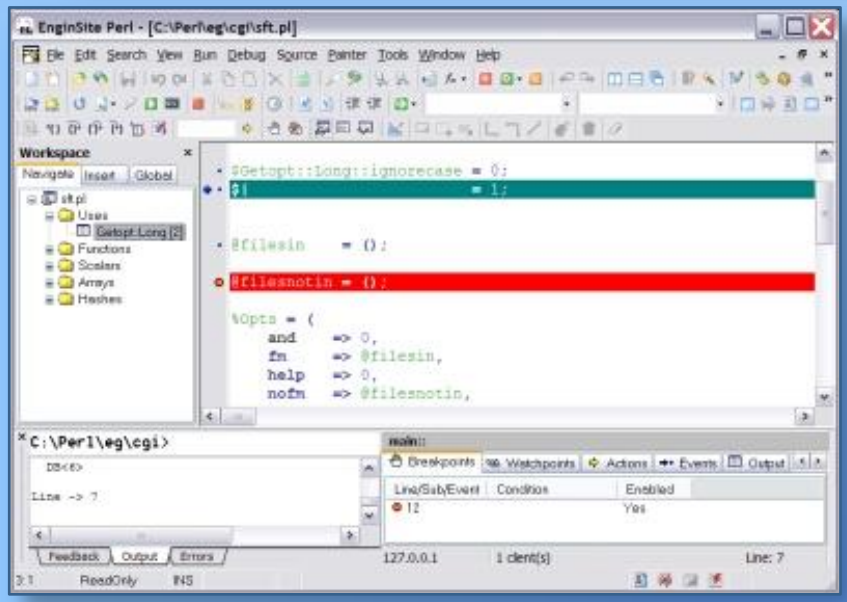
Escrever o código-fonte

Na ausência de melhor editor, o *Notepad* (Bloco de Notas) pode servir perfeitamente.

Alternativa: Um IDE (ambiente integrado de desenvolvimento)

http://www.enginsite.com/Perl.htm





Para executar ou depurar o programa

Guardar o código num ficheiro com a extensão "pl" e executar na linha de comandos:

>perl nomeprograma.pl

© Sérgio Deusdado

Começar a programar em BioPerl

Um exemplo básico , sem necessidade de módulos BioPerl

```
# Sequência DNA
$DNA = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';

# Mostrar a sequência no ecran
print "Aqui esta a sequencia de DNA:\n\n";
print "$DNA\n\n";

# Transcrver a sequência de  DNA para RNA substituindo todos os Ts por Us.
$RNA = $DNA;
$RNA =~ s/T/U/g;

# Mostrar  no ecran a cadeia de RNA resultante
print "O resultado da transcrição de DNA para RNA:\n\n";

print "$RNA\n";
```

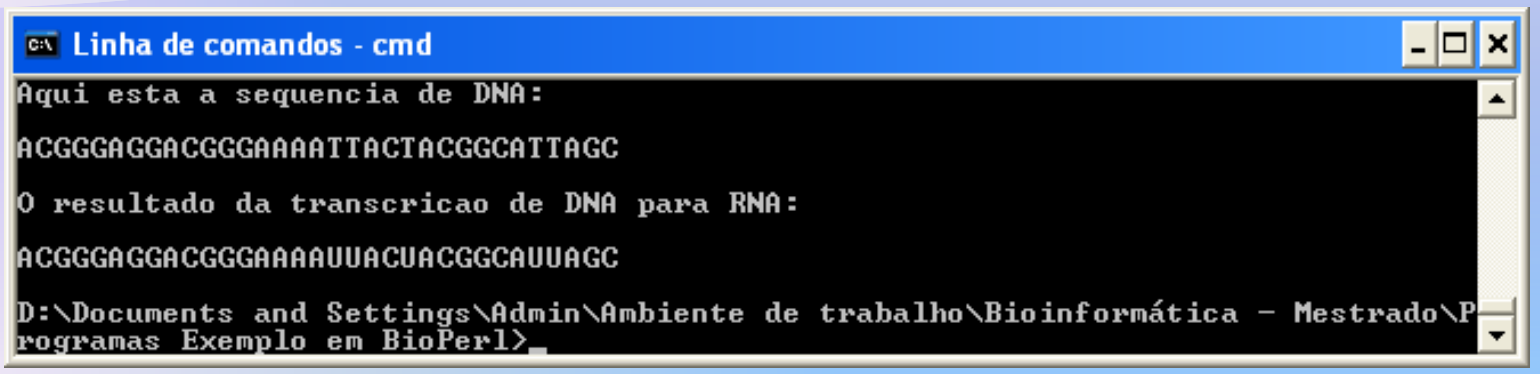
Guardar este código num ficheiro como código Perl (extensão: pl), por exemplo com nome "exemplo.pl" e executar na linha de comandos:

> perl exemplo.pl

© Sérgio Deusdado

Começar a programar em BioPerl

Resultado:




```
$RNA =~ s/T/U/g;
```

substituir

por

Variável escalar

Operador de associação

Operador de substituição

modificador (globalmente)

© Sérgio Deusdado

Começar a programar em BioPerl

Outro exemplo básico , usando os módulos: **Seq** e **SeqIO**:

```
use Bio::Seq;
use Bio::SeqIO;

# cria uma nova sequência de ADN com as bases CATGTAGATAG
my $seq = Bio::Seq->new(-id => 'seq_teste', -seq => 'CATGTAGATAG');

# imprime alguns detalhes sobre a sequência, o comprimento e a reversa complementar
print "A sequencia ", $seq->seq," tem ", $seq->length, " bases \n";
print "A sua reversa complementar: ", $seq->revcom->seq, "\n";

# escreve a sequência num ficheiro usando o formato Fasta
my $out = Bio::SeqIO->new(-file => '>seq_teste.fsa', -format => 'Fasta');
$out->write_seq($seq);
```

Guardar este código num ficheiro como código Perl (extensão: pl), por exemplo com nome "3passos.pl" e executar na linha de comandos:

> perl 3passos.pl

© Sérgio Deusdado

Manipular e converter sequências em BioPerl

O **módulo SeqIO** permite ler e escrever sequências em vários formatos possibilitando conversões entre os diferentes formatos existentes. Veja-se o seguinte exemplo:

```
use Bio::SeqIO;
# lê a sequência de entrada no formato Fasta
$in = Bio::SeqIO->new('-file' => "inputfilename.fsa",
                      '-format' => 'Fasta');

#escreve a sequência de saída no formato Raw (básico)
$out = Bio::SeqIO->new('-file' => ">outputfilename.raw",
                       '-format' => 'raw');

while ( my $seq = $in->next_seq() ) {
                      $out->write_seq($seq);
}
```

Nota: O ficheiro "inputfilename.fsa" deverá existir, para ser acedido, lido e convertido

© Sérgio Deusdado

Manipular e converter sequências em BioPerl

O módulo **PrimarySeq** permite efectuar transformações na sequência primária, por exemplo, efectuar a sua tradução para uma cadeia de amino-ácidos. Veja-se o *script* (exemplo2.pl):

```
use Bio::PrimarySeq;
my $seq = new Bio::PrimarySeq(-seq => 'ATGGGACCAAGTA', -display_id => 'exemplo1');

print " a sequencia tem ", $seq->length, " bases \n";

 print "traducao para amino-acidos: ", $seq->translate()->seq(), "\n";
```

Resultado da execução:

```
> perl exemplo2.pl
 a sequencia tem 13 bases
 traducao para amino-acidos: MGPS
```

© Sérgio Deusdado

Obter sequências do GenBank em BioPerl

O módulo **GenBank**, permite aceder e descarregar sequências do GenBank (NCBI), para tal é necessário que o computador tenha ligação à Internet. Veja-se o seguinte exemplo:

```
use Bio::DB::GenBank;
use Bio::SeqIO;

my $gb = new Bio::DB::GenBank;
my $seqout = new Bio::SeqIO(-fh => \*STDOUT, -format => 'fasta');

#uma única sequência por id
my $seq = $gb->get_Seq_by_id('MUSIGHBA1'); $seqout->write_seq($seq);

# ou por accession
$seq = $gb->get_Seq_by_acc('AF303112'); $seqout->write_seq($seq);

#ou múltiplas sequências relacionadas, pelo método get_Stream_by_id
my $seqio = $gb->get_Stream_by_id([ qw(J00522 AF303112 2981014)]);

while( defined ($seq = $seqio->next_seq )) {
 $seqout->write_seq($seq);
}
```

Definir destino, se ecrã ou ficheiro

© Sérgio Deusdado

Obter *scripts* em BioPerl disponíveis na Internet

- A Internet é uma fonte imediata de *scripts* em BioPerl

- Não se ganha nada em reinventar a roda, reutilizem-se *scripts*

Recursos:
"Googlar":
bioperl scripts

ou mais específico:
bioperl alignment script

Fórum europeu:
http://www.openbiosource.org/

Partilha de Scripts Bioperl em:

- http://www.bioperl.org/wiki/Bioperl_scripts
- http://search.cpan.org/~birney/bioperl-1.2.3/bioscripts.PL

© Sérgio Deusdado

(Algumas) Referências bibliográficas

- Conteúdos de www.bioperl.org

- James D. Tisdall, Beginning Perl for Bioinformatics: An Introduction to Perl for Biologist, edição de O'Reilly, 2001

- *Bioperl course*, por Catherine Letondal e Katja Schuerer
Acessível on-line em:
http://www.pasteur.fr/recherche/unites/sis/formation/bioperl/support.pdf

- D. Curtis Jamison, Perl Programming for Biologists, edição de Wiley-IEEE, 2003

- James D. Tisdall, Mastering Perl for Bioinformatics: [Perl programming for Bioinformatics], edição de O'Reilly, 2003

© Sérgio Deusdado