Introdução à Programação de Computadores para Biologia

Expressões Regulares II

Aula 11

https://tttorres.github.io/introprog2024/

EXPRESSÕES REGULARES

Novo operador =~

1. Busca o padrão em textos (variáveis escalares tipo "string")
2. Retorna VERDADEIRO se o padrão for encontrado
3. Retorna FALSO se o padrão for encontrado

EXPRESSÕES REGULARES

Flexibilidade

Script telefone.pl para reconhecer telefones fixos E celulares de São Paulo com os vários formatos, com somente um telefone informado na linha inteira

```
##!/usr/bin/perl

print "Telefone\:\n";
$tel = <STDIN>;

if ( $tel =~ /^/\(?\d{2}\)?\d{4,5}\-?\d{4}$/ ) {
   print "Eh um telefone!\n";
}

exit;
```

EXPRESSÕES REGULARES

Reconhecimento de padrões

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como sitio.pl
4. Copiar **exemplo01** da página da disciplina.
5. Copie o exemplo 1, e complete o script para reconhecer um sítio de reconhecimento de uma enzima de restrição em uma sequência de DNA

EXPRESSÕES REGULARES

Reconhecimento de padrões

Script: sitio.pl

```
##!/usr/bin/perl

$sequencia = "AGCTAAATCCCGAAACTAAACGTCCCGTTTTGGTATACATTCATGGTGGTGTTT
GTTATAGGGAAAATCATCGTGAATATTATGGACCTGATTATTTTATTAAAAAGGAATTCATGTTGTAT
TAATTACCATACAATATCGTTTGGGAGTTCTAGGTTTTTGAGCTTAAATTCTGAAGAGCTTAATGTAC
CCGGTAATGCTGGCCTTAAGGATCAAGTTATGGCGAATTCTTTACGTTGGATTAAAAATAATTGTGCC
AATTTCGGTGGTAATCCTGATAACATCACTGTCTTTGGTGAGAGTGCTGGTGGAGCCTCTGCCCATTA
CATGATGTTAACTGAACAAACACGTGGTCTCTTCCATCGTGGATTTTAATGTCTGGCAATGCTGTATG
TCCTTGGCCATTAGTCAAAATCAACATCGTGCTTATGCTATAGCTAAATTGACTGGGTATAGAATTCA
GGGTGAAAATAATGATAAGGA";

# A enzima EcoRI reconhece o sítio GAATTC

if ( $sequencia =~ /GAATTC/ ) {
        print "A sequencia possui sitio da enzima EcoRI!\n";
}

exit;
```

EXPRESSÕES REGULARES

Reconhecimento de padrões

E se o padrão estivesse em uma variável?

```
##!/usr/bin/perl

$sequencia = "AGCTAAATCCCGAAACTAAACGTCCCGTTTTGGTATACATTCATGGTGGTGTTT
GTTATAGGGAAAATCATCGTGAATATTATGGACCTGATTATTTTATTAAAAAGGAATTCATGTTGTAT
TAATTACCATACAATATCGTTTGGGAGTTCTAGGTTTTTGAGCTTAAATTCTGAAGAGCTTAATGTAC
CCGGTAATGCTGGCCTTAAGGATCAAGTTATGGCGAATTCTTTACGTTGGATTAAAAATAATTGTGCC
AATTTCGGTGGTAATCCTGATAACATCACTGTCTTTGGTGAGAGTGCTGGTGGAGCCTCTGCCCATTA
CATGATGTTAACTGAACAAACACGTGGTCTCTTCCATCGTGGATTTTAATGTCTGGCAATGCTGTATG
TCCTTGGCCATTAGTCAAAATCAACATCGTGCTTATGCTATAGCTAAATTGACTGGGTATAGAATTCA
GGGTGAAAATAATGATAAGGA";

# A enzima EcoRI reconhece o sítio GAATTC

if ( $sequencia =~ /GAATTC/ ) {
        print "A sequencia possui sitio da enzima EcoRI!\n";
}

exit;
```

EXPRESSÕES REGULARES

Reconhecimento de padrões

E se o padrão estivesse em uma variável?

```
##!/usr/bin/perl

$EcoRI = "GAATTC";

$sequencia = "AGCTAAATCCCGAAACTAAACGTCCCGTTTTGGTATACATTCATGGTGGTGTTT
GTTATAGGGAAAATCATCGTGAATATTATGGACCTGATTATTTTATTAAAAAGGAATTCATGTTGTAT
TAATTACCATACAATATCGTTTGGGAGTTCTAGGTTTTTGAGCTTAAATTCTGAAGAGCTTAATGTAC
CCGGTAATGCTGGCCTTAAGGATCAAGTTATGGCGAATTCTTTACGTTGGATTAAAAATAATTGTGCC
AATTTCGGTGGTAATCCTGATAACATCACTGTCTTTGGTGAGAGTGCTGGTGGAGCCTCTGCCCATTA
CATGATGTTAACTGAACAAACACGTGGTCTCTTCCATCGTGGATTTTAATGTCTGGCAATGCTGTATG
TCCTTGGCCATTAGTCAAAATCAACATCGTGCTTATGCTATAGCTAAATTGACTGGGTATAGAATTCA
GGGTGAAAATAATGATAAGGA";

# A enzima EcoRI reconhece o sítio GAATTC

if ( $sequencia =~ /$EcoRI/ ) {
        print "A sequencia possui sitio da enzima EcoRI!\n";
}

exit;
```

EXPRESSÕES REGULARES

Reconhecimento de padrões

O que o operador =~ retorna?

```
##!/usr/bin/perl

$EcoRI = "GAATTC";

$sequencia = "AGCTAAATCCCGAAACTAAACGTCCCGTTTTGGTATACATTCATGGTGGTGTTT
GTTATAGGGAAAATCATCGTGAATATTATGGACCTGATTATTTTATTAAAAAGGAATTCATGTTGTAT
TAATTACCATACAATATCGTTTGGGAGTTCTAGGTTTTTGAGCTTAAATTCTGAAGAGCTTAATGTAC
CCGGTAATGCTGGCCTTAAGGATCAAGTTATGGCGAATTCTTTACGTTGGATTAAAAATAATTGTGCC
AATTTCGGTGGTAATCCTGATAACATCACTGTCTTTGGTGAGAGTGCTGGTGGAGCCTCTGCCCAT";

# A enzima EcoRI reconhece o sítio GAATTC

if ( $sequencia =~ /$EcoRI/ ) {
        print "A sequencia possui sitio da enzima EcoRI!\n";
}

$resultado = ($sequencia =~ /$EcoRI/);

print "O operador =~ retorna $resultado\n";

exit;
```

EXPRESSÕES REGULARES

Reconhecimento de padrões

O que o operador =~ retorna?

- Em contexto escalar, ele retorna 1 caso o padrão seja encontrado (VERDADEIRO);
- Em contexto escalar, ele retorna "undef" se o padrão não for encontrado (FALSO);

EXPRESSÕES REGULARES

Reconhecimento de padrões

O que o operador =~ retorna em contexto de ARRAY?

```
##!/usr/bin/perl

$EcoRI = "GAATTC";

$sequencia = "AGCTAAATCCCGAAACTAAACGTCCCGTTTTGGTATACATTCATGGTGGTGTTT
GTTATAGGGAAAATCATCGTGAATATTATGGACCTGATTATTTTATTAAAAAGGAATTCATGTTGTAT
TAATTACCATACAATATCGTTTGGGAGTTCTAGGTTTTTGAGCTTAAATTCTGAAGAGCTTAATGTAC
CCGGTAATGCTGGCCTTAAGGATCAAGTTATGGCGAATTCTTTACGTTGGATTAAAAATAATTGTGCC
AATTTCGGTGGTAATCCTGATAACATCACTGTCTTTGGTGAGAGTGCTGGTGGAGCCTCTGCCCAT";

# A enzima EcoRI reconhece o sítio GAATTC

if ( $sequencia =~ /$EcoRI/ ) {
        print "A sequencia possui sitio da enzima EcoRI!\n";
}

@resultado = ($sequencia =~ /$EcoRI/g);

print "O operador =~ retorna @resultado\n";

exit;
```

EXPRESSÕES REGULARES

Reconhecimento de padrões

O que o operador =~ retorna?

- Em contexto de array, ele retorna um array com todos as observações do padrão buscado, se houver pelo menos um (VERDADEIRO);
- Em contexto de array, ele retorna "undef" se o padrão não for encontrado (FALSO);

EXPRESSÕES REGULARES

Reconhecimento de padrões

Contexto de ARRAY (MODIFICADOR "g")

```
##!/usr/bin/perl

$EcoRI = "GAATTC";

$sequencia = "AGCTAAATCCCGAAACTAAACGTCCCGTTTTGGTATACATTCATGGTGGTGTTT
GTTATAGGGAAAATCATCGTGAATATTATGGACCTGATTATTTTATTAAAAAGGAATTCATGTTGTAT
TAATTACCATACAATATCGTTTGGGAGTTCTAGGTTTTTGAGCTTAAATTCTGAAGAGCTTAATGTAC
CCGGTAATGCTGGCCTTAAGGATCAAGTTATGGCGAATTCTTTACGTTGGATTAAAAATAATTGTGCC
AATTTCGGTGGTAATCCTGATAACATCACTGTCTTTGGTGAGAGTGCTGGTGGAGCCTCTGCCCAT";

# A enzima EcoRI reconhece o sítio GAATTC

if ( @sitios = $sequencia =~ /$EcoRI/g ) {
        $num = @sitios;
        print "A sequencia possui $num sitios da enzima EcoRI!\n";
}

exit;
```

EXPRESSÕES REGULARES

Modificadores

Opções para reconhecimento: m/pattern/igmso

CARACTERES	FUNÇÃO
i	Maiúsculas e minúsculas (case-**i**nsensitive)
g	**G**lobal (todas as ocorrências)
m	Modo **m**ultilinhas (se a string tem newline, os operadores "^" e "$" reconhecerão padrões delimitados por newline em vez da string
s	Trata string como uma única linha (**s**ingle line); "." reconhece newline
o	Testa o padrão só uma vez (**o**nce)

EXPRESSÕES REGULARES

Reconhecimento de padrões

E a enzima Hinf I que reconhece o sítio **GANTC**?

```
##!/usr/bin/perl

$HinfI = <completar>;

$sequencia = "AGCTAAATCCCGAAACTAAACGTCCCGTTTTGGTATACATTCATGGTGGTGTTT
GTTATAGGGAAAATCATCGTGAATATTATGGACCTGATTATTTTATTAAAAAGGAATTCATGTTGTAT
TAATTACCATACAATATCGTTTGGGAGTTCTAGGTTTTTGAGCTTAAATTCTGAAGAGCTTAATGTAC
CCGGTAATGCTGGCCTTAAGGATCAAGTTATGGCGAATTCTTTACGTTGGATTAAAAATAATTGTGCC
AATTTCGGTGGTAATCCTGATAACATCACTGTCTTTGGTGAGAGTGCTGGTGGAGCCTCTGCCCAT";

# A enzima HinfI reconhece o sítio GANTC

if ( @sitios = $sequencia =~ /$HinfI/g ) {
        $num = @sitios;
        print "A sequencia possui $num sitios da enzima HinfI!\n";
}

exit;
```

EXPRESSÕES REGULARES

Extração de Dados

Problema:

> Como extrair informação de texto? Por exemplo, extrair o código de área e o telefone em variáveis separadas.

EXPRESSÕES REGULARES

Extração de Dados

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como telefone.pl
4. Copiar **exemplo02** da página da disciplina.
5. Copie o exemplo 2, e teste para telefones nos diferentes formatos.

EXPRESSÕES REGULARES

Extração de Dados

Teste o programa telefone.pl com os seguintes inputs:

- (11))3091-8759
- (11)3091--8759
- ((11)3091-8759
- (11)3091-8759999999999999999999

EXPRESSÕES REGULARES

Extração de Dados

Script telefone.pl

Extrair o código de área e o telefone em variáveis separadas.

```
##!/usr/bin/perl

print "Telefone\:\n";
$tel = <STDIN>;

if ( $tel =~ /^\(?\d{2}\)?\d{4,5}\-?\d{4}$/ ) {
   print "Eh um telefone!\n";
}

exit;
```

EXPRESSÕES REGULARES

Extração de Dados

Script telefone.pl

Extrair o código de área e o telefone em variáveis separadas.

Parênteses ISOLAM o bloco de interesse.

```
##!/usr/bin/perl

print "Telefone\:\n";
$tel = <STDIN>;

if ( $tel =~ /^\(?(\d{2})\)?(\d{4,5}\-?\d{4})$/ ) {
   print "Eh um telefone!\n";
}

exit;
```

EXPRESSÕES REGULARES

Extração de Dados

Alteração no script telefone.pl para extrair o código de área e o telefone em variáveis separadas.

```
##!/usr/bin/perl

print "Telefone\:\n";
$tel = <STDIN>;

if ( $tel =~ /^\(?(\d{2})\)?(\d{4,5}\-?\d{4})$/ ) {
   print "Eh um telefone!\n";
   print "A area eh $1 e o telefone eh $2.\n";
}

exit;
```

EXPRESSÕES REGULARES

Extração de Dados

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como data.pl
4. Copiar **exemplo03** da página da disciplina.
5. Complete o script para extrair dia, mês e ano em uma data no formato DD/MM/AAAA

EXPRESSÕES REGULARES

Extração de Dados

Complete o script para extrair dia, mês e ano em varáveis diferentes a partir de uma data no formato DD/MM/AAAA

```
#!/usr/bin/perl

# formato DD/MM/AAAA
$texto = "Hoje é 21/10/2024.";

if ($texto =~ /##COMPLETE COM A EXPRESSÃO REGULAR##/) {

    # COMANDOS PARA EXTRAIR DIA, MÊS E ANO
    print "Dia: $dia, Mês: $mes, Ano: $ano\n";
}

exit;
```

EXPRESSÕES REGULARES

Extração de Dados

Complete o script para extrair dia, mês e ano em varáveis diferentes a partir de uma data no formato DD/MM/AAAA

```
#!/usr/bin/perl

# formato DD/MM/AAAA
$texto = "Hoje é 21/10/2024.";

if ($texto =~ /(\d{2})\/(\d{2})\/(\d{4})/) {

    ($dia, $mes, $ano) = ($1, $2, $3);

    print "Dia: $dia, Mês: $mes, Ano: $ano\n";

}

exit;
```

EXPRESSÕES REGULARES

Extração de Dados

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como email.pl
4. Copiar **exemplo04** da página da disciplina.
5. Complete o script para identificar e capturar nome de usuário e domínio de um e-mail.
6. `Uso: perl email.pl meu.email@dominio.com`

EXPRESSÕES REGULARES

Extração de Dados

Complete o script para identificar e capturar nome de usuário e domínio de um e-mail.

```
perl email.pl meu.email@dominio.com
```

```
#!/usr/bin/perl

@ARGV || die "USO: perl $0 meu.email@dominio.com";

$email # COMPLETE PARA RECUPERAR O EMAIL #;

if ($email =~ ## COMPLETE COM A EXPRESSÃO REGULAR ##) {
    print "Usuário: $1, Domínio: $2\n";
}

exit;
```

EXPRESSÕES REGULARES

Extração de Dados

Complete o script para identificar e capturar nome de usuário e domínio de um e-mail.

`perl email.pl meu.email@dominio.com`

```
#!/usr/bin/perl

@ARGV || die "USO: perl $0 meu.email@dominio.com";

$email = $ARGV[0];

if ($email =~ /(\w+)\@(\w+\.\w+)/) {

        print "Usuário: $1, Domínio: $2\n";

}

exit;
```

EXPRESSÕES REGULARES

Substituição

Operador s///

```
$string =~ s/PADRÃO/NOVO_PADRÃO/;
```

EXPRESSÕES REGULARES

Substituição

Inclua um bloco de comandos no script telefone.pl para retirar os parênteses do número

```
##!/usr/bin/perl

print "Telefone\:\n";
$tel = <STDIN>;

if ( $tel =~ /^\(?(\d{2})\)?(\d{4,5}\-?\d{4})$/ ) {
   print "Eh um telefone!\n";
   print "A area eh $1 e o telefone eh $2.\n";
}




exit;
```

EXPRESSÕES REGULARES

Substituição

Inclua um bloco de comandos no script telefone.pl para retirar os parênteses do número

```
##!/usr/bin/perl

print "Telefone\:\n";
$tel = <STDIN>;

if ( $tel =~ /^\(?(\d{2})\)?(\d{4,5}\-?\d{4})$/ ) {
   print "Eh um telefone!\n";
   print "A area eh $1 e o telefone eh $2.\n";
}

$tel =~ s/\(//;
$tel =~ s/\)//;

chomp $tel;
print "Sem parenteses: $tel.\n";

exit;
```

EXPRESSÕES REGULARES

Substituição

Altere o script email.pl, para excluir o ".com".

```
perl email.pl meu.email@dominio.com
```

```
#!/usr/bin/perl

@ARGV || die "USO: perl $0 meu.email@dominio.com";

$email = $ARGV[0];

if ($email =~ /(\w+)\@(\w+\.\w+)/) {
        print "Usuário: $1, Domínio: $2\n";
}



exit;
```

EXPRESSÕES REGULARES

Substituição

Altere o script email.pl, para excluir o ".com".

```
perl email.pl meu.email@dominio.com
```

```
#!/usr/bin/perl

@ARGV || die "USO: perl $0 meu.email@dominio.com";

$email = $ARGV[0];

if ($email =~ /(\w+)\@(\w+\.\w+)/) {
        print "Usuário: $1, Domínio: $2\n";
}

$email =~ s/\.com//;
print "E-mail alterado: $email\n";

exit;
```

EXPRESSÕES REGULARES

Substituição

Altere o script email.pl, para trocar o ".com" por ".com.br".

```
perl email.pl meu.email@dominio.com
```

```
#!/usr/bin/perl

@ARGV || die "USO: perl $0 meu.email@dominio.com";

$email = $ARGV[0];

if ($email =~ /(\w+)\@(\w+\.\w+)/) {
        print "Usuário: $1, Domínio: $2\n";
}

$email =~ s/\.com//;
print "E-mail alterado: $email\n";

exit;
```

EXPRESSÕES REGULARES

Substituição

Altere o script email.pl, para trocar o ".com" por ".com.br".

```
perl email.pl meu.email@dominio.com
```

```
#!/usr/bin/perl

@ARGV || die "USO: perl $0 meu.email@dominio.com";

$email = $ARGV[0];

if ($email =~ /(\w+)\@(\w+\.\w+)/) {
        print "Usuário: $1, Domínio: $2\n";
}

$email =~ s/\.com/\.com\.br/;
print "E-mail alterado: $email\n";

exit;
```

EXPRESSÕES REGULARES

Substituição e Transliteração

Operador s///

```
$string =~ s/PADRÃO/NOVO_PADRÃO/;
```

Operador tr///

```
$string =~ tr/PADRÃO/NOVO_PADRÃO/;
```

EXPRESSÕES REGULARES

Substituição e Transliteração

Inclua um bloco de comandos no script telefone.pl para retirar os hífens do número usando tr

```
##!/usr/bin/perl

print "Telefone\:\n";
$tel = <STDIN>;

if ( $tel =~ /^\(?(\d{2})\)?(\d{4,5}\-?\d{4})$/ ) {
   print "Eh um telefone!\n";
   print "A area eh $1 e o telefone eh $2.\n";
}

$tel =~ s/\(//;
$tel =~ s/\)//;

chomp $tel;
print "Sem parenteses: $tel.\n";

exit;
```

EXPRESSÕES REGULARES

Substituição e Transliteração

Inclua um bloco de comandos no script telefone.pl para retirar os hífens do número usando tr

```
##!/usr/bin/perl

print "Telefone\:\n";
$tel = <STDIN>;

if ( $tel =~ /^\(?(\d{2})\)?(\d{4,5}\-?\d{4})$/ ) {
   print "Eh um telefone!\n";
   print "A area eh $1 e o telefone eh $2.\n";
}

$tel =~ tr/\-/ /;

chomp $tel;
print "Sem hifen: $tel.\n";


exit;
```

EXPRESSÕES REGULARES

Substituição e Transliteração

1. No Geany, File > New File.
2. File > Save as...
3. Gravar arquivo como complemento.pl
4. Copiar **exemplo04** da página e complete o script com a *regex* para ler uma sequência de DNA do input padrão e gerar o complemento reverso;

```
Darwin:~ Tatiana$ perl complemento.pl

Sequencia:
ATAGTCG #input do usuario

Complemento: tatcagc #resposta do script
```

EXPRESSÕES REGULARES

Substituição e Transliteração

Script: complemento.pl

```
#!/usr/bin/perl

print "Sequencia\:\n";
$seq = <STDIN>;

chomp $seq;

$seq =~ tr/ATCG/tagc/;

print "Complemento: $seq.\n";

exit;
```

EXPRESSÕES REGULARES

Substituição e Transliteração

Script: complemento.pl

Inclua uma linha de comando para inverter a sequência.

Função *reverse()*

```
Darwin:~ Tatiana$ perl complemento.pl

Sequencia:
ATAGTCG #input do usuario

Complemento reverso: cgactat #resposta do script
```

EXPRESSÕES REGULARES

Substituição e Transliteração

Script: complemento.pl

Inclua uma linha de comando para inverter a sequência.

```
#!/usr/bin/perl

print "Sequencia\:\n";
$seq = <STDIN>;

chomp $seq;

$seq =~ tr/ATCG/tagc/;
$comp = reverse($seq);

print "Complemento reverso: $comp.\n";

exit;
```

EXPRESSÕES REGULARES - EXERCÍCIO 1

INPUT 1: Personagens.txt

```
Nome: Ahsoka Tano
Espécie: Togruta
Planeta: Shili

Nome: Leia Organa
Espécie: Humana
Planeta: Polis Massa
...
```

INPUT 2: PrimeiraAparicao.txt

```
Nome: Ahsoka Tano
Primeira aparicao: Star Wars: The Clone Wars (2008)

Nome: Leia Organa
Primeira aparicao: Star Wars: a New Hope (1977)
...
```

EXPRESSÕES REGULARES - EXERCÍCIO 1

OUTPUT:

PERSONAGEM	ESPECIE	PLANETA	PRIMEIRA APARICAO
Ahsoka Tano	Togruta	Shili	Star Wars: The Clone Wars (2008)
Leia Organa	Humana	Polis Massa	Star Wars: a New Hope (1977)
Luke Skywalker	Humana	Polis Massa	Star Wars: a New Hope (1977)
...	...	...	...

EXPRESSÕES REGULARES - EXERCÍCIO 2

INPUT: dmel-gene-r5.45.fasta

```
>FBgn0033056 type=gene; loc=2R:complement(1944862..194706
3); ID=FBgn0033056; name=CG7856; dbxref=FlyBase:FBgn00330
56,FlyBase:FBan0007856,FlyBase_Annotation_IDs:CG7856,GB_p
rotein:AAF57287,GB:AI945278,GB:AY113248,GB_protein:AAM292
53,GB:BF499103,UniProt/TrEMBL:Q8MZC8,UniProt/TrEMBL:A1Z6J
6,INTERPRO:IPR008957,OrthoDB5_Drosophila:EOG57WNH4,Entrez
Gene:35533,InterologFinder:35533,BIOGRID:61438,DroID:FBgn
0033056,DRSC:FBgn0033056,FLIGHT:FBgn0033056,FlyAtlas:CG78
56-RA,FlyMine:FBgn0033056,GenomeRNAi:35533,modMine:FBgn00
33056; derived_computed_cyto=42A8-42A9%3B Limits computat
ionally determined from genome sequence between @P{PZ}l(2
)09851<up>09851</up>@%26@P{lacW}Src42A<up>k10108</up>@ an
d @P{lacW}l(2)k09848<up>k09848</up>@%26@P{EP}EP407@; gbun
it=AE013599; MD5=7ca9c4371b97aaf7046cf83fefb65eb8; length
=2202; release=r5.45; species=Dmel;
GGTAAAGTTGCCTTGGCGTCAGTTGGCAGTTTGGGAAAAGCCTACACACT
TAATATTTCGATAGATACACTTATTTCGCAATCGTAGAAGATACCACAAA
TCTCTCTTCCGTAAATTATAAGTATGTCCAAGAGGGTGAGCATCATGTTG
CCCGACGAAATACCTGCGGCTCCGTCAGGCAGCAGGAGGAACCCGATGCC
...
```

EXPRESSÕES REGULARES - EXERCÍCIO 2

OUTPUT: dmel-genes.fasta

```
>CG7856
GGTAAAGTTGCCTTGGCGTCAGTTGGCAGTTTGGGAAAAGCCTACACACT
TAATATTTCGATAGATACACTTATTTCGCAATCGTAGAAGATACCACAAA
TCTCTCTTCCGTAAATTATAAGTATGTCCAAGAGGGTGAGCATCATGTTG
CCCGACGAAATACCTGCGGCTCCGTCAGGCAGCAGGAGGAACCCGATGCC
...
>Oatp58Da
TTGTTCGCAGGTAAAAGTTGAGACATGACAGAGGAGCGAGGAAAGGTAGA
CTTGGAGAAAAGTGAAACTTTGCCCTTTATTGAAAAGCCATTTGCAATAT
CGGATAAAGAAAGGGAATCTGCTCCAGAGAATGAAACGGAAGGAGAAGAT
GGTGGAAAAGACACTTATTGCGGTTTCTGGATATTCAAGGGCCCCTCCAT
GCAAAGGTAAGCTCTAAGGTACTCTACAACTACCTTCATGCTTAGGAATC
...
```