課題 2-0,2-1（コード領域の推測とその翻訳）

0.1 課題 2-0,2-1

・2-0：アミノ酸コード表に基づき塩基の 3 つの組をアミノ酸に変換する関数 trans-codon を定義する。

・2-1：DNA 配列をファイルから読み込み、先頭から 3 文字ずつ逐次 trans-codon によってアミノ酸に変換し出力するプログラムを書く。

1 課題 2 の Overview

課題 2 全体を通して、みなさんにやってもらいたいことは 2 つあります。それは、

・自分の担当する塩基配列のなかで、どこからどこまでが実際にタンパク質となる部分（コード領域）かを推測する

・自分の担当する遺伝子のコード領域をアミノ酸配列に翻訳する

の 2 つです。課題 2-1 では、2 番目の目標のうち、コドン／アミノ酸変換の基本的な部分をプログラムしてもらいます。

1.1 コドン／アミノ酸変換の基礎知識

前回までの課題で Perl の詳しい説明は大体したので、今回は省略します。そんなの手抜きでねえか、といわれてしまうかもしれませんが、手抜きです。もちろん。

講義のなかでもあったように、1 つの遺伝子は 1 つのタンパク質を表現しています。では、遺伝子はどのようにタンパク質を表現しているのでしょうか。

実はタンパク質というのは 20 種類のアミノ酸が鎖のように直列にたくさんつながった高分子のことです（ポリペプチドという）。タンパク質の性質はこの鎖のアミノ酸がどんな順番でくっついているかということだけで一意に決まります。だから、アミノ酸の配列がどうなっているか、という一次元の情報があればタンパク質はコードできるわけです。

この性質は DNA の塩基配列が一次元情報だということと、とても親和性があります。実際には、塩基配列（ATGC の並び）は 3 つ 1 組で 1 つのアミノ酸をコードしています。この 3 つの塩基からなる 1 組のことをコドンといいます。例えば、300 塩基の長さの配列は 100 個のアミノ酸が連なってできるタンパク質を表現しています。

つまり、1つのコドンという塩基配列のユニットが1つのアミノ酸というタンパク質を構成するユニットに対応していて、このユニットが連続することで 1 つの遺伝子つまりタンパク質が成立しているわけです。

では、コドンが何で、タンパク質が何でできているのか、よくわかったところで、まず自分の担当の配列をアミノ酸の配列に翻訳してみましょう。

2　課題 2-0

まず、3 文字の塩基配列をアミノ酸表記文字に変換してくれる関数が必要ですね。

各自手元の生物学の本でコドン表を探してみてください。例えば、'ctt' という配列に対応するアミノ酸を見つけてみましょう。講義のなかであったように、T を U に読み換えることに注意してください。'ctt' つまり CUU に対応するのは、Leu つまりロイシンだということがわかります。これを 1 文字で表記すると 'L' になります。

こんな感じの仕事をしてくれる関数を書きます。Perl では関数は sub をつけて定義し、& をつけて読み出します。つまりこの関数の形は、

```
sub trans_codon(){
}
```

このようなものでしょう。

```
$amino_acid=&trans_codon('ctt')
```

とすると、$amino_acid にはロイシンを表す 'L' という文字が入っている、ということになると嬉しいですね。そして、'ctt' だけではなくて、例えば 'atgcttctggtg' のようにある程度長い文字列でも 'MLLV' のように変換できればさらに良いでしょう。というわけで、コドン表とアミノ酸表記表を見ながら、trans_codon() を完成させましょう。実はこれは結構簡単です。前回の課題で使用したハッシュにコドン表を記録し、

```
my %CodonTable = (
                  'ctt', 'L',  'cct', 'P',  'cat', 'H',  'cgt', 'R',
                  'ctc', 'L',  'ccc', 'P',  'cac', 'H',  'cgc', 'R',
                                      .
                                      .
                                      .
                  );
```

のようにしてしまえば、

```
$amino=$CodonTable{'ctt'};
```

とするだけで $amino に一つのアミノ酸が入ります。つまり、あとはこれを前回の課題のように for 文を使って 3 文字ずつ変換してやればよいのです。ただし、ここでは 1 文字ずれるのではなくて 3 文字ずれることに注意してください。関数は以下のようになると思います。

```
sub trans_codon(){
        my $nucleotides = shift;                # 引数である塩基配列を $nucleotides に受け継ぐ
        my $amino = '';

        my %CodonTable = (
                  'ctt', 'L',  'cct', 'P',  'cat', 'H',  'cgt', 'R',
                  'ctc', 'L',  'ccc', 'P',  'cac', 'H',  'cgc', 'R',
                                      .
                                      .
                                      .
```

```
                    );

  for (**********************){
    ******************  #for 文で使っているカウンタをもとに、substr で 3 文字切り出す。
    ******************  # コドンが入っている変数を %CodonTable に引渡してアミノ酸に変換。
    ******************  #$amino に上でもとめたアミノ酸を連結。
  }

  return $amino;                        # 関数の結果を返す
}
```

3　課題 2-1

コドンを翻訳してくれる関数ができてしまえば後は簡単です。

１．課題 1 で作ったファイル読み出しのプログラムでまず塩基配列を読み込み、$seq に入れる。

２．$seq を &trans_codon() 関数でアミノ酸に変換

３．結果を出力

以上の手順で完成です。

4　課題 2-1：もう少し美しく書く方法

ここでは課題 2-0 と 2-1 を少し高度な Perl 言語の文法を使って完結に書き直してみます。

sub trans_codon の中の for 文では処理が 3 行に渡っていましたが、これは無駄が多い処理です。1 行目の結果を 2 行目で処理し、その結果を 3 行目で処理しているので、これは一行にまとめられます。方法としては、結果を新しい変数にしないで、そのままそれを次の変数に引き渡す、というやりかたを使います。

```
$amino .= $CodonTable{substr($seq, $i, 3)};
```

このように高度な書き方をすれば、for 文も一行でまとめられます。

```
for(***********){ $amino .= $CodonTable{substr($seq, $i, 3)};}
```

Perl では非常に簡潔にプログラムを書くことが可能で、それが余計なバグを回避する方法につながる場合も多いので、日ごろから簡潔に書くように心がけておくと良いでしょう。