The distribution of nucleotides in human mitochondrial DNA

From

Essentials of Programming in Mathematica

Exercises from Chapter 7, Strings

7.2 Operations on strings

Exercises

1. In Exercise 8 from Section 5.1, a function was created to generate *n*-grams from a given alphabet. For example, that function could be used to create all bigrams (words of length two) from the nucleotide alphabet `{"G", "C", "A", "T"}`.

   Read in a nucleotide sequence such as the human mitochondrial genome `hsMito` below and then create a histogram showing the frequency of each of the sixteen possible bigrams AA, AC, AT, etc.

```
In[1]:= hsMito = First@Import["ExampleData/mitochondrion.fa.gz"];
        Short[hsMito]
Out[2]//Short= GATCACAGGTCTATCACCCTATTAACCACTCACGGGAGC … AAATAGCCCACACGTTCCCCTTAAATAAGACATCACGATG
```

Solution

1. We are interested in finding all bigrams from the alphabet `{"A", "T", "C", "G"}`. `Outer` can be used to generate all possible pairs. We will use the symbol `sj` below as the function that will be applied to each pair.

```
In[3]:= alphabet = {"A", "T", "C", "G"};

In[4]:= Outer[sj, alphabet, alphabet]

Out[4]= {{sj[A, A], sj[A, T], sj[A, C], sj[A, G]}, {sj[T, A], sj[T, T], sj[T, C], sj[T, G]},
         {sj[C, A], sj[C, T], sj[C, C], sj[C, G]}, {sj[G, A], sj[G, T], sj[G, C], sj[G, G]}}
```

   Now use `StringJoin` instead of `sj`:

```
In[5]:= Outer[StringJoin, alphabet, alphabet]

Out[5]= {{AA, AT, AC, AG}, {TA, TT, TC, TG}, {CA, CT, CC, CG}, {GA, GT, GC, GG}}
```

   After flattening this list, we have all sixteen possible pairs. But what if we want possible triples or quadruples? Basically, we need a sequence of three or four or *n* lists of the alphabet. Hence we need to apply `Sequence` to avoid deeply nested lists and get the correct structure.

```
In[6]:= With[{n = 3},
         Outer[StringJoin, Apply[Sequence, Table[alphabet, {n}]]]
        ]
Out[6]= {{{AAA, AAT, AAC, AAG}, {ATA, ATT, ATC, ATG}, {ACA, ACT, ACC, ACG}, {AGA, AGT, AGC, AGG}},
         {{TAA, TAT, TAC, TAG}, {TTA, TTT, TTC, TTG}, {TCA, TCT, TCC, TCG}, {TGA, TGT, TGC, TGG}},
         {{CAA, CAT, CAC, CAG}, {CTA, CTT, CTC, CTG}, {CCA, CCT, CCC, CCG}, {CGA, CGT, CGC, CGG}},
         {{GAA, GAT, GAC, GAG}, {GTA, GTT, GTC, GTG}, {GCA, GCT, GCC, GCG}, {GGA, GGT, GGC, GGG}}}
```

   Here then is the `NGrams` function:

```
In[7]:= NGrams[alphabet : {__String}, n_Integer?Positive] :=
         Flatten[Outer[StringJoin, Apply[Sequence, Table[alphabet, {n}]]]]
```

   This gives all possible bigrams from the DNA alphabet.

```
In[8]:= bigrams = NGrams[{"A", "T", "G", "C"}, 2]

Out[8]= {AA, AT, AG, AC, TA, TT, TG, TC, GA, GT, GG, GC, CA, CT, CG, CC}
```

   This finds all the above bigrams in the sequence.

```
In[9]:= data = StringCases[hsMito, Alternatives @@ bigrams, Overlaps → True];

In[10]:= RandomSample[data, 12]

Out[10]= {AC, GC, TG, AA, GT, AC, CG, TC, CC, CT, TC, GC}
```

   And here is the histogram plot. Hovering your mouse over the bars (in a notebook or CDF) gives the percentages of each bigram.

In[11]:=
```
Histogram[data /. Thread[bigrams → Range[Length[bigrams]]], Automatic,
 "PDF", ChartLabels → {None, Placed[Style[#, "SR", 6] & /@ bigrams, Bottom]},
 Ticks → {None, Automatic}, AspectRatio → .4, ChartStyle → "Pastel"]
```

Out[11]=