Exercises and solutions for chapter 'Working with Character Data'

August 11, 2008

Exercise 1
Using the code above, create a simple function that maps from DNA to the amino acid sequence.

Solutions: A very bare bones solution is given below. You might want to add some checking (is the input really a DNA sequence?). It can only really deal with inputs of single DNA sequences (largely due to the semantics of `substring`.

```
> DNA2AA = function(DNAseq) {
+     nc = nchar(DNAseq)
+     Dtriples = substring(DNAseq, seq(1, nc,
+         by = 3), seq(3, nc, 3))
+     paste(GENETIC_CODE[Dtriples], collapse = "")
+ }
```

Exercise 2
What happens if the stop, or last, argument to `substr` *or* `substring` *is larger than the number of characters? Is it different for the replacement version? In the replacement version, what happens if the length of the string to assign is longer than the character vector.*

Exercise 3
Compare the function `strbreak` *with* `strwrap` *and* `strtrim`*. What are the differences in terms of the output generated?*

Exercise 4
Write a function for translating from RNA to DNA. Test it and `dna2rna` *on a vector of inputs.*

Exercise 5
Write a function to test whether a sequence is a DNA sequence or an RNA sequence. Modify the function `compSeq` *above to use the test and perform the appropriate translation, depending on the type of input sequence.*

Solutions: It will be much easier to solve this problem, using regular expressions, see Section **??**, the code here works, but is very inefficient and should not be used in any real application.

```
> isDNA = function(x) {
+     xU = toupper(x)
+     spx = strsplit(xU, NULL)
+     sapply(spx, function(z) all(z %in% c("A",
+         "C", "G", "T")))
+ }
```

One can then write a similar function to test whether a character vector represents RNA.

Exercise 6
Look at the manual page for `strsplit` *to get an idea of how to write a function that reverses the order of characters in the character strings of a character vector. Use this to write a reverseComplement function.*

Solutions: The definition, from the `strsplit` manual page is given next.

```
> strReverse <- function(x) sapply(lapply(strsplit(x,
+     NULL), rev), paste, collapse = "")
```

And our reverse complement function would look something like:

```
> reverseComplement = function(x) strReverse(compSeq(x))
```

Exercise 7
Test the claims made above about matching of the empty string; show that with `pmatch` *there is no match, while with* `charmatch` *there is.*

Exercise 8
Write a function that takes a character vector as input and checks to see which elements have only nucleotide characters in them.

Solutions: The function `onlyNuc`, below, returns a logical vector of the same length as its input, indicating which of the elements of the input vector contain only the four nucleotides.

```
> onlyNuc = function(x) {
      ans = rep(TRUE, length(x))
      noNuc = grep("[^ACTGactg]", x)
      ans[noNuc] = FALSE
      ans
  }
```

Exercise 9
Create a valid regular expression that checks to make sure that both the month and day specifications are correct.

Solutions: There are of course, many different ways to do that. Among them, for the month specification is to use alternation, for example `(0[1-9]|1[0-2])`

```
> regexpr("(0[1-9]|1[0-2])\\/\\d\\d\\/\\d\\d\\d\\d",
      "today is 12/01/1977", perl = TRUE)
[1] 10
attr(,"match.length")
[1] 10
> regexpr("(0[1-9]|1[0-2])\\/\\d\\d\\/\\d\\d\\d\\d",
      "today is 21/01/1977", perl = TRUE)
[1] -1
attr(,"match.length")
[1] -1
```

Exercise 10
What is the purpose of the `*` *in the regular expressions? Can you extend this to deal with white space as defined by* `[:space:]`*? Write a function similar to* `strwhite` *that replaces two or more leading blanks with a single space. Modify* `strwhite` *to also strip* `\n` *from the end of a line.*

Exercise 11
Write a version of `complementSeq` *that works for either DNA or RNA using* `chartr`*. How does the speed compare with that of the version in the* ***matchprobes*** *package? Write a version of* `reverseSeq` *using* `strsplit`*,* `rev` *and* `paste`*. How does the speed of that function compare with the one in the* ***matchprobes*** *package?*

Solutions:

```
> cS = function(x, DNA = TRUE) {
      if (DNA)
          chartr("GCAT", "CGTA", x)
      else chartr("GCAU", "CGUA", x)
  }
```

The `strrev` function in **Biostrings** is answer to the second programming problem. Use `system.time` on some reasonably long strings to obtain timing comparisons.

Exercise 12
Find all of the palindromes that have all four bases present. Are their sequences also highly repetitive?

Solutions:

```
> allThere = function(x) all(x[1:4] > 0)
> x1 = apply(ans, 1, allThere)
> chr22_pals[x1]
  Views on a 49691432-letter DNAString subject
subject: NNNNNNNNNNNNNNNNNNNNNNNNN...NNNNNNNNNNNNNNNNNNNNNNNNN
views:
       start      end width
[1] 32511111 32511207    97 [CATATATGTGTATAC...ATATGTGTATATAC]
[2] 32511112 32511238   127 [ATATATGTGTATACA...CATATGTGTATATA]
[3] 32511144 32511238    95 [ATATATGTGTATACA...CATATGTGTATATA]
```

And yes, they do seem to be highly repetative.

Exercise 13
Find all the complemented palindromes on Chromosome 22.

Solutions:

```
> cpals = findComplementedPalindromes(chr22NoN, min.armlength = 40,
      max.looplength = 20)
> cpals
  Views on a 49691432-letter DNAString subject
subject: NNNNNNNNNNNNNNNNNNNNNNNNN...NNNNNNNNNNNNNNNNNNNNNNNNN
views:
       start      end width
[1] 15384014 15384111    98 [TATATATATATATAC...TATATATATATATA]
```

```
[2] 15384014 15384139   126 [TATATATATATATAC...TATATATATATATA]
[3] 15384072 15384165    94 [TATATATATATACAT...TGTATATATATATA]
[4] 26634205 26634296    92 [GAGAATATTTATCAC...TGATAAATATTCTC]
```

Exercise 14
Over evolutionary time methylated cytosines (C) are converted to thymines (T) due to spontaneous deamination. Modify the penalty matrix `mat` *above to penalize less for this conversion than for the others. How does that change the two alignments?*

Solutions: We change the cost to -1, and observe that the effect is somewhat more on the alignment with a low gap penalty. With the high gap penalty there seems to be no difference between using the first penalty matrix and the new one.

```
> mat2 = mat
> mat2["C", "T"] = mat2["T", "C"] = -1L
> dnaAlign3 = needwunsQS(Sc, Sp, mat2, gappen = 1)
> dnaAlign4 = needwunsQS(Sc, Sp, mat2, gappen = 6)
> nchar(dnaAlign4)
[1] 1587
```