Analyzing Peptide Libraries with peptider

Eric Riemer Hare

January 7, 2014

1 Introduction

Libraries of peptides, or amino acid sequences, have a number of applications in the Biological sciences, from studying protein interactions, to vaccine research. Despite their importance, little analysis has been done to assess the statistical properties of different peptide libraries.

Peptider is a newly-released R package which helps to evaluate many important statistical properties of these libraries. It supports a number of built-in library schemes, including NNN, NNB, NNK, NNS, and trimer schemes. It also allows for easy analysis of user-created custom library schemes. Peptider makes use of the R package *discreteRV*, which allows for manipulation and analysis of discrete random variables. By treating each amino acid in a peptide as a realization of an independent draw from the pool of all possible amino acids, probabilities for the occurrence of peptides can easily be formulated.

This paper will focus on two distinct functional areas of peptider. The first is Library Diversity, or statistical measures of the quality of the library itself. The second is Peptide Coverage, or how likely the library is to include particularly desired peptides, or peptides that are most similar to desired peptides. Before proceeding to discuss these measures, we will first discuss the built-in library schemes, and how to define custom schemes.

1.1 Library Schemes

Peptider has several built-in library schemes. The first is the NNN scheme, in which all four bases (Adenine, Guanine, Cytosine, and Thymine) can occur at all three positions in a particular codon, and hence there are 64 possible nucleotides. The second is the NNB scheme, where the first two positions are unrestricted, but the third position can only be three bases, yielding 48 nucleotides. Both NNK and NNS have identical statistical properties in this analysis, with the third position restricted to two bases for a total of 32 nucleotides. Finally, there are trimer-based libraries in which the codons are pre-defined. Each of these scheme definitions can be accessed with the *scheme* function.

```
scheme("NNN")
```

```
##    class      aacids c
## 1      A         SLR 6
## 2      B       AGPTV 4
## 3      C           I 3
## 4      D   DEFHKNQYC 2
## 5      E          MW 1
## 6      Z           * 3
```

To build a library of an appropriate scheme, the libscheme function is used. By default, peptides of length one amino acid (k = 1) will be used, but this can be specified.

```
nnk6 <- libscheme("NNK", k = 6)
```

libscheme returns a list containing two elements. The first, data, describes the probability of occurrence of each possible peptide class. The second, info, describes the number of nucleotides, the number of valid nucleotides, and the scheme definition used.

We can also create a custom library scheme by building a data frame of the same format as in Code Example 1. This code creates a custom trimer-based library with peptides of length six.

```
custom <- data.frame(class = c("A", "Z"), aacids = c("SLRAGPTVIDEFHKNQYMW",
    "*"), c = c(1, 0))
custom6 <- libscheme(custom, k = 6)
```

Having created the library of interest, we now turn our attention to assessment of these libraries.

2 Library Diversity

In this section, we introduce a number of properties which can be used to determine the quality of a given peptide library, and which are computable using peptider.

2.1 Functional Diversity

The functional diversity of a library is the overall number of different peptides in the library. Analyzing the peptide sequences directly is complex, so a useful approach is to partition the library into amino acid classes, wherein each amino acid belonging to a particular class has the same number of codon representations as all other amino acids in that class. Letting v represent the number of valid amino acid classes, k represent the number of amino acids in each peptide, b_i represent the number of different peptides in class i, N represent the total size of the peptide library in number of peptides, and p_i represent the probability of peptide class i, then the functional diversity is:

$$D(N, k) = \sum_{i=1}^{v^k} b_i (1 - e^{-Np_i/b_i})$$

To compute this diversity measure in peptider, the *makowski* function can be used.

```
makowski(6, "NNK")
```

```
## [1] 0.2918
```

2.2 Expected Coverage

The expected coverage of the library is directly related to the diversity of the library. It is the percentage of all possible peptides that the library contains. Letting c represent the number of viable amino acids in the library scheme, the expected coverage is:

$$C(N, k) = D(N, k)/c^k$$

Note that the diversity of the library can range between 0 (for a library with no peptides) and c^k (wherein the library includes every possible peptide). Hence, the coverage must range from 0 to 1. Peptider includes a function to compute the coverage, which again takes the peptide length and library scheme as parameters. It also accepts a parameter N representing the overall number of peptides in the library.

```
coverage(6, "NNK", N = 10^8)
```

```
## [1] 0.5229
```

2.3 Relative Efficiency

In general, it is desireable to achieve a coverage as close as possible to one if we'd like the library to contain all possible peptides. One can achieve this with arbitrarily high probability by increasing the library size (N). However, doing so can drastically increase the cost of the library and may not always be possible. Ideally, the library should contain as high as possible diversity with as small as possible size. Relative Efficiency is a measure to quantify this, and is defined as:

$$R(N, k) = D(N, k)/N$$

As N increases, the relative efficiency subsequently decreases. An ideal peptide library from a cost-benefit perspective would contain both a high diversity (and hence high expected coverage) and still a high relative effeciency. Efficiency can be computed with peptider in the same way as coverage.

```
efficiency(6, "NNK", N = 10^8)
```

```
## [1] 0.3347
```

3 Peptide Coverage

The measures of library diversity included in peptider introduced to this point are useful to broadly assess the library performance. However, applications of peptide libraries often require knowledge of the probability of observing particular peptides in the library[REF].

3.1 Peptide Inclusion

The peptide inclusion probability is the probability of obtaining at least one instance of a particular peptide in the library. If X is defined as a random variable representing the number of occurrences of this peptide, the probability is

$$P(X \geq 1) = 1 - P(X = 0) \approx 1 - e^{-N \sum_i p_i}$$

p_i again is the probability of peptide class i. Because of the dependence on the peptide class, this inclusion probability will vary depending on the particular peptide of interest, and the library scheme. The *ppeptide* function in peptider allows computation of the probability. It accepts a peptide sequence as a character vector, the library scheme, and the library size.

```
ppeptide("HENNING", "NNK", N = 10^10)
```

```
## [1] 0.5166
```

We may also be interested in the range of inclusion probabilities for all peptides rather than just a particular sequence. In this case, the *detect* function is useful. detect differs a bit syntactically as it requires the created library to be passed in as an argument. For an NNK library with peptide lengths 6 and library size 10^7, we have:
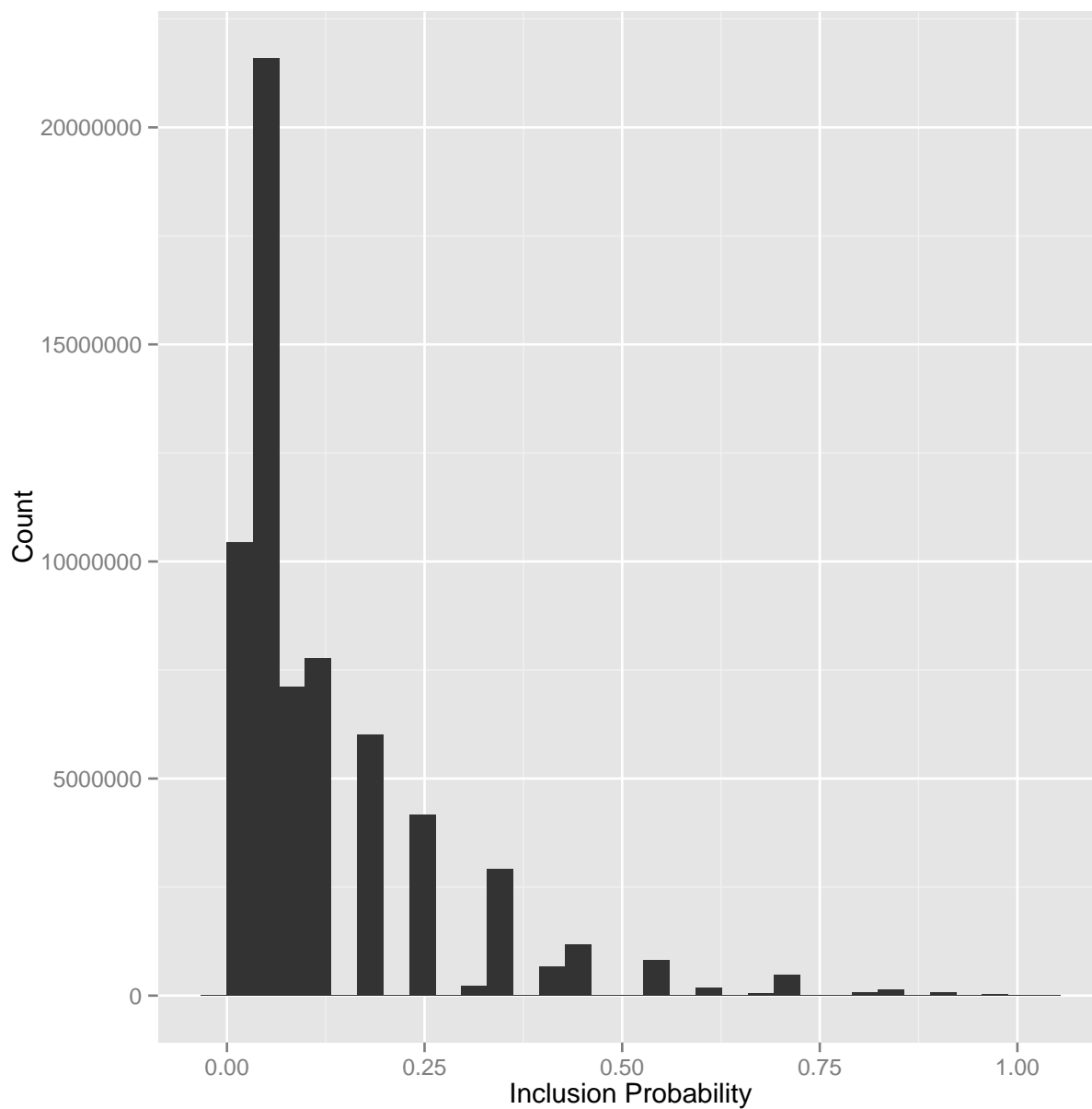
```
my_lib <- libscheme("NNK", 6)

summary(detect(my_lib, 10^7))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0112  0.1840  0.3330  0.3980  0.5560  1.0000
```

```
qplot(detect(my_lib, 10^7), weight = di, geom = "histogram", data = my_lib$data) +
    xlab("Inclusion Probability") + ylab("Count")
```

```
## stat_bin:  binwidth defaulted to range/30.  Use 'binwidth = x' to adjust this.
```

3.2 Neighborhoods

A related concern to inclusion of a particular peptide is inclusion of a peptide's "neighbors". A degree-n neighbor of a peptide p is any peptide that differs by at most n amino acids from p. For example, the length 7 peptide HENNING includes degree-1 neighbor QENNING as well as HQNNING, and a degree-2 neighbor YENNLNG. Peptider includes a function for working with neighborhoods, *getNeighbors*, which accepts a peptide sequence as a character vector and returns all degree-1 neighbors.

```
getNeighbors("HENNING")
```

```
##  [1] "HENNING" "QENNING" "YENNING" "HDNNING" "HQNNING" "HKNNING" "HEDNING"
```

```
##  [8] "HENDING" "HENNLNG" "HENNMNG" "HENNVNG" "HENNIDG"
```

Although there is no built-in function for computing degree-2 and greater neighborhoods for a given peptide, the functionality can be emulated by successive calling of getNeighbors, i.e.,

```r
unique(unlist(getNeighbors(getNeighbors("HENNING"))))
```

```
##  [1] "HENNING" "QENNING" "YENNING" "HDNNING" "HQNNING" "HKNNING" "HEDNING"
##  [8] "HENDING" "HENNLNG" "HENNMNG" "HENNVNG" "HENNIDG" "RENNING" "EENNING"
## [15] "KENNING" "QDNNING" "QQNNING" "QKNNING" "QEDNING" "QENDING" "QENNLNG"
## [22] "QENNMNG" "QENNVNG" "QENNIDG" "FENNING" "WENNING" "YDNNING" "YQNNING"
## [29] "YKNNING" "YEDNING" "YENDING" "YENNLNG" "YENNMNG" "YENNVNG" "YENNIDG"
## [36] "HNNNING" "HDDNING" "HDNDING" "HDNNLNG" "HDNNMNG" "HDNNVNG" "HDNNIDG"
## [43] "HRNNING" "HHNNING" "HQDNING" "HQNDING" "HQNNLNG" "HQNNMNG" "HQNNVNG"
## [50] "HQNNIDG" "HKDNING" "HKNDING" "HKNNLNG" "HKNNMNG" "HKNNVNG" "HKNNIDG"
## [57] "HEENING" "HEDDING" "HEDNLNG" "HEDNMNG" "HEDNVNG" "HEDNIDG" "HENEING"
## [64] "HENDLNG" "HENDMNG" "HENDVNG" "HENDIDG" "HENNLDG" "HENNMDG" "HENNVDG"
## [71] "HENNIEG"
```

4 Further Work

Although these functions are suitable for working with peptides up to about length ten, they become slower and more problematic for larger peptides. Work has progressed on a new suite of functions which simplify the encoding of peptides. By recognizing the assumption of independence of each amino acid, we can store counts of each peptide class in order to avoid storing all possible permutations of peptide classes. This greatly simplifies the computation time and allows for peptides of length 20 and beyond to be analyzed in the context of peptide libraries.

Replacement functions for most of the previously described functionality using this new encoding scheme is available unexported in peptider. For instance, to compute the coverage of a size $10^2 5$ NNK library with peptides of length 18, one can call:

```r
peptider:::coverage_new(18, "NNK", N = 10^25)
```

```
## [1] 0.7923
```

Functions for coverage, efficiency, diversity, and inclusion probabilities have all been written and have the "new" suffix.

5 Conclusion