Analyzing Peptide Libraries with peptider

by Eric Hare, Heike Hofmann, and Timo Sieber

Abstract Peptide libraries have important theoretical and practical applications in the fields of biology and medicine. This paper introduces a new R package **peptider** which allows for a statistical analysis of these peptide libraries. Based on the research of Sieber et al. (2014), **peptider** implements a suite of functions to calculate functional diversity, relative efficiency, expected coverage, and other measures of peptide library diversity. With flexible support for a number of encoding schemes and library sizes, **peptider** can be used to analyze a wide variety of peptide libraries.

Introduction

Libraries of peptides, or amino acid sequences, have a number of applications in the Biological sciences, from studying protein interactions, to vaccine research (Rodi et al., 1999; Irving et al., 2001). 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 20/20 schemes. It also allows for easy analysis of user-created custom library schemes. **peptider** makes use of the R package **discreteRV** (Buja, 2013), which allows for manipulation and analysis of discrete random variables; all of the graphical output is presented within the **ggplot2** framework (Wickham, 2009). 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. The corresponding definitions for the statistical functions in **peptider** are taken from Sieber et al. (2014).

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.

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 $4^3 = 64$ possible nucleotides. The second is the NNB scheme, where the first two positions are unrestricted, but the third position can only be one of the three bases C, G, or T, yielding 48 nucleotides. Both NNK and NNS have identical statistical properties in this analysis, with the third position restricted to two bases (G or T for NNK, and G or C for NNS, respectively) for a total of 32 nucleotides. Finally, there are trimer-based libraries in which the codons are pre-defined, as in the 20/20 scheme included in **peptider**. Figure **??** illstruates a codon wheel by which the three bases lead to specific codon representations for amino acids (Blin, 2012). During peptide library construction, these codons are built from the bases, and restrictions may be imposed in order to reduce the number of codon representations of particular amino acids.

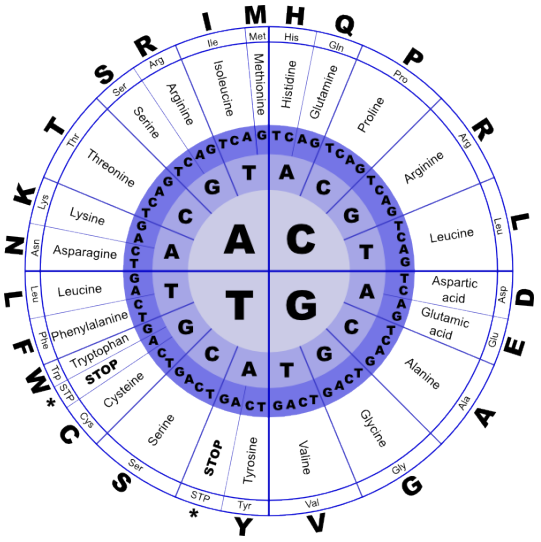


Figure 1: Codon wheel illustrating which bases lead to codon representations for each amino acid.

Each of these scheme definitions can be accessed with the `scheme` function.

```
scheme("NNN")

##   class     aacid 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 also provide the possibility to define a custom library scheme. To do so, a data frame containing three columns “class", “aacid", and “c" must be created. Here is an example of a data frame describing an NNK scheme in which Cysteine is regarded as inviable, and hence ignored.

```
custom_nnk <- data.frame(class = c("A", "B", "C", "Z"),
                         aacids = c("SLR", "AGPTV", "DEFHIKMNQWY", "C*"),
                         c = c(3, 2, 1, 1))
custom_nnk

##   class      aacids c
## 1     A         SLR 3
## 2     B       AGPTV 2
## 3     C DEFHIKMNQWY 1
## 4     Z          C* 1
```

Note that there are three columns required in order to define a custom scheme:

- class - This should store the first n capital letters in the alphabet, where n is the number of sets of amino acids with different numbers of codon representations, and then the letter Z. In this example, the amino acids SLR each have three codon representations, AGPTV each have two, and DEFHIKMNQWY each have one. Hence, we store three classes, A, B, and C, followed by the Z (ignored) class.

- aacid - This should define which amino acids belong to each of the classes defined.
- c - This should store the count of the number of codon representations for each class.

Once a scheme is defined, we can pass it to the `libscheme` function in order to generate a new custom library. This function accepts an argument k representing the length of the peptides in the library.

```
custom_nnk6 <- libscheme(custom_nnk, k = 6)
```

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

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**.

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 expected 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 `diversity` function is used:

```
diversity(6, "NNK", N = 10^6)

## [1] 808963.1
```

A related measure of diversity is available in **peptider**. Makowski Diversity is a measure of library diversity where ranging from zero to one, in which a one represents a library in which each possible peptide has an even probability of selection, and tends towards zero for increasingly skewed distributions (**?**). This can be computed using the `makowski` function:

```
makowski(6, "NNK")

## [1] 0.2917751
```

Expected Coverage

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

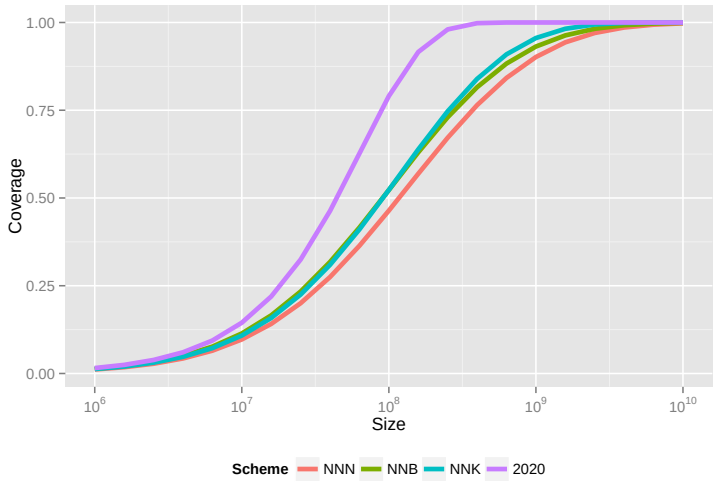


Figure 2: Overview of coverage rates across different hexapeptide library schemes of varying size.

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

## [1] 0.5229421
```

It may be of interest to compare the expected coverage across the different encoding schemes available in **peptider**. We can plot the expected coverage as a function of the library size for all four built-in encoding schemes as follows:

```
library(plyr)
library(ggplot2)

dframe <- expand.grid(Scheme = c("NNN", "NNB", "NNK", "2020"), Size = 10^seq(6,10, by=0.2))
results.dframe <- ddply(dframe, .(Scheme, Size), function(row){
    coverage(6, as.character(row$Scheme), row$Size)
})
names(results.dframe)[3] <- "Coverage"
```

```
qplot(Size, Coverage, data = results.dframe, geom = "line", colour = Scheme, size = I(1.5)) +
   scale_x_log10(breaks = 10^(6:10), labels = expression(10^6, 10^7, 10^8, 10^9, 10^10)) +
    theme(legend.position = "bottom")
```

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] 0.3346829
```

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. **peptider** helps to analyze the the probability of observing both the peptide of interest to the user, as well as peptides that are most similar.

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] 0.5166138
```

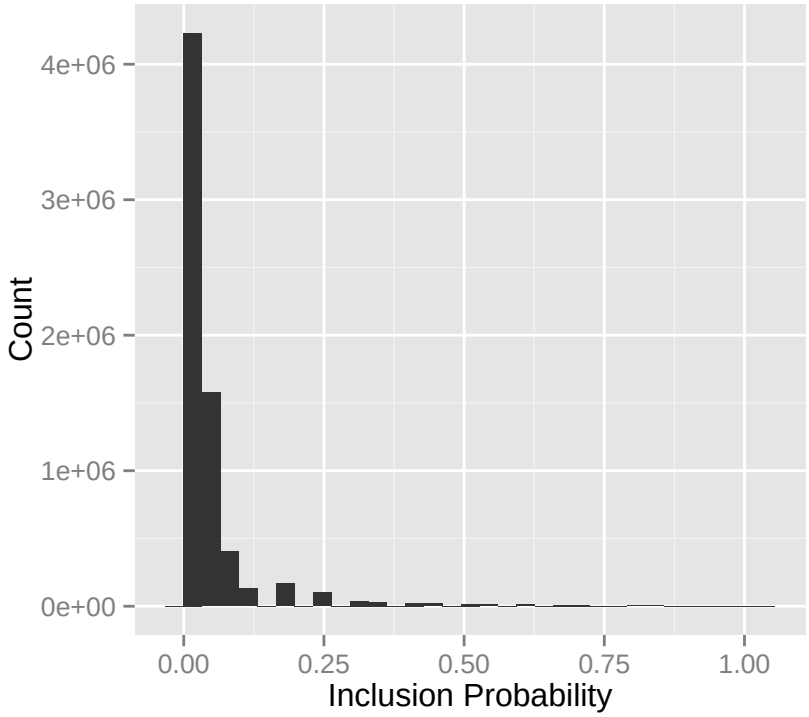
We may also be interested in the range of inclusion probabilities for all peptides rather than just a particular sequence. In this case, the `detect` function is useful. detect differs a bit syntactically as it requires the created library to be passed in as an argument. For an NNK library with peptide 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  0.1190  0.3756  0.4405  0.7285  0.9997
```

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



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.,

```
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"
```

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.

Conclusion

peptider aims to provide a seamless way to assess the quality of different peptide libraries. By providing functions to calculate both the diversity of the peptide library itself, as well as inclusion probabilities of particular peptides of interest, it can help researchers in other fields to make a more informed decision regarding which libraries to invest in.

Bibliography

K. Blin. *antiSMASH: Searching for New Antibiotics Using Open Source Tools*, 2012. URL http://kblin.org/talks/lca12/antismash_lca2012.html. [p1]

A. Buja. *discreteRV: Functions to create and manipulate discrete random variables*, 2013. R package version 1.0.2. [p1]

H. Hofmann, E. Hare, and ggobi Foundation. *peptider: Evaluation of diversity in nucleotide libraries*, 2013. R package version 0.1.2. [p]

M. B. Irving, O. Pan, and J. K. Scott. Random-peptide libraries and antigen-fragment libraries for epitope mapping and the development of vaccines and diagnostics. *Current opinion in chemical biology*, 5(3):314–324, 2001. [p1]

L. Makowski and A. Soares. Estimating the diversity of peptide populations from limited sequence data. *Bioinformatics*, 19(4):483–489, 2003. [p]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. URL http://www.R-project.org/. [p]

D. J. Rodi, R. W. Janes, H. J. Sanganee, R. A. Holton, B. Wallace, and L. Makowski. Screening of a library of phage-displayed peptides identifies human bcl-2 as a taxol-binding protein. *Journal of Molecular Biology*, 285(1):197 – 203, 1999. ISSN 0022-2836. doi: 10.1006/jmbi.1998.2303. URL http://www.sciencedirect.com/science/article/pii/S0022283698923038. [p1]

T. Sieber, E. Hare, H. Hofmann, and M. Trepel. Biomathematical description of peptide library properties. *Bioinformatics*, 2014. [p1]

H. Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. ISBN 978-0-387-98140-6. URL http://had.co.nz/ggplot2/book. [p1]