A Framework and Application for Efficient Analysis of Peptide Libraries

Eric Riemer Hare

March 6, 2014

My creative component consists of three separate papers, each covering a different component of the overall project I worked on. The layering of the three components is illustrated in Figure 1. The structure of this document is organized in the same manner, beginning with discreteRV, continuing with peptider, and ending with PeLiCa. The discreteRV and peptider papers are to be submitted to the R journal, while the PeLiCa paper is to be submitted to the software issue of the Journal of the ACM. I have also attached the paper on peptide libraries, by Sieber et al., of which I was a contributor, which inspired much of this work. It was submitted to the Bioinformatics section of the Oxford Journals.

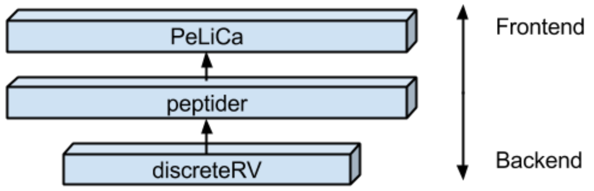


Figure 1: The three components, or layers, of my project.

Manipulation of Discrete Random Variables with discreteRV

by Eric Riemer Hare, Andreas Buja, and Heike Hofmann

Abstract A major issue in statistics education is the sometimes large disparity between the mathematical and theoretical coursework, and the computational coursework. **discreteRV** is an R package for manipulation of discrete random variables which uses clean and familiar syntax similar to that which is found in introductory probability courses. It is simple enough for those less experienced with statistical programming, but has more advanced features which are suitable for a large number of more complex applications. In this paper, I introduce and motivate **discreteRV**, describe its functionality, and provide reproducible examples illustrating its use.

Introduction

One of the primary hurdles in teaching probability courses in an undergraduate setting is the missing link between theoretical statements from textbooks and lectures, and the notation used in statistical software required in more and more classes. Depending on the background of the student, this missing link can manifest itself in a couple of different ways. Some students will master the theoretical concepts and notation, but struggle in a computing environment. Others will feel very comfortable with statistical programming, but sometimes struggle to translate that back to the classroom probability setting.

discreteRV is an attempt to bridge this gap. It provides a comprehensive set of functions to create, manipulate, and simulate from discrete random variables. It uses syntax which in most cases closely matches probability textbooks to allow for a more seamless connection between a probability classroom setting and the use of statistical software. **discreteRV** is available for download on the Comprehensive R Archive Network (CRAN).

The functions of **discreteRV** are organized into two logical areas, termed probabilities and simulations.

Probabilities

discreteRV includes a suite of functions to create, manipulate, and compute distributional quantities for the random variables defined. A list of these functions and brief discriptions of their functionality is available in Table 1. In Table 2, the notational similarities between **discreteRV** and the commonly used book by Casella and Berger (2001) is shown.

Creating random variables

The centerpiece of **discreteRV** is a set of functions to create and manipulate discrete random variables. A random variable X is defined (see e.g. Wild and Seber, 1999) as a theoretical construct representing the value of an outcome of a random experiment. A discrete random variable is a special case that can only take on a countable and finite set of values. Discrete random variables are associated with probability mass functions, which map the set of possible outcomes of the random experiment to probabilities which must sum to one. Throughout this document, we will work with a simple example of a discrete random variable representing the value of a roll of a fair die. Formally, we can define such a random variable and its probability mass function as follows:

Let X be a random variable representing a single roll of a fair die. Then,

$$f(x) = P(X = x) = \begin{cases} \frac{1}{6} & x \in 1,2,3,4,5,6 \\ 0 & \text{otherwise} \end{cases}$$

In **discreteRV**, a discrete random variable is defined through the use of the `make.RV` function. `make.RV` accepts a vector of probabilities and a vector of outcome values, and returns an `RV` object. Consider our example of an RV object `X` which we will use to represent a single roll of a fair die. The code to create such a random variable is as follows:

Name	Description
Creation	
`as.RV`	Turn a probability vector with possible outcome values in the names() attribute into a random variable
`make.RV`	Make a random variable consisting of possible outcome values and their probabilities or odds
Manipulation	
`margins`	Marginal distribution of a joint random variable
`mult`	Joint probability mass function of random variables X and Y
`multN`	Probability mass function of X^n
`SofI`	Sum of independent random variables
`SofIID`	Sum of independent identically distributed random variables
Probabilities	
`E`	Expected value of a random variable
`KURT`	Kurtosis of a random variable
`P`	Calculate probabilities of events
`probs`	Probability mass function of random variable X
`SD`	Standard deviation of a random variable
`SKEW`	Skewness of a random variable
`V`	Variance of a random variable
Methods	
`plot.RV`	Plot a random variable of class RV
`print.RV`	Print a random variable of class RV
`qqnorm.RV`	Normal quantile plot for RVs to answer the question how close to normal it is

Table 1: Overview of functions provided in **discreteRV** ordered by topics.

discreteRV	Casella and Berger
`E(X)`	E(X)
`P(X == x)`	$P(X = x)$
`P(X >= x)`	$P(X \geq x)$
`P((X < x1) %AND% (X > x2))`	$P(X < x_1 \cap X > x_2)$
`P((X < x1) %OR% (X > x2))`	$P(X < x_1 \cup X > x_2)$
`P((X == x1) | (X > x2))`	$P(X < x_1 \| X > x_2)$
`probs(X)`	$f(x)$
`SD(X)`	$sd(X)$
`V(X)`	$var(X)$

Table 2: Probability functions in **discreteRV** and their corresponding syntax in introductory statistics courses.

```
X <- make.RV(vals = 1:6, probs = rep("1/6", times = 6))
X

## random variable with 6 outcomes
##
##   1   2   3   4   5   6
## 1/6 1/6 1/6 1/6 1/6 1/6
```

Structure

The syntactic structure of the included functions lends itself both to a natural presentation in an introductory probability course, as well as more advanced modeling of discrete random variables. The object is constructed by setting a standard R vector object to the possible values that the random variable can take (the sample space). It is preferred, though not required, that these be encoded as integers, since this allows to compute expected values, variances, and other distributional properties. This vector of outcomes is then named by the respective probability of each outcome. The probability can be encoded as a string, such as "1/6", if this aids in readability, but the string must be coercable to a numeric.

The choice to encode the probabilities in the names of the vector may seem counterintuitive. However, it is this choice which allows for the familiar syntax employed by introductory statistics courses and textbooks to be seamlessly replicated.

Note that although the print method does not illustrate the inherent structure of the object, the probabilities (1/6, for each of the 6 outcomes of the die roll) are actually stored in the names of the object `X`.

```
names(X)

## [1] "1/6" "1/6" "1/6" "1/6" "1/6" "1/6"
```

Probabilities

By storing the outcomes as the principle component of the object X, we can now make a number of probability statements in R. For instance, we can ask what the probability of obtaining a roll greater than 1 is by using the code `P(X >1)`. R will check which values in the vector X are greater than 1. In this case, these are the outcomes 2, 3, 4, 5, and 6. Hence, R will return TRUE for these elements of X, and then we can encode a function P to compute the probability of this occurrence by simply summing over the probability values stored in the names of these particular outcomes. Likewise, we can make slightly more complicated probability statements such as $P(X > 5 \cup X = 1)$, using the `%OR%` and `%AND%` operators.

In addition, conditional probabilities are also supported. For instance, to calculate the probability of obtaining a roll of one given that the roll is no more than 3, you would use the code `P(X == 1 | X <= 3)`. The use of the pipe operator may be less intuitive to the seasoned R programmer, but overcomes a major notational issue in that conditional probabilities are most commonly specified with the pipe. This issue led to the creation of the `%OR%` and `%AND%` operators specified previously.

Several other distributional quantities are computable, including the expected value and the variance of a random variable. As in notation from probability courses, expected values can be found with the `E` function. To compute the expected value for a single roll of a fair die, we run the code `E(X)`.

Joint Distributions

Aside from moments and probability statements, **discreteRV** includes a powerful set of functions used to create joint probability distributions. Once again letting X be a random variable representing a single die roll, we can use the `multN` function to compute the probability mass function of n trials of X. Table 3 gives the first eight outcomes for $n = 2$, and Table 4 gives an the first eight outcomes for $n = 3$. Notice again that the probabilities have been coerced into fractions for readability. Notice also that the outcomes are encoded by the outcomes on each trial separated by a period.

discreteRV also includes functions to compute the sum of independent random variables. If the variables are identically distributed, the `SofIID` function can be used to compute probabilities for the sum of n independent realizations of the random variable. In our fair die example, `SofIID(X,2)` would create a random variable object with the representation given in 5

Outcome	1,1	1,2	1,3	1,4	1,5	1,6	2,1	2,2
Probability	1/36	1/36	1/36	1/36	1/36	1/36	1/36	1/36

Table 3: First eight Outcomes and their associated Probabilities for a variable representing two independent rolls of a die.

Outcome	1,1,1	1,1,2	1,1,3	1,1,4	1,1,5	1,1,6	1,2,1	1,2,2
Probability	1/216	1/216	1/216	1/216	1/216	1/216	1/216	1/216

Table 4: First eight Outcomes and their associated Probabilities for a variable representing three independent rolls of a die.

Plotting

discreteRV includes a `plot` method for random variable objects so that a visualization of the outcomes and probabilities can be made simply by calling `plot(X)`. The result of plotting the random variable representing a fair die is given in Figure 1. The x axis includes all outcomes, and the y axis includes the probabilities of each particular outcome. The result of plotting a random variable representing the sum of two independent rolls of a die is given in Figure 2. The result of plotting a random variable representing the sum of 100 independent rolls of a die is given in Figure 3.

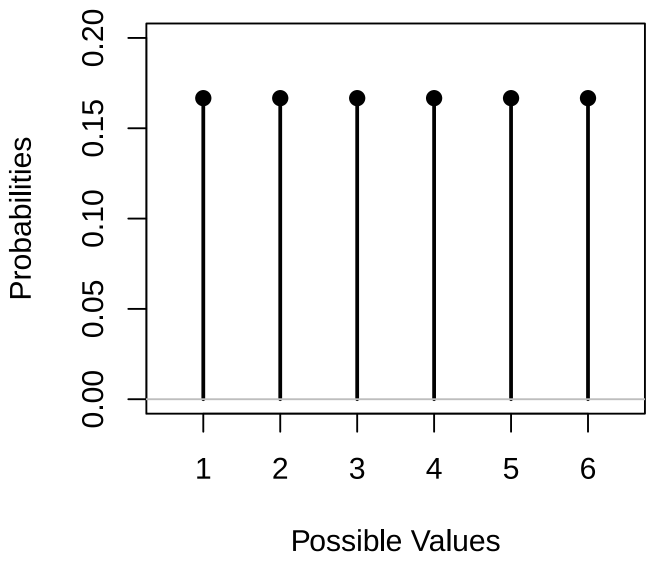


Figure 1: Plot method called on a fair die random variable.

Outcome	2	3	4	5	6	7	8	9	10	11	12
Probability	1/36	1/18	1/12	1/9	5/36	1/6	5/36	1/9	1/12	1/18	1/36

Table 5: Outcomes and their associated Probabilities for a variable representing the sum of two independent rolls of a die.

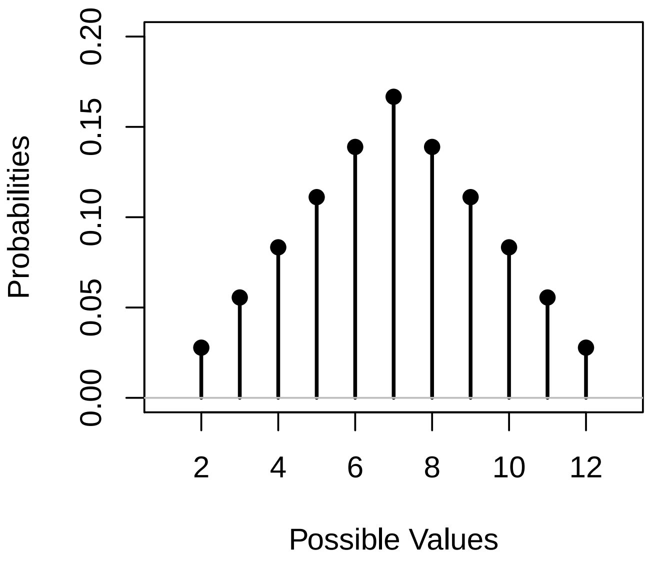


Figure 2: Plot method called on a sum of two fair die random variable.

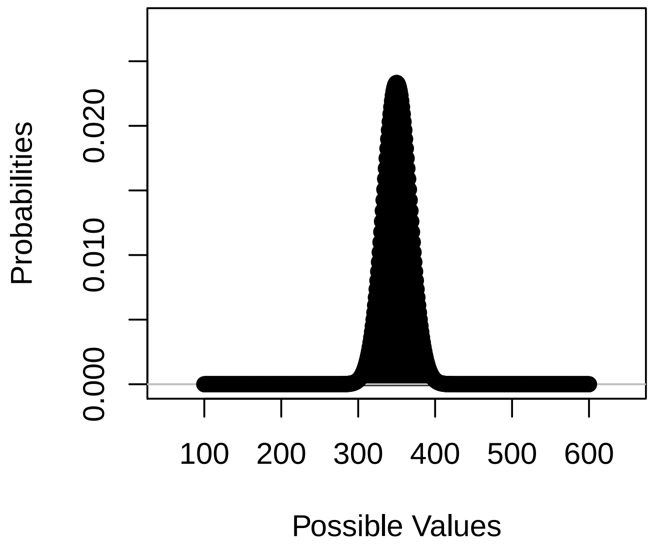


Figure 3: Plot method called on a sum of 100 fair die random variable.

In addition to a plotting method, there is also a method for `qqnorm` to allow assessment of normality for random variable objects, as displayed in Figure 4.

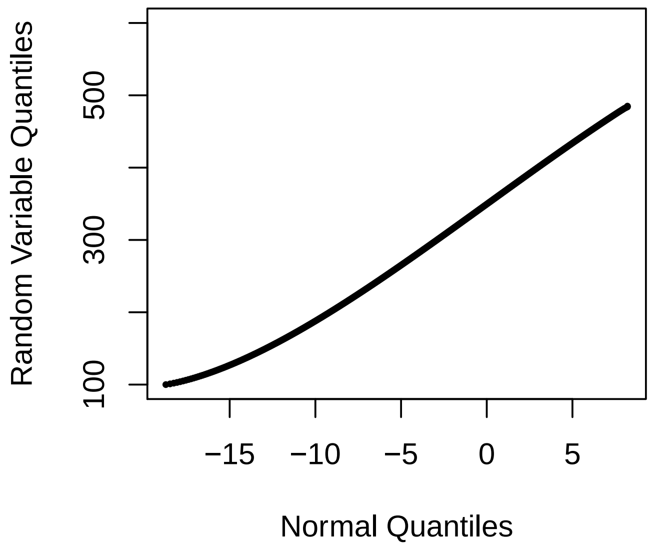


Figure 4: qqnorm method called on a sum of 100 fair die random variable.

Simulation

discreteRV also includes a set of functions to simulate trials from a random variable. A list of these functions and brief discriptions of their functionality is available in Table 6.

Name	Description
plot.RVsim	Plot a simulated random vector
Prop	Proportion of an event observed in a vector of simulated trials
props	Proportions of observed outcomes in one or more vectors of simulated trials
rsim	Simulate n independent trials from a random variable X
skewSim	Skew of the empirical distribution of simulated data

Table 6: List of the simulation functions contained in **discreteRV**.

Creation

Creating a simulated random vector is done by using the `rsim` function. `rsim` accepts a parameter n representing the number of independent trials to simulate, and a parameter X representing the random variable with which to simulate from. For example, suppose we'd like to simulate ten trials from a fair die. We have already created a random variable object X, so we simply call rsim as follows:

```
X.sim <- rsim(10, X)
X.sim

## 1/6 1/6 1/6 1/6 1/6 1/6 1/6 1/6 1/6 1/6
##   4   1   1   5   5   2   5   4   1   3
## attr(,"RV")
## random variable with 6 outcomes
##
##   1   2   3   4   5   6
## 1/6 1/6 1/6 1/6 1/6 1/6
## attr(,"class")
## [1] "RVsim"
```

The object returned is a vector of simulated values, with a class attribute to contain the random variable that was used for the simulation. If we would like to retrieve only the simulated values and exclude the attached probabilities, we can coerce the object into a vector using R's built-in `as.vector` function.

```
as.vector(X.sim)

##  [1] 4 1 1 5 5 2 5 4 1 3
```

It is also possible to retrieve some quantities from the simulation. We can retrieve the empirical distribution of simulated values with the `props` function. This will return the outcomes from the original random variable object, and the observed proportion of simulated values for each of the outcomes. We can also compute observed proportions of events by using the `Prop` function. Similar to the `P` function for probability computations on random variable objects, `Prop` accepts a variety of logical statements.

```
props(X.sim)

## RV
##   1   2   3   4   5   6
## 0.3 0.1 0.1 0.2 0.3 0.0

Prop(X.sim == 3)

## [1] 0.1

Prop(X.sim > 3)

## [1] 0.5
```

Conclusion

The power of **discreteRV** is truly in its simplicity. Because it uses familiar introductory probability syntax, it can allow students who may not be experienced or comfortable with programming to ease into computer-based computations. Nonetheless, **discreteRV** also includes several powerful functions for analyzing, summing, and combining discrete random variables which can be of use to the experienced programmer.

Bibliography

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

G. Casella and R. L. Berger. *Statistical Inference*, volume 2. Cengage Learning, 2001. [p1]

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]

C. Wild and G. Seber. *CHANCE ENCOUNTERS: A First Course in Data Analysis and Inference*. John Wiley & Sons, 1999. [p1]

Analyzing Peptide Libraries with peptider

by Eric Riemer 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 trimer 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. 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.

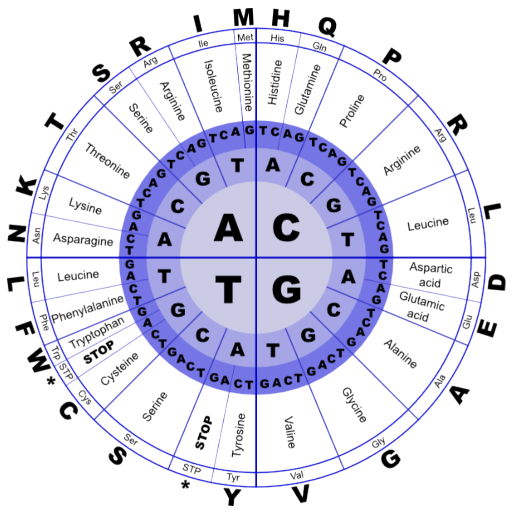


Figure 1: Codon wheel illustrating which bases lead to codon representations for each amino acid.

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

```
scheme("NNN")

##   class    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 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
```

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] 0.2918
```

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.

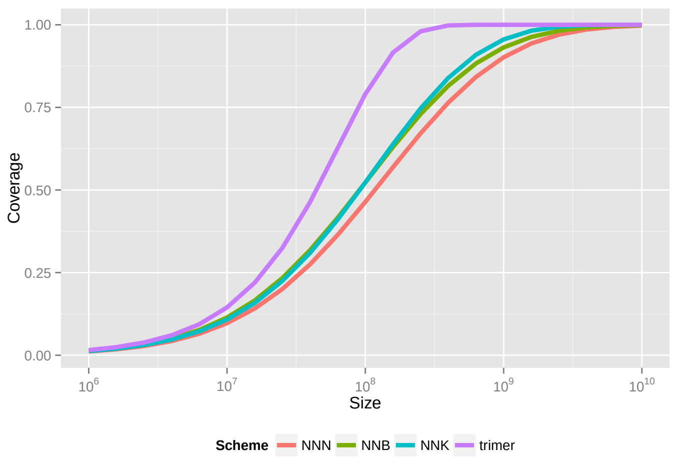


Figure 2: Overview of coverage rates across different hexapeptide library schemes of varying size.

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

## [1] 0.5229
```

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", "trimer"), 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] 0.3347
```

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] 0.5166
```

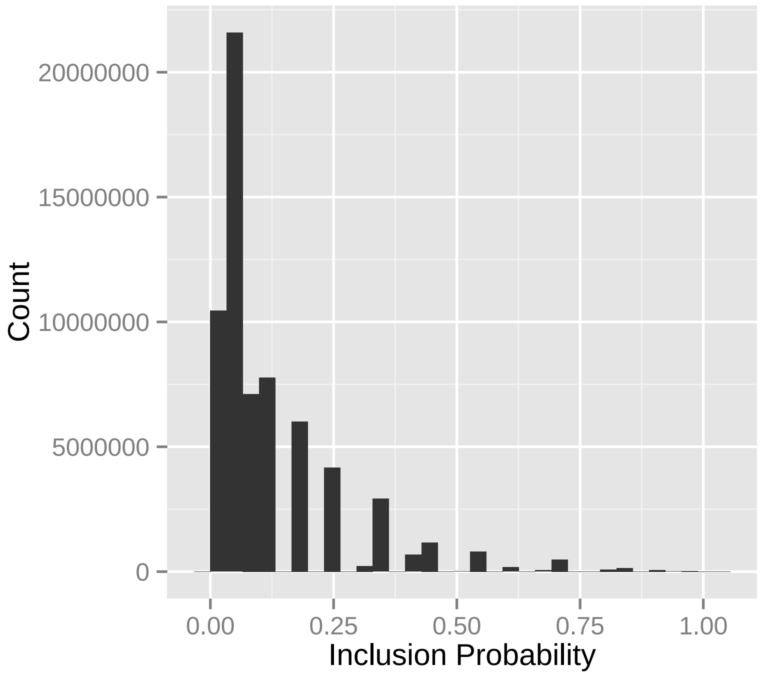
We may also be interested in the range of inclusion probabilities for all peptides rather than just a particular sequence. In this case, the `detect` function is useful. detect differs a bit syntactically as it requires the created library to be passed in as an argument. For an NNK library with peptide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



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.

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^{25} NNK library with peptides of length 18, one can call:

```
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.

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]

A Web Application for Efficient Analysis of Peptide Libraries

Eric Riemer Hare

February 2, 2014

Abstract

Peptide libraries have important theoretical and practical applications in the fields of biology and medicine. Despite their importance, little analysis has been done to assess the statistical properties of different libraries. This paper introduces a web application called PeLiCa, or Peptide Library Calculator, built upon the Shiny web-application framework. PeLiCa provides an easy-to-use and powerful front-end to the R package peptider, allowing users to conduct a statistical analysis of a set of pre-defined peptide library schemes, or a custom-defined scheme of their own choosing. Results are instantly displayed, to help the user make a more informed decision regarding what specific peptide library will be most useful for their particular application or research.

1 Introduction

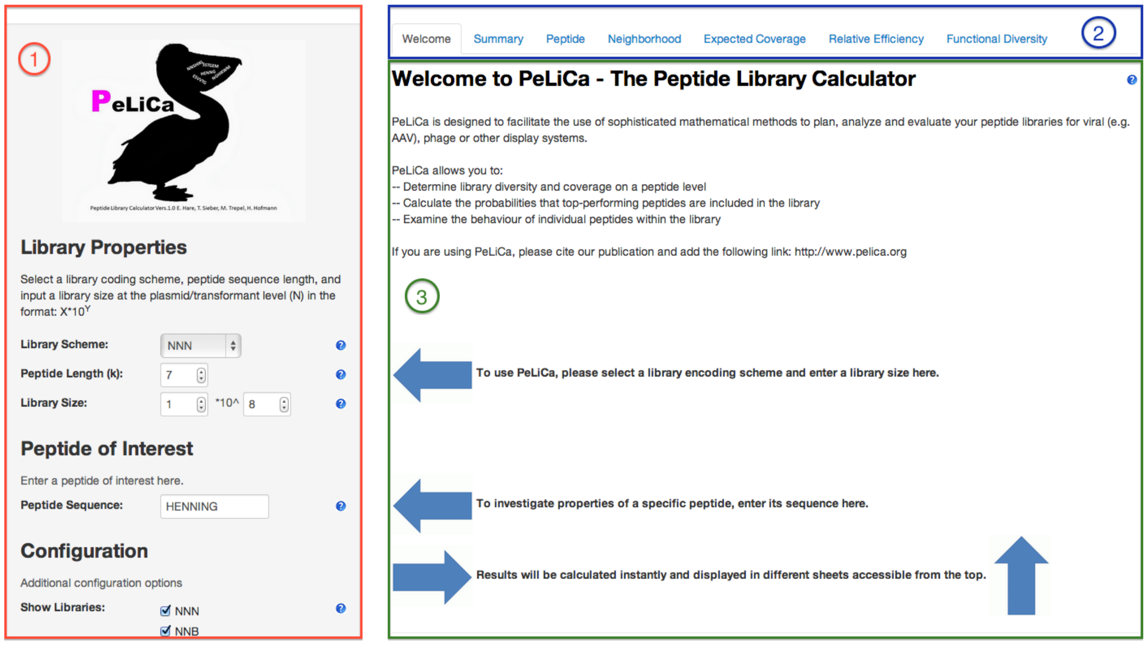
With the introduction of the R package *peptider*, analysis of the statistical properties of various peptide libraries is now possible. However, use of this package still requires familiarity with the R language, and more general programming concepts. In this paper, I introduce a new web interface called *PeLiCa* (or Peptide Library Calculator) which provides a useable and flexible front-end to peptider. PeLiCa is designed for biologists and others working with peptide libraries, and does not require programming knowledge to conduct an analysis. PeLiCa can be accessed from the url `http://www.pelica.org`.

2 Structure

PeLiCa is a Shiny application. Shiny is a framework for writing web-applications in the R language, requiring little-to-no javascript programming knowledge. PeLiCa uses this framework to provide interactivity. For instance, when the user of PeLiCa changes a property of the peptide library, such as the encoding scheme, the results, tables, and plots will instantly update to reflect the new library. PeLiCa is currently hosted on the Glimmer server provided by the RStudio team.

3 User Interface

Similar to other Shiny applications, PeLiCa consists of three primary UI components, the Configuration Panel, the Tab Panel, and the Results Panel, each illustrated in Figure 1. The Configuration Panel is located along the left-hand column. This panel allows for various parameters of peptide libraries to be adjusted, and some configuration options relating to PeLiCa to be changed depending on user preference. The top panel is the Tab Panel, which contains various tabs corresponding to different properties of peptide libraries that can be investigated. The bottom panel is the Results panel, which will contain the results of the analysis depending on the tab and configuration options selected.



Library Properties

Select a library coding scheme, peptide sequence length, and input a library size at the plasmid/transformant level (N) in the format: X*10^{Y}

Library Scheme:	NNN
Peptide Length (k):	7
Library Size:	1 *10^ 8

Peptide of Interest

Enter a peptide of interest here.

Peptide Sequence:	HENNING

Configuration

Additional configuration options

Show Libraries: ✓ NNN ✓ NNB

Welcome to PeLiCa - The Peptide Library Calculator

PeLiCa is designed to facilitate the use of sophisticated mathematical methods to plan, analyze and evaluate your peptide libraries for viral (e.g. AAV), phage or other display systems.

PeLiCa allows you to:
-- Determine library diversity and coverage on a peptide level
-- Calculate the probabilities that top-performing peptides are included in the library
-- Examine the behaviour of individual peptides within the library

If you are using PeLiCa, please cite our publication and add the following link: http://www.pelica.org

To use PeLiCa, please select a library encoding scheme and enter a library size here.

To investigate properties of a specific peptide, enter its sequence here.

Results will be calculated instantly and displayed in different sheets accessible from the top.

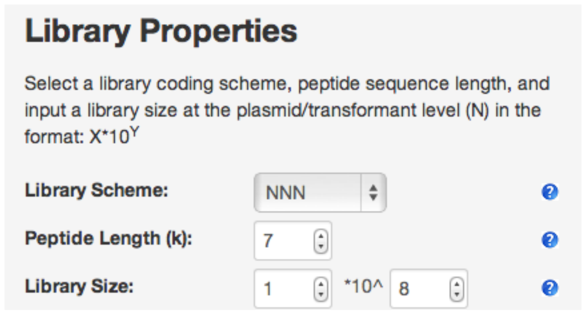
Figure 1: Screenshot of PeLiCa indicating the three primary UI components. (1) The Configuration Panel (2) The Tab Panel (3) The Results Panel.

Another important component of the user interface are the help tooltips. Throughout the application, blue question marks will be available. When the user moves their mouse cursor over these icons, helpful tooltips will appear to instruct the user on how to proceed, or provide more information about the library property being investigated.

4 Features

4.1 Configuration Panel

The first set of features available in PeLiCa involve the specification of properties of the library of interest. These features are displayed in Figure 2. Users can first select a library scheme. PeLiCa provides several built-in library schemes, the first of which is the NNN scheme, in which all four bases (Adenine, Guanine, Cytosine, and Thymine) can occur at all three positions in a particular codon. The second is the NNB scheme, where the first two positions are unrestricted, but the third position can only be three bases. The third is NNK/S, which covers both NNK and NNS schemes, with the third position restricted to two bases. Both NNK and NNS have identical statistical properties in the analysis available in PeLiCa[REF]. Finally, there are trimer-based libraries in which the codons are pre-defined. PeLiCa also includes variations of these four library types in which Cysteine is treated as a non-viable amino acid.


Library Properties

Select a library coding scheme, peptide sequence length, and input a library size at the plasmid/transformant level (N) in the format: X*10^{Y}

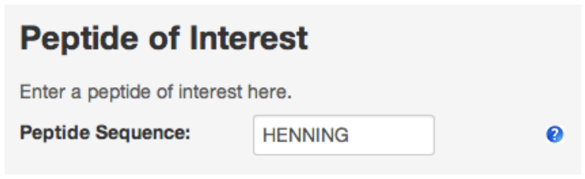
Library Scheme:	NNN
Peptide Length (k):	7
Library Size:	1 *10^ 8

Figure 2: The Library Properties section of the Configuration Panel.

PeLiCa also has support for user-defined library schemes. If the user selects "Custom Scheme" for the library scheme, they will be presented with an upload dialog, along with a set of instructions for uploading a custom scheme. Using a custom scheme with PeLiCa requires a minimal use of programming and may be less suitable for those unfamiliar with R.

Users can also specify the peptide length and the library size. Peptide lengths can range from six to ten amino acids, with support for larger peptides coming soon. The library size is specified in scientific notation of the form $x \times 10^y$. The user will specify values for x and y in this equation. x can range from 1.0 to 9.9 in increments of 0.1, and y can range from six to 14 in increments of one, yielding a supported range of library sizes between 1.0×10^6 and 9.9×10^{14}. Support for large library sizes is also in progress.

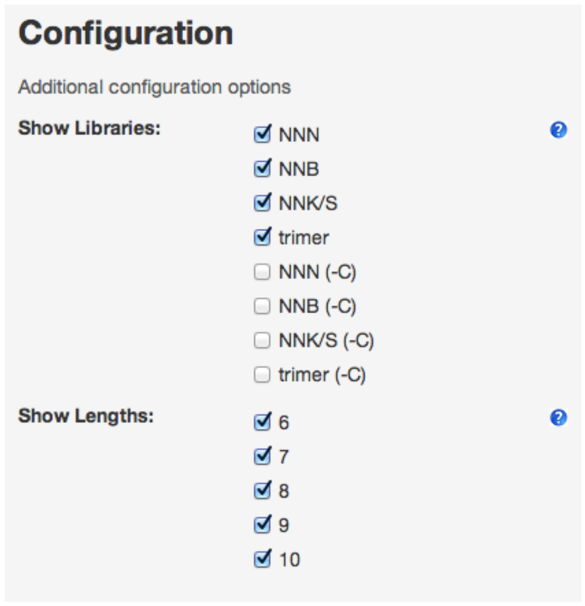
Users can then specify a peptide of interest, as shown in Figure 3, and configure which other libraries to display in the Results Panel in Figure 4.


Peptide of Interest

Enter a peptide of interest here.

Peptide Sequence:	HENNING

Figure 3: The Peptide of Interest section of the Configuration Panel.


Configuration

Additional configuration options

Show Libraries:
- ✓ NNN
- ✓ NNB
- ✓ NNK/S
- ✓ trimer
- ☐ NNN (-C)
- ☐ NNB (-C)
- ☐ NNK/S (-C)
- ☐ trimer (-C)

Show Lengths:
- ✓ 6
- ✓ 7
- ✓ 8
- ✓ 9
- ✓ 10

Figure 4: The Configuration section of the Configuration Panel.

4.2 Results Panel

The primary functionality for PeLiCa is available in the Results Panel, which are accessible through each of the tabs in the Tab Panel.

4.2.1 Welcome

PeLiCa begins on the Welcome tab. The Welcome tab provides information on the functionality of PeLiCa, and a quick guide for its use. The tooltip on this tab illustrates the system requirements.

4.2.2 Summary

The Summary tab contains most of the key information from the other tabs, condensed into an easy-to-digest format. First, information on your library is displayed. Some of this information includes the coverage, the peptide diversity, and the probabilities of peptide inclusion. Information on your selected peptide is displayed below this, summarizing the inclusion probability and the number of different DNA encodings of this particular peptide. Finally, a table displaying a randomly generated sample of peptides is shown at the bottom. This table includes the amino acids comprosing the peptide, the peptide class under the chosen encoding scheme, the number of DNA encodings, and the probability of inclusion in your library.

4.2.3 Peptide

The peptide tab includes information allowing investigation of the peptide selected in Figure 3. PeLiCa will display the inclusion probability of this peptide, as well as a set of boxplots illustrating the inclusion probability of all peptides. The boxplots allow for a comparison of the relative differences between different schemes and library sizes. An example of this plot is given in Figure 5. In this case, the selected library is an NNN scheme library with peptide length seven, and a library size of 10^8. The probability of inclusion is given across different library sizes, with the particular peptide of interest displayed as a circle on the plot.

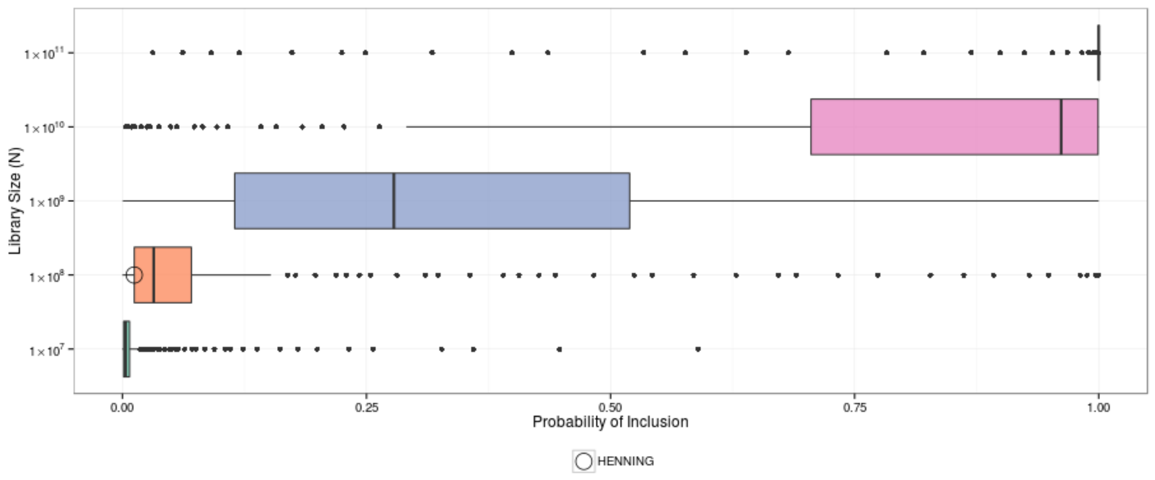


Figure 5: Boxplots displayed in PeLiCa representing the inclusion probabilities for an NNN library with peptide length seven and a library size of 1×10^8. The inclusion probability of HENNING is displayed in the plot as a circle.

4.2.4 Neighborhood

The neighborhood tab provides information on the inclusion probabilities of peptides in the degree-one and degree-two neighborhoods of the selected peptide. It also provides a range of inclusion probabilities for degree-one and degree-two neighborhoods of all peptides. As for the Peptide tab, plots of the inclusion probabilities across different library sizes and schemes is displayed to allow for a quick comparison.

4.2.5 Expected Coverage

The Expected Coverage tab displays a numerical value for the expected coverage of your library, and a plot of coverages for different schemes and library sizes. An example of the plot, using an NNN library with peptide length seven and library size 10^8, is shown in Figure 6.

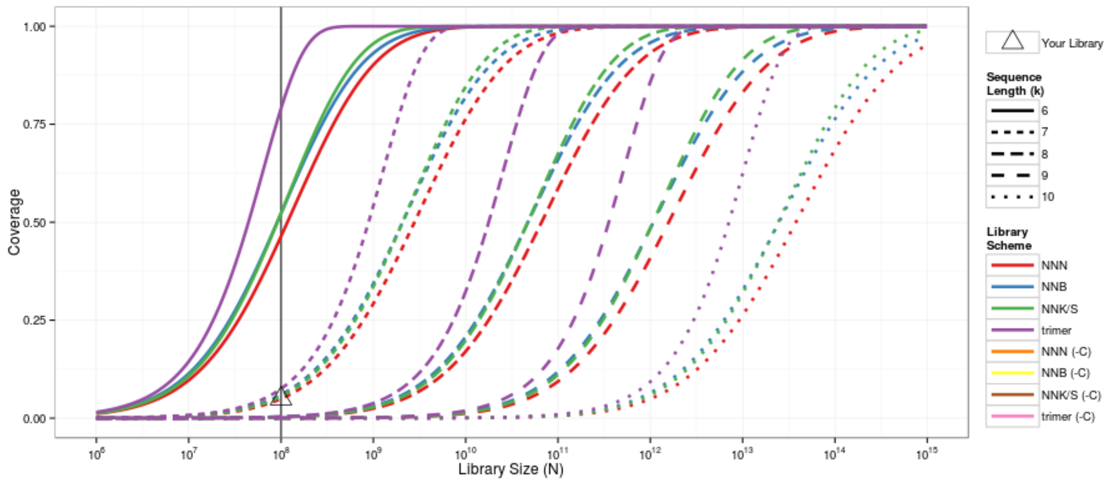


Figure 6: Plots of expected coverage for an NNN library with peptide length seven and a library size of 1×10^8

4.2.6 Relative Efficiency

Similar the the Expected Coverage tab, the Relative Efficiency tab provides a numerical value for the relative efficiency, and a comparison of the selected library to other library schemes and sizes. In conjunction with expected coverage, this allows an optimization of the cost-benefit properties of the library. As relative efficiency decreases with the library size, a library which optimizes the relative efficiency while still maintaining a desired coverage level will set a bound on the size of the library. This will help to identify a library size and scheme that has the diversity properties desired and is not prohibitively expensive.

4.2.7 Functional Diversity

The final tab in PeLiCa is the Functional Diversity tab, which displays both the functional diversity of your library, and a table of the functional diversity for other schemes and peptide lengths (Note that the functional diversity does not depend on the size of the library).

5 Further Work

A new version of PeLiCa is in currently in progress. The new version supports lower resolution monitors, includes a new framework for tooltips, and supports a wider range of peptide lengths and library sizes. This new version is currently deployed on ShinyApps at `http://erichare.shinyapps.io/pelica`.

6 Conclusion

By utilizing the R package *peptider* and the web application framework Shiny, PeLiCa allows for a powerful statistical analysis of peptide libraries. It is flexible enough to allow investigation of a wide variety of different library schemes, peptide lengths, and library sizes. Still, the application is web-based and easy to use, making the barrier of entry for those outside the field of statistics very low.

References

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

[2] Heike Hofmann, Eric Hare, and ggobi Foundation. *peptider: Evaluation of diversity in nucleotide libraries*. R package version 0.1.2. 2013.

[3] 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/`.

[4] Timo Sieber et al. “Biomathematical Description of Peptide Library Properties”. In: *Bioinformatics* (2014).

BIOINFORMATICS

Vol. 00 no. 00 2013
Pages 1–8

Biomathematical Description of Peptide Library Properties

Timo Sieber [1,*], Eric Hare [2], Heike Hofmann [2] and Martin Trepel [1*]

[1]University Medical Center Hamburg-Eppendorf, Department of Oncology and Hematology, 20246 Hamburg, Germany.
[2]Department of Statistics, Iowa State University, Ames IA 50011-1210, United States.

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Motivation: Libraries of randomised peptides displayed on phages or viral particles are essential tools in a wide spectrum of applications. However, there is only limited understanding of a library's fundamental dynamics and the influences of encoding schemes and sizes on their quality measured as peptide diversity.

Results: We are presenting a mathematical framework to derive the expected number of different peptides expressed in specified libraries. We also define coverage and relative efficiency, which allows researchers to describe libraries in enough detail to plan new experiments in a more informed manner. In particular, these values allow an answer to "What are the chances that a library contains one of the 'best' possible peptides?"

Availability: The framework is implemented in two freely available packages to the statistical software environment `R`: `discreteRV` and `peptider`. A graphical user interface implementing all aspects is made available at the PeLiCa website `http://www.pelica.org/`.

Contact: t.sieber@uke.de

Supplementary Information: At Bioinformatics online

1 INTRODUCTION

Since the year 2000, almost 500 publications per year have been published in which peptide libraries have been used (PubMed query November 2013 on *"peptide library"*), reflecting the importance of such libraries as tools for a wide spectrum of biological applications ranging from the identification of protein interaction sites (e.g. Rodi *et al.*, 1999) and the development of enzyme inhibitors (e.g. Lu *et al.*, 2012) to identification of peptides that mediate cell type specific gene delivery by viral vector systems (e.g. Müller *et al.*, 2003). For such studies, randomised oligonucleotides are introduced into plasmids encoding structural proteins of bacteriophages (Binder *et al.*, 2011) or viruses, such as adeno-associated viruses (Müller *et al.*, 2003), adenoviruses (Nishimoto *et al.*, 2012) or retroviruses (Bupp and Roth, 2003). These plasmids are ligated and transformed into bacteria to generate a plasmid library, which then is used to produce virus or phage libraries. These can be utilised in a variety of selection procedures, aiming to isolate peptide bearing viruses and phages with desired properties or scaffold independent, functional peptides (e.g. peptide inhibitors; Lu *et al.*, 2012). The success of this method is highly dependent on the diversity of the initial peptide pool, as the chance to identify the "best possible" sequence, or even a suitable sequence, directly correlates with the number and diversity of the peptides contained in the library used for the screening procedure.

However, determining the diversity of a library is problematic, as the number of distinct peptides cannot be measured easily. Even with the advent of widely accessible next-generation sequencing, the size of current libraries (e.g. 2×10^{10} clones; Deshayes *et al.*, 2002) makes the use of this technique generally impracticable due to time and financial effort necessary to reach the very high sequencing depth necessary to gain sufficient sequencing coverage. Therefore, several attempts to describe diversity of peptide libraries by other means have been made. DeGraaf *et al.* (1993) show the diversity of their phage decapeptide display library by estimating the distribution of single amino acids and dipeptides in a sample. While this is a useful way to show that a population is diverse, it does not quantify the diversity nor does this method give any information about the actual number of distinct peptides in the library.

Rodi *et al.* (2002) present another approach by defining *functional diversity* as a measure to describe the quality of a peptide library. The functional diversity reflects the distribution of the different peptides encoded in the library. If every peptide has the same frequency, the functional diversity is at its maximum (set to 1). With increasingly skewed distributions, this value drops towards zero. Functional diversity does not reflect the actual number of sequences in a library but describes diversity at a theoretical level based on specific peptide length and encoding scheme. *Clonal diversity* is another approach to describe diversity at the level of the plasmid library by counting successfully transformed bacterial colonies (e.g. Noren and Noren, 2001; Michelfelder *et al.*, 2009). This number is easily assessable, and represents the maximally achievable diversity for the phage/virus library, as the diversity cannot be increased after the cloning and transformation process. Particular precautions must be taken to avoid – or at least, to minimise – losses to diversity in subsequent steps of the library production to make the clonal diversity a valid qualifier for the peptide library. Clonal diversity on its own is of limited value, as the relevant metric is the diversity of the presented peptides (*peptide diversity*). However, both values are connected and clonal diversity can be used to determine peptide diversity if certain considerations are taken into account: peptide diversity of the library is always lower than clonal

*to whom correspondence should be addressed diversity, due to the possibility that different bacterial clones encode identical peptides. This is caused either by several clones containing identical peptide encoding DNA, or by clones harboring distinct DNA sequences that encode the same peptide. The reason for the latter possibility is the degenerate nature of the genetic code: amino acids can be encoded by up to six distinct codons, and therefore the same peptide can be described by multiple DNA sequences. This has the effect that, for instance, a pool of randomised seven codon DNA sequences has a nominal diversity of 64^7 (64 codons; 4.4×10^{12}) while it encodes only 23^7 (20 amino acids and three stop codons; 3.4×10^9) distinct amino acid sequences. Further, stop codons in the random nucleotide sequence prematurely terminate the peptide and can cause dysfunctional proteins in display systems (Lindner *et al.*, 2011; Michelfelder and Trepel, 2009).

Libraries can also be encoded by limited subsets of the standard 64 codons to at least partially counteract both effects (as also discussed in Patrick and Firth, 2005). Instead of the NNN scheme, where "N" represents any of the four bases, encoding schemes like NNB, NNK or NNS (B: C/G/T; K: G/T; S: G/C) can be used. These schemes encode all twenty amino acids and one stop codon each, while the total number of codons is reduced to 48 (NNB) and 32 (NNK and NNS), respectively. Another approach are trimer libraries (Kayushin *et al.*, 1996). Here, oligonucleotides are synthesised by assembling prefabricated trinucleotide phosphoramidites or trimers. This allows a ratio of one codon per amino acid for all amino acids and a complete avoidance of stop codons. It has been shown that such libraries possess increased functional diversity in phage display (Krumpe *et al.*, 2007).

Another important consideration regarding peptide diversity are cysteines. Pairs of cysteines flanking randomised sequences are often used in phage display as they form controlled disulfide bridges that enhance half-lives and binding characteristics of the library peptides (McConnell *et al.*, 1994). However, random integration of odd numbers of cysteines has repeatedly been shown to inhibit the generation of peptide bearing phages (reviewed by Fukunaga and Taki, 2012). Further, even though the situation is less well understood for other display systems, a strong underrepresentation of cysteine-containing peptides was observed in peptide libraries on different AAV vectors (Waterkamp *et al.*, 2006; Naumer *et al.*, 2012; Perabo *et al.*, 2006; Varadi *et al.*, 2012). This again suggests unfavorable effects of cysteine incorporation on basic functions of the display system. In line with this is the notable lack of capsid surface-exposed cysteine residues on wild type AAV2 (Xie *et al.*, 2002). Also, the surface of human Adenovirus type 5 is naturally devoid of cysteines. If they are artificially integrated, the particles were shown to be prone to aggregation due to the formation of interparticle disulfide bridges (Kreppel *et al.*, 2005).

With regard to the aforementioned factors, we will determine peptide diversity by using clonal diversity, but consider effects of encoding schemes and stop codons. For the purpose of discussing diversity, we will consider cysteine-containing peptides as non-functional. Diversity discussions considering cysteines are available at our website PeLiCa. Other biological restraints that negatively affect peptide diversity do exist, but are not taken into account here, as they are largely unknown and highly dependent on the individual system and its specific characteristics, such as the differences between distinct incorporation sites (Naumer *et al.*, 2012; Girod *et al.*, 1999). However, depending on the system and its intended use (e.g. generation of a functional viral vector with peptide mediated tropism), compatibility with such restrictions might be considered as a first step in the selection process.

Due to the sheer number of peptides possible in a library, determining the peptide diversity is a mathematically taxing problem that becomes ever more challenging with increasing peptide length. In this publication, we introduce a mathematical framework capable of facilitating this task. As the quality of a peptide library is not only defined by the peptide diversity, we further use the concepts of *coverage* and *efficiency* to allow a detailed evaluation of libraries. Further, we discuss effects of insert length, different encoding schemes (NNN, NNB, NNK, NNS, and trimer), and answer one of the important questions for researchers working with peptide libraries: "What are the chances that my library contains one of the 'best' possible peptides?"

In contrast to other methods describing library diversity, our framework allows a systematic analysis of the behavior of large peptide libraries, which in turn facilitates a deeper understanding and allows for a more informed planning of new, optimised libraries. To make the framework easily accessible, we generated a user-friendly web-interface called PeLiCa (available at `http://www.pelica.org`), which allows the user to determine all of these factors for libraries of sizes 10^6 to 10^{15} bacterial clones, using different encoding schemes (including custom-designed schemes and those that consider cysteine viable) and peptide lengths (6 to 10 aa).

2 MEASURING DIVERSITY

While not studied in detail for peptide libraries, studies on diversity at the amino acid level have been performed in the related field of site saturation mutagenesis generated protein libraries. Here, proteins are mutated at a limited number of positions to detect variants with improved properties. The program "GLUE-IT" (available at `http://guinevere.otago.ac.nz/stats.html`) generates values for diversity and coverage for protein libraries with up to six modified codons per protein (Firth and Patrick, 2008). Though this program was designed for another purpose and does not allow evaluation of cysteines as disruptive, it can also be used to gain some information for peptide libraries with short peptides. However, it is insufficient to describe most libraries currently used, which are generally longer and range from five up to twenty or more amino acids in length (e.g. Naumer *et al.*, 2012; Fukunaga and Taki, 2012; Scholle *et al.*, 2005).

In our approach to develop a mathematical framework for the characterization of peptide libraries we use three basic assumptions of randomness: (I) *the randomised oligonucleotides are in fact completely random (in a statistical sense).* (II) *the used pool of oligonucleotides can contain the same nucleotide sequence multiple times as its synthesis is completely random and* (III) *the technical processes that limit the number of DNA sequences in the libraries work at random, to the effect that each nucleotide sequence has an equal chance of getting selected into the library.*

We define and discuss three measures of library quality: *peptide diversity*, defined as the number of distinct peptides in a library, *expected coverage*, describing the fraction of all theoretically possible peptide sequences covered by the library, and *relative efficiency* given as the ratio of the expected number of distinct peptides in a library relative to the overall number of encoding oligonucleotides. We investigate these measures for NNN, NNK, NNS, NNB and trimer encoding schemes. Trimer libraries can be constructed with any number of selected codons. In this publication, we regard them as composed of peptides assembled from 19 distinct codons (one per amino acid; excluding cysteine).

2.1 Libraries with equal codon representations

An easily tractable case for determining diversity is a setting in which all different sequences have the same probability of being included in the library. This can be assumed if diversity is investigated at DNA level or for the special case of trimer libraries in which every amino acid has the same number of codon representations. In that case, calculating expected peptide diversity of a library is relatively simple: the probability that a peptide is present in the library is determined by the maximum number of different peptide sequences and the size of the library. Denote the number of all different possible peptides in the library by b, the size, measured as clonal diversity, of the library by N.

The number of different peptides Z that can actually be achieved in the library is the primary point of interest. In practice, this will differ from library to library, but we can give an expected value $E[Z]$ describing this diversity.

THEOREM 1. *For a library of size n chosen from a scheme with b different peptides, the expected number of different peptides in the library is given as*

$$E[Z] \approx b(1 - e^{-N/b}). \tag{1}$$

The approximation becomes more accurate as values of b and $|\log(N/b)|$ increase. The proof and a more detailed discussion of the approximation error can be found in the supplementary material, parts A.1 and A.2.

In investigating DNA diversity in site saturation mutagenesis libraries, other groups (Patrick *et al.*, 2003; Bosley and Ostermeier, 2005) obtained the same result as Theorem 1 based on a Poisson approximation. While this approach is usable for an analysis at the DNA level or trimer libraries, it cannot be used directly for library schemes in which the number of codons per amino acid varies, because in this case, the probability that a peptide will be included in the library varies in dependence of its sequence. In a standard 64 codon based library there are one to six codons describing individual amino acids (aa). Therefore, some peptide sequences like `SLRLLRS` are encoded by $6^7 = 279936$ distinct codon sequences, as each amino acid in the sequence has six independent possibilities to be encoded. At the other end of the scale, there are peptides that are encoded by a single nucleotide sequence. We will therefore partition the overall library into classes of peptides that all have the same number of encodings (similar conceptual approaches have previously been mentioned, e.g. Firth and Patrick, 2008; Scott and Smith, 1990) and determine overall diversity based on diversity seen within each of these classes. For that, we need to specify the library under observation in more detail.

2.2 Partitioning of Peptide Libraries

To be able to determine the peptide diversity, we have to partition the libraries. In the following, we focus on the 32 codon-based encoding schemes NNK and NNS. All other schemes work similarly, an overview of the class partitioning is given in tables 1 and 2 of the supplementary material part B, According to their multiplicity and functionality NNK and NNS are equivalent, and we can distinguish four classes of aa based on a common NNK/S scheme (see Table 1). Amino acids are given in single letter code. Size s defines

Table 1. NNK/S Library Scheme

aa class	amino acids	size s	# codons c
A	S, L, R	3	3
B	A, G, P, T, V	5	2
C	D, E, F, H, I, K, M, N, Q, W, Y	11	1
Z	cysteine C, stop `TAG`	2	1

the number of different amino acids in an aa class, the number of codons, c, reflects how many codons describe each amino acid in the class. Classes A to C contain all codons for feasible amino acids, while class Z contains corruptive codons. The number of valid classes is therefore $v = 3$.

As discussed earlier, stop codons as well as cysteines are treated as nonviable amino acids (aa class 'Z'); sequences containing one or more of these codons will therefore be excluded.

We are now employing a two-step analysis to retrieve all the relevant probabilistic information to calculate peptide diversity in the resulting library: (I) In a first step we are only interested in whether the outcome is a *valid sequence*, defined to be the case that there is no element of the newly defined aa class Z in the sequence. Valid sequences are therefore those that are expected to be functional in the biological system. (II) In a second step we will investigate the diversity among the remaining peptide sequences.

Any peptide sequence containing a member of aa class Z is by definition not useful for further analysis. In a randomly generated NNK/S library of heptapeptides, these make up 36.35% $= 1 - (1 - P(Z))^7$ of the total. We will call this percentage of invalid sequences the *initial loss* and restrict our analysis to valid sequences only.

Analysing peptide sequences directly is too computationally complex of a problem. In order to reduce this complexity, we only differentiate between peptide sequences at the level of the previously introduced classes. If this is performed for an exemplary library of dipeptide sequences, we have a set of nine different peptide classes (peptides composed of aa classes A to C) corresponding to valid peptide sequences based on the classification of amino acids according to Table 1 (as shown in Table 2). Within each of these aa classes, all peptides have an equal number of oligonucleotide sequence representations. The total of all valid peptide sequences adds up to $19^2 = 361$ different peptide sequences of length two.

The peptide class completely determines both the number of unique peptides and the number of nucleotide representations for each of the peptide sequences. For a given sequence, let n_A, n_B, and n_C be the number of codons from A, B, and C (the sum of n_A, n_B, and n_C then adds up to the total length of the sequence). Here, s_A, s_B, and s_C represent the number of different amino acids

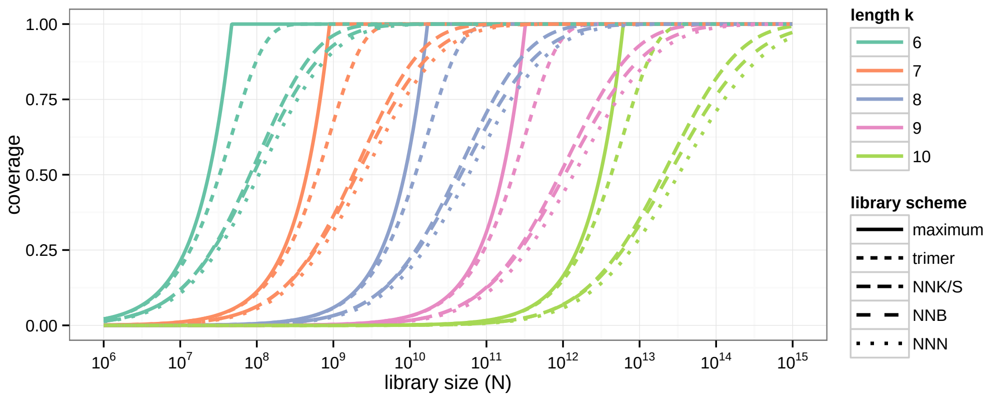


Fig. 1. Overview of expected coverage for k-peptide libraries of different sizes N with the different encoding schemes (NNN, NNB, NNK/S, and trimer). An additional line for maximum possible coverage is shown.

Table 2. Overview of all peptide sequences of length two partitioned according to peptide classes. The peptide class (first line) is defined by the aa class memberships of their codons as defined for NNK/S libraries in Table 1. The number of different unique peptide sequences in each class (second line), and the number of codon representations for each peptide sequence in the class (third line) are given.

peptide class	AA	AB	AC	BA	BB	BC	CA	CB	CC
#peptides	9	15	33	15	25	55	33	55	121
#nucleotides	9	6	3	6	4	2	3	2	1

in aa classes A, B, and C, and c_A, c_B, and c_C stand for the number of codons per amino acid within the corresponding aa class.

The number of peptides (#peptides) and corresponding nucleotide representations for each peptide (#oligonucleotides) is then calculated as

$$\begin{aligned}\text{\#peptides} &= {s_A}^{c_A} \cdot {s_B}^{n_B} \cdot {s_C}^{n_C} \\ \text{\#oligonucleotides} &= {c_A}^{n_A} \cdot {c_B}^{n_B} \cdot {c_C}^{n_C}\end{aligned}$$

The number of oligonucleotide sequences representing a whole peptide class is given as the product of the number of peptides and the number of individual codon representations per peptide. Under the assumption that in a library of peptides with a length of k amino acids all viable codons v (30 codons for NNK/S usage, excluding any class Z codons) are represented with the same probability, this allows us to calculate the probability p for a peptide class to be present in a library as

$$p = \text{\# peptides} \cdot \text{\# oligonucleotides}/v^k. \tag{2}$$

2.3 Diversity in general peptide libraries

Combining the information from individual peptide classes we can determine the diversity in the general peptide library.

For a k-peptide library of size N we expect Np_i sequences to be selected from peptide class i, where p_i is the probability (effectively, the size) of peptide class i. Within this class, all peptides have the same number of oligonucleotides. Assuming b_i different peptides in peptide-class i are theoretically possible, we have, according to theorem 1, a diversity given by the number of different peptides as $b_i(1 - e^{-Np_i/b_i})$, resulting in an overall number of different peptides in the NNK/S library of

$$D(N,k) = \sum_{i=1}^{v^k} b_i(1 - e^{-Np_i/b_i}). \tag{3}$$

Based on the overall peptide diversity, we now define two indices measuring different aspects of quality of k-peptide libraries: expected coverage and relative efficiency.

DEFINITION 1 (Expected coverage). *For a k-peptide library of size N the expected coverage is defined as*

$$C(N,k) = D(N,k)/19^k.$$

Expected coverage is an index in $[0,1]$. 0 indicates that no peptide is in the library (which can only happen for a library of size 0), and 1 indicates that every single possible peptide is included in the library; for this, the size of the library has to be at least $N \geq (19 \text{ viable amino acids})^k$.

Figure 1 shows the expected coverage of k-peptide libraries of sizes between 10^6 and 10^{15} with different encoding schemes. It is obvious that increasing peptide length k has a dramatic negative influence on coverage for a given library size N. Additionally, the used encoding scheme has a profound effect on coverage, with trimer libraries being far superior to the other schemes.

DEFINITION 2 (Relative efficiency). *Relative efficiency is defined as the ratio of expected peptide diversity of a library relative to its overall number of oligonucleotides:*

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

This makes relative efficiency a number between 0 and 1.

Figure 2 gives an overview of relative efficiency of k-peptide libraries of various sizes. In contrast to ideal (maximum) or trimer libraries, libraries encoded by NNK/S, NNB and NNN schemes suffer from an initial loss due to sequences containing aa class Z codons. This limits their maximal relative efficiency depending on

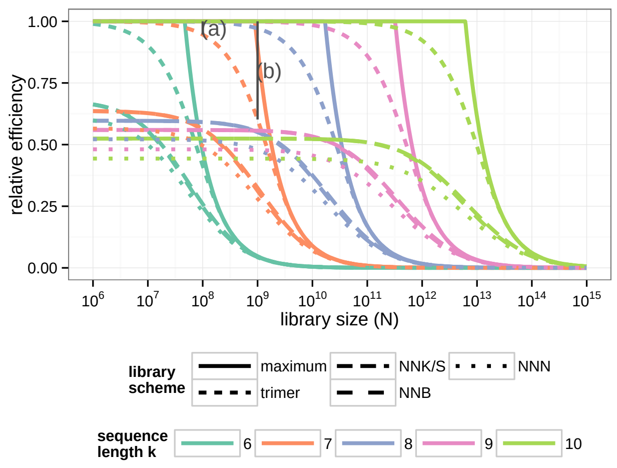


Fig. 2. Overview of relative efficiency for k-peptide libraries (6 to 10) of sizes N from 10^6 to 10^{15}. Relative efficiency decreases with an increased number of oligonucleotides.

encoding scheme and peptide length k. With increasing library size, relative efficiency decreases due to increasing effects of multiplicity. In an ideal case, this drop only occurs when the library size reaches the maximal possible diversity for the given peptide length k. In practice, however, this loss becomes notable when a library reaches a size of about 1% of the maximal number of possible peptides.

Current AAV library sizes are in the order of 10^8. Here, the loss due to multiplicity makes up for less than 10% in heptapeptide trimer libraries (see (a) in Figure 2). As peptide libraries increase, the problem grows exponentially. In heptapeptide libraries of size 10^9, the loss due to multiplicity (see (b) in Figure 2) is nearly as large as the initial loss (27.9% compared to 36.3%).

3 APPLICATIONS

Full coverage – especially with longer peptide sequences – might be very difficult to achieve in practice. However, as Yuval Nov describes for saturation mutagenesis in protein evolution (Nov, 2012), it might not always be reasonable to aim for full coverage to ensure that the one ‘best’ peptide is included in a library (what is ‘best’ is always defined by the goals of a specific library selection, e.g. to identify the peptide that shows the strongest interaction with a protein). The reasoning behind this is simple: one would expect that there are in fact several peptides which perform similarly well. This assumption is supported by the fact that even in selections using libraries with incomplete coverage, we often observe an enrichment of several sequences that share common sequence motifs (e.g. Naumer *et al.*, 2012; Michelfelder *et al.*, 2007, 2009). With this in mind, it might be more reasonable, instead, to raise the question: “What diversity is necessary to find *at least one of* the best possible peptides?” To answer this, we first estimate the probability that the single best sequence is part of the library (see 3.1), in a next step we assess the probability that any related sequence from an appropriately specified neighborhood around it is included (see 3.2).

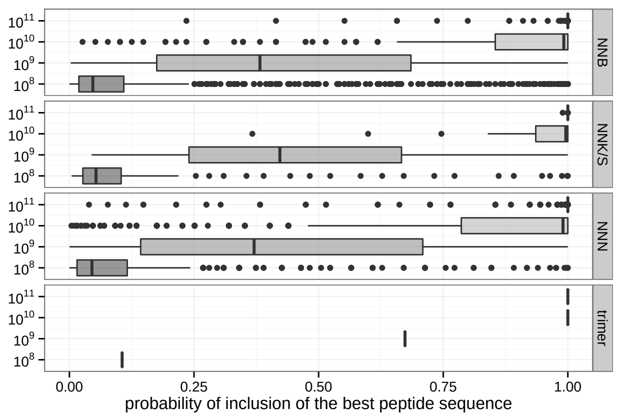


Fig. 3. Overview of the inclusion probabilities for any peptide sequence of length 7 in libraries of sizes $10^8, 10^9$, 10^{10}, and 10^{11} (left) for different encoding schemes (right).

3.1 Inclusion probabilities

The probability that a specific peptide sequence is present in a library depends on the overall size of the library and its scheme. Let p_i be the probability that peptide i is in the library, and $\sum_{i=1}^{t} p_i$ be the cumulative probability for the occurrence of any one of a group of t peptide sequences in the library. Define X to be the number of the specified t peptides that occur in a library of size N. The probability that at least one of the t peptides is in the library is then:

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

The approximation is based on the same argument as Theorem 1 and holds for any reasonably large values of N.

The probability p_i of a peptide sequence to occur in a library depends on the number of codons of each of its amino acids. This number varies between library schemes, making an exact *a priori* assessment of the inclusion probability of the ‘best’ peptide sequence impossible except in the case of trimer libraries, in which each peptide sequence occurs with equal probability. In all other library schemes, the probability of sequences to be included in the library is highly variable. Figure 3 gives an overview of just how much the probability of including the ‘best’ peptide sequence varies in each encoding scheme with different library sizes. Each of the boxes of the boxplots contain half of all possible 19^7 peptide sequences, with the other quarters on the left and right of the boxes, respectively. For example, in an NNK/S library of size 100 Million, 75% of all peptides have an inclusion probability of below 12.5%.

Compared to trimer libraries, the other library schemes introduce an enormous amount of variability into the inclusion probabilities. This is reflected in more variable results from experiments and makes the probability of peptide inclusion depend strongly on the actual peptide sequence, which *a priori* is unknown to the experimenter.

3.2 Neighborhoods

To determine if at least one of the best possible peptides (or a *top* peptide) is included in a given library, we have to define first what a *top* peptide is. For that we use a rather restrictive definition: a *top* peptide is any peptide that differs from the best possible peptide s in up to one (first degree neighborhood) or up to two (second degree neighborhood) amino acid positions which are conservatively exchanged. To objectively define conservative exchanges we employ the BLOSUM80 matrix (Henikoff and Henikoff, 1992), which provides log-odds scores for the chance to observe a substitution of one amino acid for another. Only exchanges with a positive BLOSUM80 score were considered in determining neighborhoods of top peptides. Further, exchanges to stop codons and cysteines were defined here to lead to invalid sequences. In general, a neighborhood of degree d includes all sequences that differ in at most d amino acids from peptide s. It is obvious, that a d-neighborhood of s includes s itself as well as all sequences of lower degree neighborhoods.

Neighborhoods and their sizes depend on the individual peptide sequence. Therefore, we cannot give a single inclusion probability, but we rather have to cite a range of probabilities for including top peptides. To set the boundaries of this range, we consider a best and a worst case scenario under all encoding schemes. In the worst case scenario, the top sequence consists of amino acids with only a single codon each (minimizing the probability to be part of the library) along with the smallest possible number of viable exchanges (minimizing the size of the top peptide neighborhood). Analogously, the top sequence in the best case scenario is one that consists of amino acids with a maximum number of codons in the encoding scheme (maximizing the probability to be found in the library) combined with the largest possible number of viable exchanges (maximizing the size of the top peptide neighborhood).

Figure 4 gives an overview of the minimum and maximum possible probabilities of including one of the sequences in the first and second degree neighborhoods of the best heptapeptide sequence. For an NNK/S library of size 100 Million, we have a minimum chance of about 12% (worst case scenario) that one of the sequences of the first degree neighborhood around the best sequence is included. This chance increases to over 60% to contain at least one sequence for the second degree neighborhood of the sequence.

For individual sequences we can be more specific, and calculate the probability of including any of its d degree neighbors (for $d = 1, 2$) based on the BLOSUM80 matrix, see supplement C.

4 DISCUSSION AND CONCLUSIONS

Peptide libraries are used in a variety of biological systems. For optimum exploitation of this powerful technology, evaluation parameters are needed to allow an in-depth description of the library properties and a more informed library design that can be adjusted to the requirements of the needed peptides and to the technical feasibility in a given experimental setting.

In this publication, we have provided a mathematical framework for an objective assessment of the quality of a wide variety of peptide libraries with different library sizes, encoding schemes (NNN, NNB, NNK, NNS, and trimer) and peptide lengths. In particular, it allows for the first time the mathematical description of peptide libraries of length > 6 aa, and defines peptide diversity, expected coverage, and relative efficiency as evaluation parameters. Further, the framework is suitable to describe properties of even very large current libraries (e.g. AAV libraries of 5×10^8 transformed clones; phage libraries of 2×10^{10} transformed clones; Deshayes *et al.*, 2002; Naumer *et al.*, 2012).

The core of our approach is to classify peptides according to the multiplicity of their encodings first, and then use these peptide classes to regard individual peptide sequences in a second step. This two-step procedure reduces the complexity of the problem sufficiently, making a mathematical assessment of complete libraries analytically feasible. The sheer size of peptide libraries causes alternative approaches to fail. Direct simulation, for instance, is impossible to implement on standard machines due to the limitations of main memory and disk space. Even if these hurdles were taken by more sophisticated simulation strategies, the process would be too slow to be of practical use.

In this publication, we limited our analysis to peptides of 6 to 10 amino acids in length. However, shorter or longer peptides can be investigated as well. Reasonably sized libraries of peptides shorter than 6 aa in length can be expected to show near complete coverage and to contain almost all possible peptides in the investigated encoding schemes. For example, a pentapeptide NNN library of size $N = 10$ Million has an estimated coverage of 98% and a peptide diversity of 3×10^6. When investigating peptide lengths of 11 amino acids or more, it can be assumed even for large current libraries (N up to 2×10^{10}) that the probability that two bacterial clones encode the same peptide is close to zero. Under these conditions, the expected coverage drops towards zero while relative efficiency stays close to the possible maximum defined by peptide length and encoding scheme (Figures 1 and 2). For such libraries, losses in efficiency are strongly dominated by the initial loss and a relative efficiency R, peptide diversity D, and expected coverage C can be approximately described as (defined in dfn 2, eqn 3, and dfn 1, respectively):

$$R(N,k) = \left(\frac{\text{\# of viable codons in the encoding scheme}}{\text{\# of all codons in the encoding scheme}} \right)^k$$

$$D(N,k) = R(N,k) \cdot N$$

$$C(N,k) = D(N,k)/19^k$$

For the experimenter, desirable library features are high peptide diversity as well as good coverage. To improve these factors, the size of the library can be increased. However, this possibility is often limited by technical restraints. Diversity and coverage are, however, also influenced by the used encoding scheme and peptide length. While short peptides (e.g. $k < 6$) reach high coverage with all encoding schemes, their restricted coding capacity limits the number of possible variants and thereby the achievable peptide diversity. If peptides of lengths between 6 and 10 amino acids are considered, the encoding scheme has an important influence. Trimer libraries are superior, as they avoid the initial loss and suffer less from redundancy effects caused by multiple bacterial clones encoding the same peptide. Further, they prevent the bias for preferentially including amino acids with a high number of codons. As trimer libraries are still rather expensive, the majority of currently used libraries are made based on other encoding schemes. When comparing these regarding coverage and relative

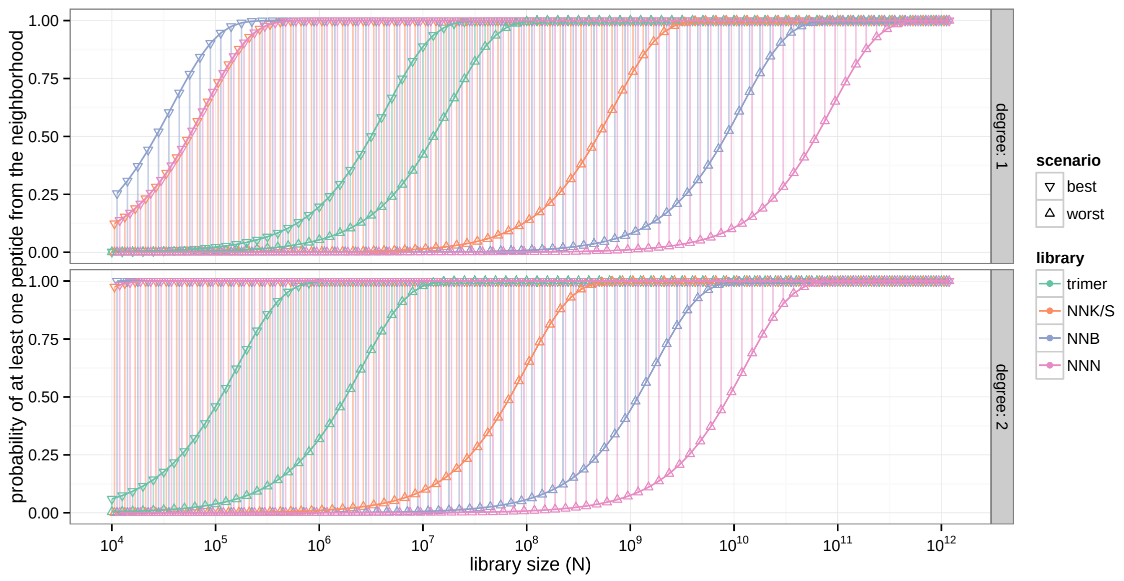


Fig. 4. Overview of the probability to include at least one of the sequences belonging to the first and second degree neighborhoods of the best heptapeptide. Best and worst case probabilities depend on the number of encodings for a sequence and the exchangeability of amino acids. In first degree neighborhoods the best case scenarios for NNK/S and NNN are identical.

efficiency, NNK/S and NNB are very similar and preferable to NNN (see Figures 1 and 2). NNK/S has a slight advantage over NNB in peptide diversity, expected coverage, and relative efficiency. However, if cysteines are considered as viable, NNB encoding has a minor advantage over NNK/S for libraries with a low coverage. At higher coverage, NNK/S encoding again is slightly better than NNB (data not shown). Between NNK and NNS, there is no statistical difference. However, biological considerations like codon preferences of the relevant host organism might cause a difference in a given experimental setting. In *E. coli* and especially in *S. cerevisiae*, codon usage suggests that NNK may generally be the better option (Patrick and Firth, 2005), while in human cells NNS codons are preferred. For all encoding schemes except trimer, there is a peptide length that maximises peptide diversity for a given library size N (see Figure 2). For example, libraries of size 10^8 have a maximum relative efficiency and peptide diversity if k is 8. The peptide length that gives optimum peptide diversity increases with increasing library size. Compared to other library schemes, in trimer libraries longer peptides should in theory always yield higher diversity and efficiency. In practice however, peptides longer than 9 aa are not expected to be notably beneficial in regard to relative efficiency (see Figure 2) at any currently achievable library size. Additionally, increasing peptide length in all encoding schemes will reduce coverage and thereby the chance that top peptides are included in the library.

In conclusion, our framework offers for the first time a set of evaluation parameters that allows for an in-depth analysis of peptide libraries. Thereby, a more informed library design becomes possible, which can be adjusted to the requirements of the needed peptides and to the technical feasibility in a given experimental setting. Further, the chances that a top peptide is included in the library, and thereby indirectly, the probability that the best peptide isolated from a selection is in fact the best possible peptide can now be estimated. This in turn can help to decide if a further optimization of the isolated peptide is potentially worthwhile.

The mathematical framework presented here was implemented in PeLiCa (`http://www.pelica.org`), a web-based tool for scientists to perform *in silico* analyses of various peptide library designs. PeLiCa allows calculations for a wide variety of library sizes, peptide lengths, and encoding schemes.

REFERENCES

Binder, M., Müller, F., Jackst, A., L'echenne, B., Pantic, M., Bacher, U., Zu Eulenburg, C., Veelken, H., Mertelsmann, R., Pasqualini, R., Arap, W., and Trepel, M. (2011). B-cell receptor epitope recognition correlates with the clinical course of chronic lymphocytic leukemia. *Cancer*, **117**(9), 1891–1900.

Bosley, A. D. and Ostermeier, M. (2005). Mathematical expressions useful in the construction, description and evaluation of protein libraries. *Biomol. Eng.*, **22**(1-3), 57–61.

Bupp, K. and Roth, M. J. (2003). Targeting a retroviral vector in the absence of a known cell-targeting ligand. *Hum. Gene Ther.*, **14**(16), 1557–1564.

DeGraaf, M. E., Miceli, R. M., Mott, J. E., and Fischer, H. D. (1993). Biochemical diversity in a phage display library of random decapeptides. *Gene*, **128**(1), 13–17.

Deshayes, K., Schaffer, M. L., Skelton, N. J., Nakamura, G. R., Kadkhodayan, S., and Sidhu, S. S. (2002). Rapid identification of small binding motifs with high-throughput phage display: discovery of peptidic antagonists of IGF-1 function. *Chem. Biol.*, **9**(4), 495–505.

Dörrie, H. (1965). *100 great problems of elementary mathematics : their history and solution*. Dover books on elementary and intermediate mathematics. Dover Publications, New York.

Firth, A. E. and Patrick, W. M. (2008). GLUE-IT and PEDEL-AA: new programmes for analyzing protein diversity in randomized libraries. *Nucleic Acids Research*, **36**.

Fukunaga, K. and Taki, M. (2012). Practical tips for construction of custom Peptide libraries and affinity selection by using commercially available phage display cloning systems. *J Nucleic Acids*, **2012**, 295719.

Girod, A., Ried, M., Wobus, C., Lahm, H., Leike, K., Kleinschmidt, J., Deléage, G., and Hallek, M. (1999). Genetic capsid modifications allow efficient re-targeting of adeno-associated virus type 2. *Nat Med.*, **5**(9), 1052–1056.

Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, **89**(22), 10915–10919.

Kayushin, A. L., Korosteleva, M. D., Miroshnikov, A. I., Kosch, W., Zubov, D., and Piel, N. (1996). A convenient approach to the synthesis of trinucleotide phosphoramidites–synthons for the generation of oligonucleotide/peptide libraries. *Nucleic Acids Res.*, **24**(19), 3748–3755.

Kreppel, F., Gackowski, J., Schmidt, E., and Kochanek, S. (2005). Combined genetic and chemical capsid modifications enable flexible and efficient de- and retargeting of adenovirus vectors. *Mol. Ther.*, **12**(1), 107–117.

Krumpe, L. R., Schumacher, K. M., McMahon, J. B., Makowski, L., and Mori, T. (2007). Trinucleotide cassettes increase diversity of T7 phage-displayed peptide library. *BMC Biotechnol.*, **7**, 65.

Lindner, T., Kolmar, H., Haberkorn, U., and Mier, W. (2011). DNA libraries for the construction of phage libraries: statistical and structural requirements and synthetic methods. *Molecules*, **16**(2), 1625–1641.

Lu, G., Zheng, M., Zhu, Y., Sha, M., Wu, Y., and Han, X. (2012). Selection of peptide inhibitor to matrix metalloproteinase-2 using phage display and its effects on pancreatic cancer cell lines PANC-1 and CFPAC-1. *Int J Biol Sci*, **8**(5), 650–662.

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

McConnell, S. J., Kendall, M. L., Reilly, T. M., and Hoess, R. H. (1994). Constrained peptide libraries as a tool for finding mimotopes. *Gene*, **151**(1-2), 115–118.

Michelfelder, S. and Trepel, M. (2009). Adeno-associated viral vectors and their redirection to cell-type specific receptors. *Adv. Genet.*, **67**, 29–60.

Michelfelder, S., Lee, M. K., deLima Hahn, E., Wilmes, T., Kaul, F., Muller, O., Kleinschmidt, J. A., and Trepel, M. (2007). Vectors selected from adeno-associated viral display peptide libraries for leukemia cell-targeted cytotoxic gene therapy. *Exp. Hematol.*, **35**(12), 1766–1776.

Michelfelder, S., Kohlschütter, J., Skorupa, A., Pfennings, S., Muller, O., Kleinschmidt, J. A., and Trepel, M. (2009). Successful expansion but not complete restriction of tropism of adeno-associated virus by in vivo biopanning of random virus display peptide libraries. *PLoS ONE*, **4**(4), e5122.

Müller, O. J., Kaul, F., Weitzman, M. D., Pasqualini, R., Arap, W., Kleinschmidt, J. A., and Trepel, M. (2003). Random peptide libraries displayed on adeno-associated virus to select for targeted gene therapy vectors. *Nat. Biotechnol.*, **21**(9), 1040–1046.

Naumer, M., Ying, Y., Michelfelder, S., Reuter, A., Trepel, M., Müller, O. J., and Kleinschmidt, J. A. (2012). Development and validation of novel AAV2 random libraries displaying peptides of diverse lengths and at diverse capsid positions. *Hum. Gene Ther.*, **23**(5), 492–507.

Nishimoto, T., Yamamoto, Y., Yoshida, K., Goto, N., Ohnami, S., and Aoki, K. (2012). Development of peritoneal tumor-targeting vector by in vivo screening with a random peptide-displaying adenovirus library. *PLoS ONE*, **7**(9), e45550.

Noren, K. A. and Noren, C. J. (2001). Construction of high-complexity combinatorial phage display peptide libraries. *Methods*, **23**(2), 169–178.

Nov, Y. (2012). When second best is good enough: another probabilistic look at saturation mutagenesis. *Appl Environ Microbiol*, **78**(1), 258–62.

Patrick, W. M. and Firth, A. E. (2005). Strategies and computational tools for improving randomized protein libraries. *Biomol. Eng.*, **22**(4), 105–112.

Patrick, W. M., Firth, A. E., and Blackburn, J. M. (2003). User-friendly algorithms for estimating completeness and diversity in randomized proteinencoding libraries. *Protein Engineering*, **16**(6), 451–457.

Perabo, L., Goldnau, D., White, K., Endell, J., Boucas, J., Humme, S., Work, L. M., Janicki, H., Hallek, M., Baker, A. H., and Buning, H. (2006). Heparan sulfate proteoglycan binding properties of adeno-associated virus retargeting mutants and consequences for their in vivo tropism. *J. Virol.*, **80**(14), 7265–7269.

Rodi, D. J., Janes, R. W., Sanganee, H. J., Holton, R. A., Wallace, B., and Makowski, L. (1999). 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.

Rodi, D. J., Soares, A. S., and Makowski, L. (2002). Quantitative assessment of peptide sequence diversity in M13 combinatorial peptide phage display libraries. *J. Mol. Biol.*, **322**(5), 1039–1052.

Scholle, M., Kehoe, J., and Kay, B. (2005). Efficient construction of a large collection of phage-displayed combinatorial peptide libraries. *Comb Chem High Throughput Screen*, **8**(6), 545–51.

Scott, J. K. and Smith, G. P. (1990). Searching for peptide ligands with an epitope library. *Science*, **249**(4967), 386–390.

Varadi, K., Michelfelder, S., Korff, T., Hecker, M., Trepel, M., Katus, H. A., Kleinschmidt, J. A., and Müller, O. J. (2012). Novel random peptide libraries displayed on AAV serotype 9 for selection of endothelial cell-directed gene transfer vectors. *Gene Ther.*, **19**(8), 800–809.

Waterkamp, D. A., Müller, O. J., Ying, Y., Trepel, M., and Kleinschmidt, J. A. (2006). Isolation of targeted AAV2 vectors from novel virus display libraries. *J Gene Med*, **8**(11), 1307–1319.

Xie, Q., Bu, W., Bhatia, S., Hare, J., Somasundaram, T., Azzi, A., and Chapman, M. S. (2002). The atomic structure of adeno-associated virus (AAV-2), a vector for human gene therapy. *Proc. Natl. Acad. Sci. U.S.A.*, **99**(16), 10405–10410.