# A Framework and Application for Efficient Analysis of Peptide Libraries

Eric Riemer Hare

April 10, 2015

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.

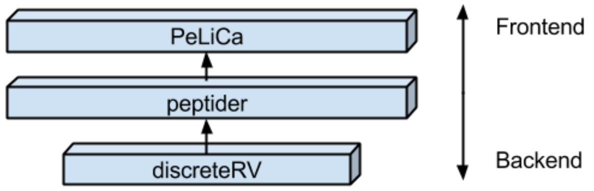


Figure 1: The three components, or layers, of my project.

# Manipulation of Discrete Random Variables with discreteRV

*by Eric Hare, Andreas Buja, Heike Hofmann*

**Abstract** A prominent issue in statistics education is the sometimes large disparity between the theoretical and the computational coursework. **discreteRV** is an R package for manipulation of discrete random variables which uses clean and familiar syntax similar to the mathematical notation in introductory probability courses. The package offers functions that are simple enough for users with little experienced with statistical programming, but has more advanced features which are suitable for a large number of more complex applications. In this paper, we 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 to bridge the gap between theoretical notation from textbooks and lectures, and the statements used in statistical software required in more and more classes. Depending on the background of the student, this missing link can manifest itself differently: some students master theoretical concepts and notation, but struggle with the computing environment, while others are very comfortable with statistical programming, but find it difficult to translate their knowledge back to the theoretical setting of the classroom.

**discreteRV** is an approach to help with bringing software commands closer to the theoretical notation. The package provides a comprehensive set of functions to create, manipulate, and simulate from discrete random variables. It is designed for introductory probability courses. **discreteRV** uses syntax that closely matches the notation of standard 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 discrete random variables. A list of these functions and brief discriptions of their functionality is available in Table 1.

### 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 as a theoretical construct representing the value of an outcome of a random experiment (see e.g. Wild and Seber, 1999). A discrete random variable is a special case that can only take on a countable set of values. Discrete random variables are associated with probability mass functions, which map the set of possible values of the random variable to probabilities. Probability mass functions must therefore define probabilities which are between zero and one, and must sum to one.

Throughout this document, we will work with two random variables, a simple example of a discrete random variable representing the value of a roll of a fair die, and one representing a realization of a Poisson random variable with mean parameter equal to two. Formally, we can define such random variables and their probability mass functions as follows:

Let $X$ be a random variable representing a single roll of a fair die; i.e., the sample space $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $X$ is the identity, mapping the upturned face of a die roll to the corresponding number of dots visible. Then,

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

Let $Y$ be a random variable distributed according to a Poisson distribution with mean parameter $\lambda$. In this case, Y takes on values in the non-negative integers $\{0, 1, 2, ...\}$. Then,

| Name | Description |
|---|---|
| Creation | |
| `RV` | Create a random variable consisting of possible outcome values and their probabilities or odds |
| `as.RV` | Turn a probability vector with possible outcome values in the names() attribute into a random variable |
| `jointRV` | Create a joint random variable consisting of possible outcome values and their probabilities or odds |
| Manipulation | |
| `iid` | Returns a random variable with joint probability mass function of random variable $X^n$ |
| `independent` | Returns a boolean indicating whether two RVs X and Y are independent |
| `joint` | Returns a random variable with joint probability mass function of random variables X and Y |
| `marginal` | The specified marginal distribution of a joint random variable |
| `margins` | All marginal distributions of a joint random variable |
| `SofI` | Sum of independent random variables |
| `SofIID` | Sum of independent identically distributed random variables |
| Probabilities | |
| `P` | Calculate probabilities of events |
| `probs` | Probability mass function of random variable X |
| `E` | Expected value of a random variable |
| `V` | Variance of a random variable |
| `SD` | Standard deviation of a random variable |
| `SKEW` | Skewness of a random variable |
| `KURT` | Kurtosis 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.

$$f(y) = P(Y = y) = \begin{cases} \frac{\lambda^y e^{-\lambda}}{y!} & y \in 0, 1, 2, \dots \\ 0 & \text{otherwise} \end{cases}$$

In **discreteRV**, a discrete random variable is defined through the use of the RV function. RV accepts a vector of outcomes, a vector of probabilities, and returns an RV object. The code to create X, a random variable representing the roll of a fair die, is as follows:

```
(X <- RV(outcomes = 1:6, probs = 1/6))

#> Random variable with 6 outcomes
#>
#> Outcomes   1   2   3   4   5   6
#> Probs    1/6 1/6 1/6 1/6 1/6 1/6
```

Defaults are chosen to simplify the random variable creation process. For instance, if the probs argument is left unspecified, **discreteRV** assumes a uniform distribution. Hence, the following code is equivalent for defining a fair die:

```
(X <- RV(1:6))

#> Random variable with 6 outcomes
#>
#> Outcomes   1   2   3   4   5   6
#> Probs    1/6 1/6 1/6 1/6 1/6 1/6
```

Outcomes can be specified as a range of values, which is useful for distributions in which the outcomes that can occur with non-zero probability are unbounded. This can be indicated with the range argument, which defaults to TRUE in the event that the range of values includes positive or negative infinity. To define our poisson random variable Y, we specify the outcomes as a range and the probabilities as a function:

```
pois.func <- function(y, lambda) { return(lambda^y * exp(-lambda) / factorial(y)) }

(Y <- RV(outcomes = c(0, Inf), probs = pois.func, lambda = 2))

#> Random variable with outcomes from 0 to Inf
#>
#> Outcomes     0     1     2     3     4     5     6     7     8     9    10    11
#> Probs    0.135 0.271 0.271 0.180 0.090 0.036 0.012 0.003 0.001 0.000 0.000 0.000
#>
#> Displaying first 12 outcomes
```

Several common distributions are natively supported so that the function need not be defined manually. For instance, an equivalent method of defining $Y$ is:

```
(Y <- RV("poisson", lambda = 2))

#> Random variable with outcomes from 0 to Inf
#>
#> Outcomes     0     1     2     3     4     5     6     7     8     9    10    11
#> Probs    0.135 0.271 0.271 0.180 0.090 0.036 0.012 0.003 0.001 0.000 0.000 0.000
#>
#> Displaying first 12 outcomes
```

The RV function also allows the definition of a random variable in terms of odds. We construct a loaded die in which a roll of one is four times as likely as any other roll as:

```
(X.loaded <- RV(outcomes = 1:6, odds = c(4, 1, 1, 1, 1, 1)))

#> Random variable with 6 outcomes
#>
#> Outcomes   1   2   3   4   5   6
#> Odds     4:5 1:8 1:8 1:8 1:8 1:8
```

| 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.

### Structure of an RV object

The syntactic structure of the included functions lends itself both to a natural presentation of elementary probabilities and properties of probability mass functions in an introductory probability course, as well as more advanced modeling of discrete random variables. In Table 2, we provide an overview of the notational similarities between **discreteRV** and the commonly used probability textbook by Casella and Berger (2001)

A random variable object is constructed by defining a standard R vector to be the possible outcomes that the random variable can take (the sample space $\Omega$). It is preferred, though not required, that these be encoded as numeric values, since this allows the computation of expected values, variances, and other distributional properties. This vector of outcomes then stores attributes which include the probability of each outcome. By default, the print method for a random variable will display the probabilities as fractions in most cases, aiding in readability. The probabilities can be retrieved as a numeric vector by using the `probs` function:

```
probs(X)

#>         1         2         3         4         5         6
#> 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667
```

### Probability-based calculations

By storing the outcomes as the principle component of the object X, we can make a number of probability statements in R. For instance, we can calculate the probability of obtaining a roll greater than 1 by using the code $P(X > 1)$. R will check which of the 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 we compute the probability of this occurrence in the function P 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. Consider our Poisson random variable $Y$, and suppose we want to obtain the probability that $Y$ is within a distance $\delta$ of its mean parameter $\lambda = 2$:

```
delta <- 3
lambda <- 2

P((Y >= lambda - delta) %AND% (Y <= lambda + delta))

#> [1] 0.9834364
```

Alternatively, we could have also used the slightly more complicated looking expression:

```
P((Y - lambda)^2 <= delta^2)

#> [1] 0.9834364
```

Conditional probabilities often provide a massive hurdle for students of introductory probability classes. These type of questions often make it necessary to first translate the problem from everyday language into the mathematical concept of conditional probability, e.g., what is the probability that you will not need an umbrella when the weather forecast said it was not going to rain? Similarly, what is the probability that a die shows a one, if we already know that the roll is no more than 3?

The mathematical solution is, of course, $P(X = 1 \mid X \leq 3)$. In **discreteRV** this translates directly to a solution of `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. Using the pipe for conditional probablity, we had to create alternative `%OR%` and `%AND%` operators, as specified previously.

We can compute several other distributional quantities, including the expected value and the variance of a random variable. 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)`. The expected value for a poisson random variable is its mean, and hence `E(Y)` in our example will return the value two. The function `V(X)` computes the variance of random variable X. Alternatively, we can also work from first principles and assure ourselves that the expression `E((X-E(X))ˆ2)` provides the same result:

```
V(X)

#> [1] 2.916667

E( (X-E(X))^2 )

#> [1] 2.916667
```

## 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 `iid` 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 the first eight outcomes for $n = 3$. Notice again that the probabilities have been coerced into fractions for readability. Notice also that the outcomes of the joint distribution are encoded by the outcomes on each trial separated by a comma.

| 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.

The `*` operator has been overloaded in order to allow a more seamless syntax for defining joint distributions. Suppose we wish to compute the joint distribution of `X`, our toss of a fair coin, and a coin flip. After defining the coin flip variable, the joint distribution can be defined as follows:

```
Z <- RV(0:1)
X * Z

#> Random variable with 12 outcomes
#>
#> Outcomes  1,0  1,1  2,0  2,1  3,0  3,1  4,0  4,1  5,0  5,1  6,0  6,1
#> Probs    1/12 1/12 1/12 1/12 1/12 1/12 1/12 1/12 1/12 1/12 1/12 1/12
```

Note that we the behavior is slightly different when using the `*` operator on the same random variable. That is, `X * X` will not compute a joint distribution of two realizations of $X$, but will rather return the random variable with the original outcomes squared, and the same probabilities. This allows us to perform computations such as `E(Xˆ2)` without encountering unexpected behavior.

Joint distributions need not be the product of iid random variables. Joint distributions in which the marginal distributions are dependent can also be defined. Consider the probability distribution defined in Table 5. Note that A and B are dependent, as the product of the marginal distributions does not equal the joint. We can define such a random variable in **discreteRV** by using the `jointRV` function, which is a wrapper for `RV`:

| | 1 | 2 | 3 |
|---|---|---|---|
| 0 | 1/45 | 4/45 | 7/45 |
| 1 | 2/45 | 1/9 | 8/45 |
| 2 | 1/15 | 2/15 | 1/5 |

**Table 5:** Outcomes and their associated probabilities for a joint distribution of random variables A (along the columns) and B (along the rows).

```
(AandB <- jointRV(outcomes = list(1:3, 0:2), probs = 1:9 / sum(1:9)))
```

```
#> Random variable with 9 outcomes
#>
#> Outcomes  1,0  1,1  1,2  2,0  2,1  2,2  3,0  3,1  3,2
#> Probs    1/45 2/45 1/15 4/45  1/9 2/15 7/45 8/45  1/5
```

The individual marginal distributions can be obtained by use of the `marginal` function:

```
A <- marginal(AandB, 1)
B <- marginal(AandB, 2)
```

Although the marginal distributions allow all the same computations of any univariate random variable, they maintain a special property. The joint distribution that produced the marginals are stored as attributes in the object. This allows for several more advanced probability calculations, involving the marginal and conditional distributions:

```
P(A < B)
```

```
#> [1] 0.06666667
```

```
P(A == 2 | B > 0)
```

```
#> [1] 0.3333333
```

```
P(A == 2 | (B == 1) %OR% (B == 2))
```

```
#> [1] 0.3333333
```

```
independent(A, B)
```

```
#> [1] FALSE
```

```
A | (A > 1)
```

```
#> Random variable with 2 outcomes
#>
#> Outcomes    2    3
#> Probs    5/13 8/13
```

```
A | (B == 2)
```

```
#> Random variable with 3 outcomes
#>
#> Outcomes   1   2   3
#> Probs    1/6 1/3 1/2
```

```
E(A | (B == 2))
```

```
#> [1] 2.333333
```

**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)` creates a random variable object for the sum of two fair dice as shown in Table 6.

The `SofI` function computes the random variable representing the sum of two independent, but not necessarily identically distributed, random variables. The + operator is overloaded to make this computation even more syntactically friendly. Note, however, that similar limitations apply as in the joint distribution case:

| 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 6:** Outcomes and their associated Probabilities for a variable representing the sum of two independent rolls of a die.

```
X + Z

#> Random variable with 7 outcomes
#>
#> Outcomes    1    2    3    4    5    6    7
#> Probs    1/12  1/6  1/6  1/6  1/6  1/6 1/12

X + X # Note that this is NOT a random variable for X1 + X2

#> Random variable with 6 outcomes
#>
#> Outcomes   2   4   6   8  10  12
#> Probs    1/6 1/6 1/6 1/6 1/6 1/6

2 * X # Same as above

#> Random variable with 6 outcomes
#>
#> Outcomes   2   4   6   8  10  12
#> Probs    1/6 1/6 1/6 1/6 1/6 1/6
```

## Plotting

**discreteRV** includes a plot method for random variable objects so that visualizing outcomes and their probabilities is as simple as calling plot(X). Figure 1 shows a visual representation of the probability mass function (pmf) of a fair die. The x axis includes all outcomes, and the y axis includes the probabilities of each particular outcome. Figure 2 shows the pmf of the sum of two independent rolls of a fair die. The pmf of a sum of 20 independent rolls of a die is given in Figure 3.

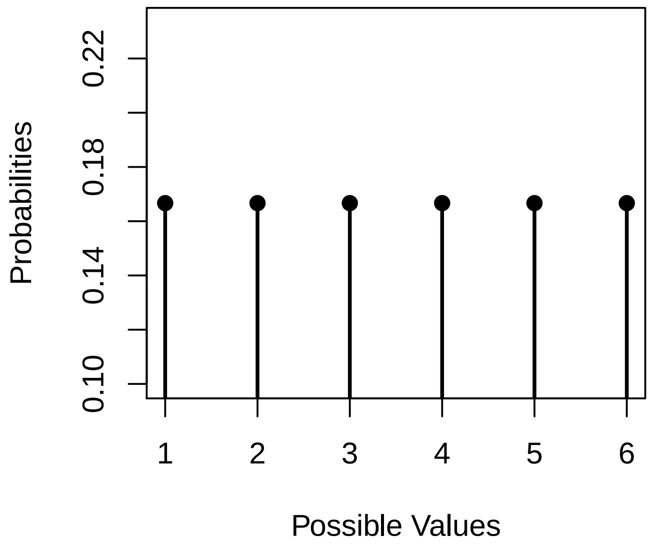


**Figure 1:** Plot method called on a fair die random variable.

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. While very close to a normal, the sum of 20 independent rolls of a fair die still shows a slight S curve in the Q-Q plot.

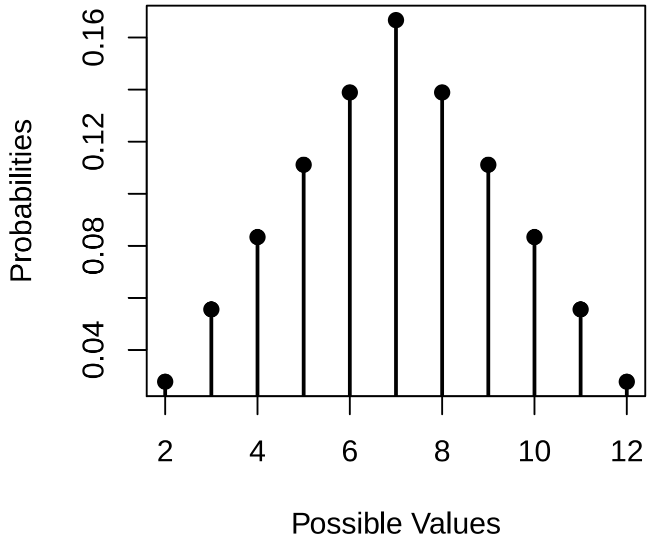


**Figure 2:** Plot method called on a sum of two fair die random variables.

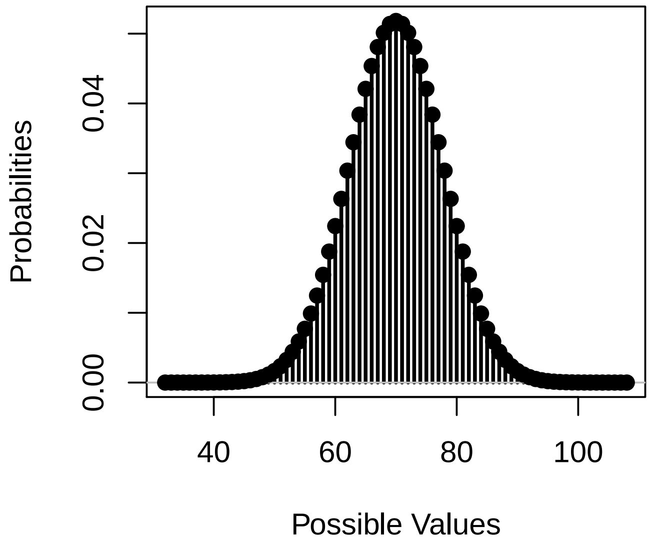


**Figure 3:** Plot method called on a sum of 20 fair die random variables.

## 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 7.

| 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 7:** List of the simulation functions contained in **discreteRV**.

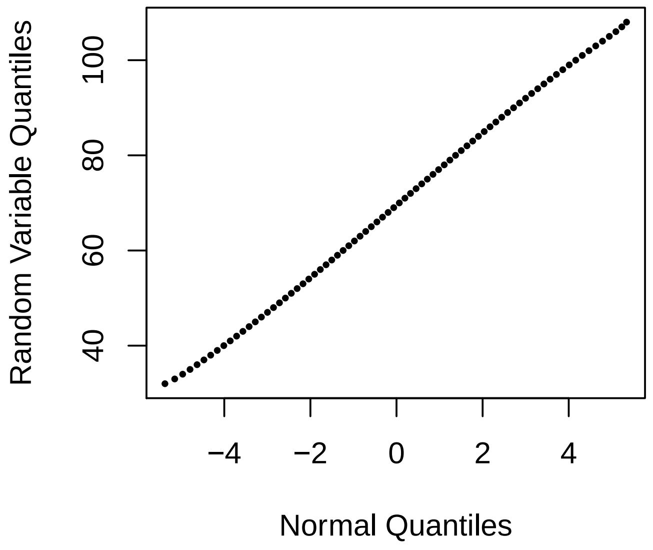


**Figure 4:** qqnorm method called on a sum of 20 fair die random variables.

### Creation

Creating a simulated random vector is done by using the `rsim` function. `rsim` accepts a parameter X representing the random variable with which to simulate from, and a parameter n representing the number of independent trials to simulate. 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(X, 10))
```

```
#> Simulated Vector:  3 1 2 5 3 5 5 3 3 4
#>
#> Random variable with 6 outcomes
#>
#> Outcomes   1   2   3   4   5   6
#> Probs    1/6 1/6 1/6 1/6 1/6 1/6
```

The object returned is a vector of simulated values, with a class attribute containing 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] 3 1 2 5 3 5 5 3 3 4
```

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
#> 0.1 0.1 0.4 0.1 0.3 0.0
```

```
Prop(X.sim == 3)
```

```
#> [1] 0.4
```

```
Prop(X.sim > 3)
```

```
#> [1] 0.4
```

## Extended example: playing craps

Craps is a common dice game played in casinos. The game begins with what is called the “Come Out” roll, in which two fair dice are rolled. If a sum of seven or eleven is obtained, the player wins. If a sum of two, three, or twelve is obtained, the player loses. In all other cases, the roll obtained is declared the “Point" and the player rolls again in an attempt to obtain this same point value. If the player rolls the Point, they win, but if they roll a seven, they lose. Rolls continue until one of these two outcomes is achieved.

**discreteRV** allows for a seamless analysis and simulation of the probabilities associated with different events in Craps. Let us begin by asking “What is the probability that the game ends after the first roll?” To answer this question we construct two random variables. We note that calling `RV(1:6)` returns a random variable for a single roll of a fair die, and then we use the overloaded `+` operator to sum over two rolls to obtain the random variable `Roll`.

```
(Roll <- RV(1:6) + RV(1:6))

#> Random variable with 11 outcomes
#>
#> Outcomes    2    3    4    5    6    7    8    9   10   11   12
#> Probs    1/36 1/18 1/12  1/9 5/36  1/6 5/36  1/9 1/12 1/18 1/36
```

Recall that the game ends after the first roll if and only if a seven or eleven is obtained (resulting in a win), or a two, three, or twelve is obtained (resulting in a loss). Hence, we calculate the probability that the game ends after the first roll as follows:

```
P(Roll %in% c(7, 11, 2, 3, 12))

#> [1] 0.3333333
```

Now suppose we would like to condition on the game having ended after the first roll. Using the conditional probability operator in **discreteRV**, we can obtain the probabilities of winning and losing given that the game ended after the first roll:

```
P(Roll %in% c(7, 11) | Roll %in% c(7, 11, 2, 3, 12))

#> [1] 0.6666667

P(Roll %in% c(2, 3, 12) | Roll %in% c(7, 11, 2, 3, 12))

#> [1] 0.3333333
```

Now, let’s turn our attention to calculating the probability of winning a game in two rolls. Recall that we can use the `iid` function to generate joint distributions of independent and identically distributed random variables. In this case, we would like to generate the joint distribution for two independent rolls of two dice. Now, we will have $11^2$ possible outcomes, and our job is to determine which outcomes result in a win. We know that any time the first roll is a seven or eleven, we will have won. We also know that if the roll is between four and ten inclusive, then we will get to roll again. To win within two rolls given that we’ve received a four through ten requires that the second roll match the first. We can enumerate the various possibilities to calculate the probability of winning in two rolls, which is approximately 30%.

```
TwoRolls <- iid(Roll, 2)

First <- marginal(TwoRolls, 1)
Second <- marginal(TwoRolls, 2)

P(First %in% c(7, 11) %OR% (First %in% 4:10 %AND% (First == Second)))

#> [1] 0.2993827
```

Finally, suppose we are interested in the empirical probability of winning a game of Craps. Using the simulation functions in **discreteRV**, we can write a routine to simulate playing Craps. Using the `rsim` function, we simulate a single game of Craps by rolling from our random variable `roll`, which represents the sum of two dice. We simulate 100 games, and then perform this simulation 1000 times. The results indicate that the player wins a game of craps approximately 49% of the time.

```
craps_game <- function(RV) {

    my.roll <- rsim(RV, 1)

    if (my.roll %in% c(7, 11)) { return(1) }
    else if (my.roll %in% c(2, 3, 12)) { return(0) }
    else {
        new.roll <- 0
        while (new.roll != my.roll & new.roll != 7) {
            new.roll <- rsim(RV, 1)
        }

        return(as.numeric(new.roll == my.roll))
    }
}

sim.results <- replicate(100000, craps_game(Roll))
mean(sim.results)
```

```
#> [1] 0.49321
```

## Conclusion

The power of **discreteRV** is truly in its simplicity. Because it uses familiar introductory probability syntax, it allows 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*, 2015. R package version 1.2. [p]

A. Buja and H. Wickham. *rv2: Discrete random variables*, 2014. R package version 0.1. [p]

G. Casella and R. L. Berger. *Statistical Inference*, volume 2. Cengage Learning, 2001. [p4]

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]

*Eric Hare*
*Iowa State University*
*1121 Snedecor Hall*
*Ames, IA, 50011*
erichare@iastate.edu

*Andreas Buja*
*University of Pennsylvania*
*400 Jon M. Huntsman Hall*
*Philadelphia, PA, 19104*
buja.at.wharton@gmail.com

*Heike Hofmann*
*Iowa State University*
*2413 Snedecor Hall*
*Ames, IA, 50011*
hofmann@iastate.edu

# Analyzing Peptide Libraries with peptider

*by Eric Hare, Heike Hofmann, and Timo Sieber*

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

## Introduction

Libraries of peptides, or amino acid sequences, have a number of applications in the Biological sciences, from studying protein interactions, to vaccine research (Rodi et al., 1999; Irving et al., 2001). Despite their importance, little analysis has been done to assess the statistical properties of different peptide libraries.

**peptider** is a newly-released R package which helps to evaluate many important statistical properties of these libraries. It supports a number of built-in library schemes, including NNN, NNB, NNK, NNS, and 20/20 schemes. It also allows for easy analysis of user-created custom library schemes. **peptider** makes use of the R package **discreteRV** (Buja, 2013), which allows for manipulation and analysis of discrete random variables; all of the graphical output is presented within the **ggplot2** framework (Wickham, 2009). By treating each amino acid in a peptide as a realization of an independent draw from the pool of all possible amino acids, probabilities for the occurrence of peptides can easily be formulated. The corresponding definitions for the statistical functions in **peptider** are taken from Sieber et al. (2014).

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

### Library Schemes

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

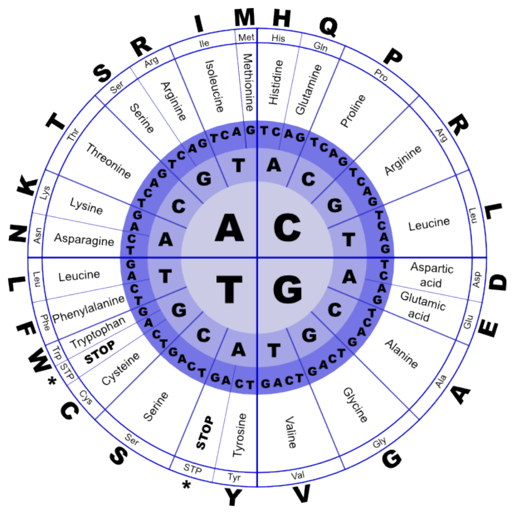


**Figure 1:** Codon wheel illustrating which bases lead to codon representations for each amino acid.

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

```
scheme("NNN")

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

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

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

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

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

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

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

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

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

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

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

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

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

## Library Diversity

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

### Functional Diversity

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

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

To compute this diversity measure in **peptider**, the `diversity` function is used:

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

## [1] 808963.1
```

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

```
makowski(6, "NNK")

## [1] 0.2917751
```

### Expected Coverage

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

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

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

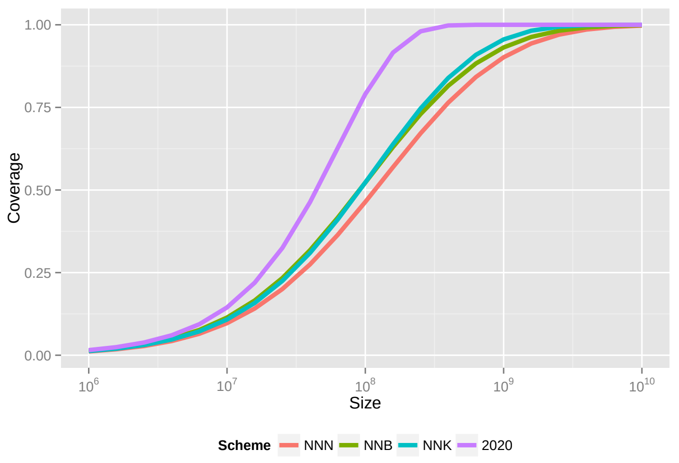


**Figure 2:** Overview of coverage rates across different hexapeptide library schemes of varying size.

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

## [1] 0.5229421
```

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

```r
library(plyr)
library(ggplot2)

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

```r
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.

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

## [1] 0.3346829
```

## Peptide Coverage

The measures of library diversity included in **peptider** introduced to this point are useful to broadly assess the library performance. However, applications of peptide libraries often require knowledge of the probability of observing particular peptides in the library. **peptider** helps to analyze the the probability of observing both the peptide of interest to the user, as well as peptides that are most similar.

### Peptide Inclusion

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

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

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

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

## [1] 0.5166138
```

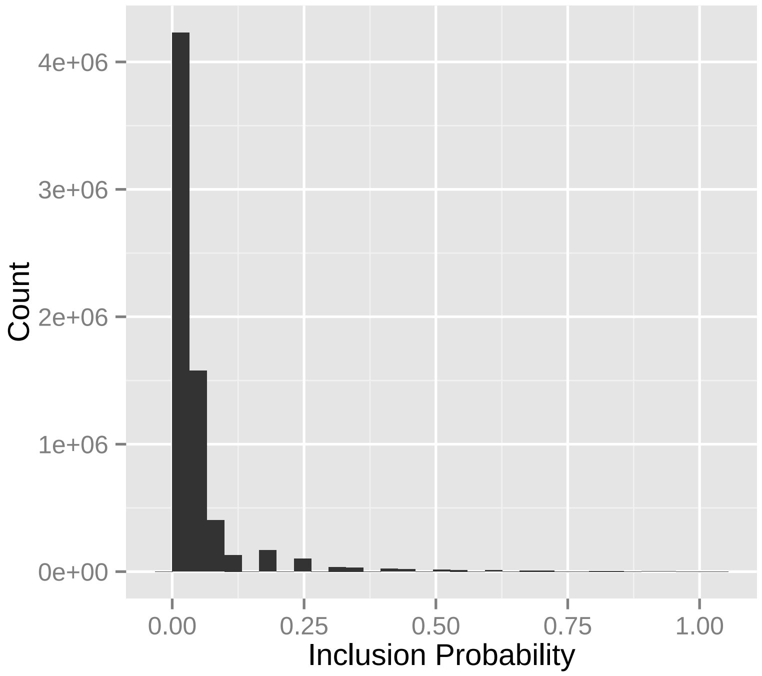
We may also be interested in the range of inclusion probabilities for all peptides rather than just a particular sequence. In this case, the `detect` function is useful. detect differs a bit syntactically as it requires the created library to be passed in as an argument. For an NNK library with peptide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

```
my_lib <- libscheme("NNK", 6)

summary(detect(my_lib, 10^7))

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0112  0.1190  0.3756  0.4405  0.7285  0.9997
```

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



### Neighborhoods

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

```
getNeighbors("HENNING")

##  [1] "HENNING" "QENNING" "YENNING" "HDNNING" "HQNNING" "HKNNING" "HEDNING"
##  [8] "HENDING" "HENNLNG" "HENNMNG" "HENNVNG" "HENNIDG"
```

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

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

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

## Further Work

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

## Conclusion

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

## Bibliography

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

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

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

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

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

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

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

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

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

*Published online XX January 2015* *Nucleic Acids Research, 2015, Vol. XX, No. XX* ***1–4***
*doi:10.1093/nar/gkn000*

# A Web Application for Efficient Analysis of Peptide Libraries

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

[1]Department of Statistics, Iowa State University, Ames IA 50011-1210, United States.
[2]New affiliation.

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

## 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.**

## INTRODUCTION

With the introduction of the R package *peptider* (1, 2), 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. This intro needs besides the technological intro an intro from the biological side - maybe in the form of a set of functions that can be answered throughout the paper

## USER INTERFACE

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.

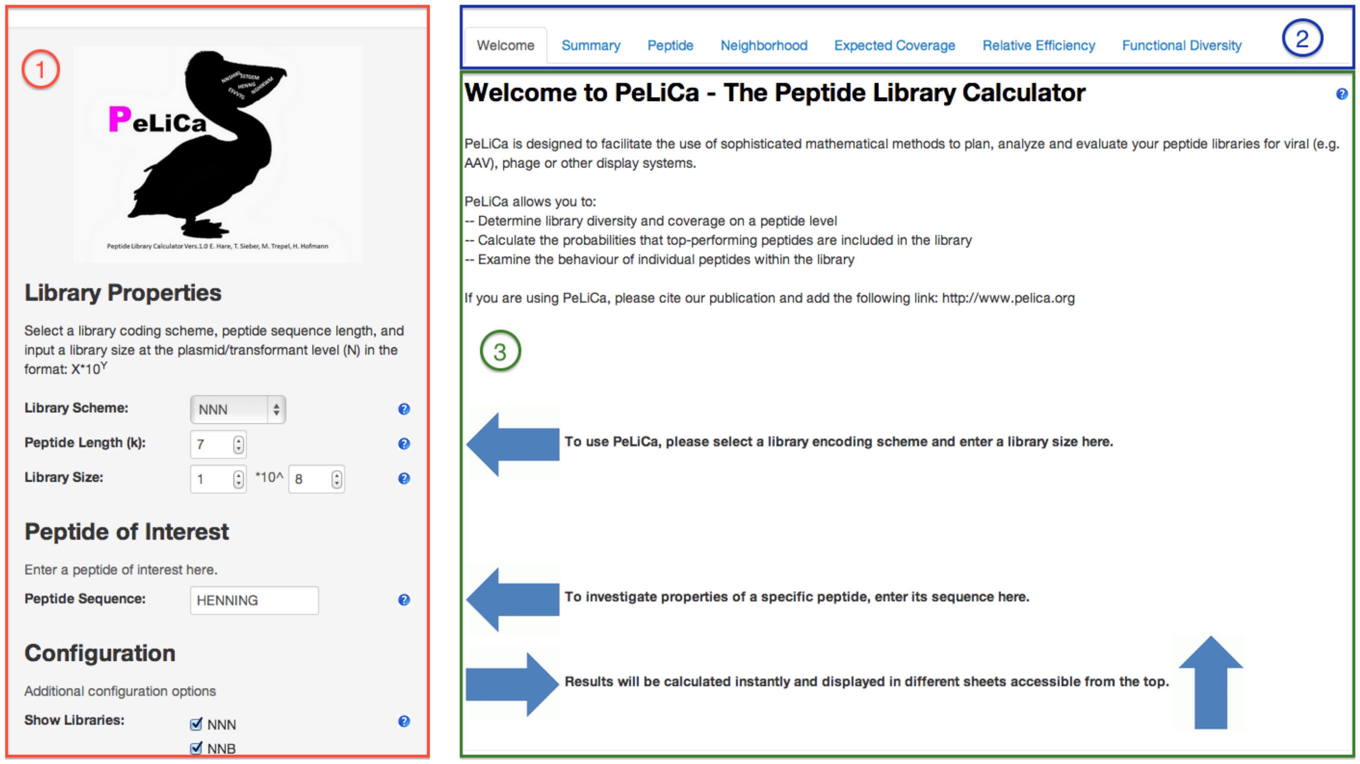
Another important component of the user interface are the help tooltips. Throughout the application, blue question marks will be available. When the user moves their mouse cursor over these icons, tooltips will appear to instruct the user on how to proceed, or provide more information about the library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

## FEATURES

### 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.

*To whom correspondence should be addressed. Tel: , Fax: , Email: erichare@iastate.edu



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

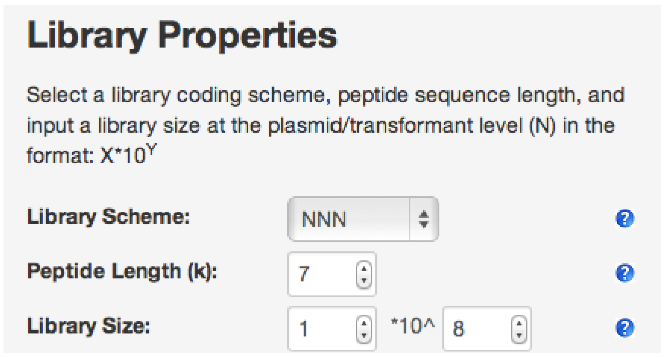
If you are using PeLiCa, please cite our publication and add the following link: http://www.pelica.org

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.



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

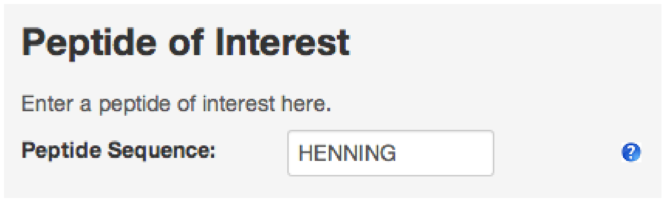
Library Size: 1 *10^ 8

**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 $1.0 \times 10^6$ and $9.9 \times 10^{14}$. 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.

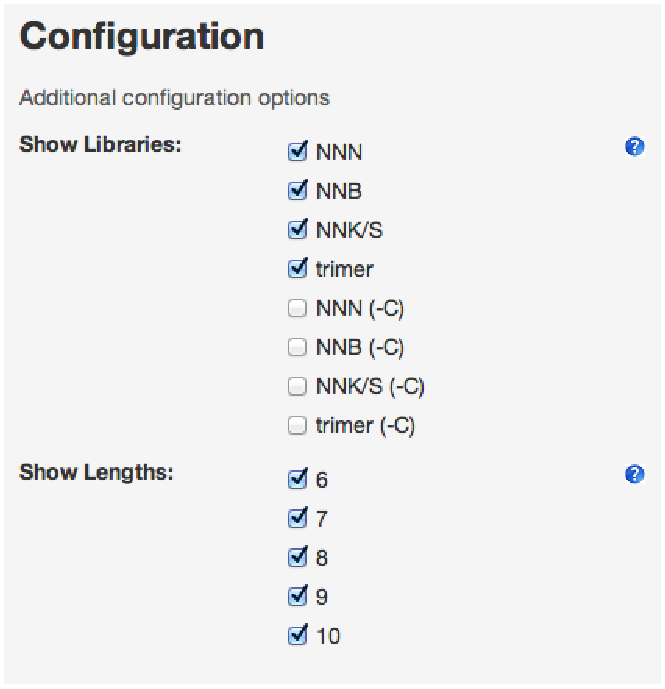


Peptide of Interest

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.

## 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.

*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.

*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.

*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.

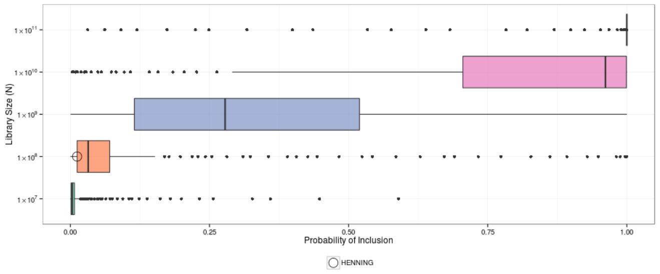


**Figure 5.** Boxplots displayed in PeLiCa representing the inclusion probabilities for an NNN library with peptide length seven and a library size of $1 \times 10^8$. The inclusion probability of HENNING is displayed in the plot as a circle.

*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.

*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.

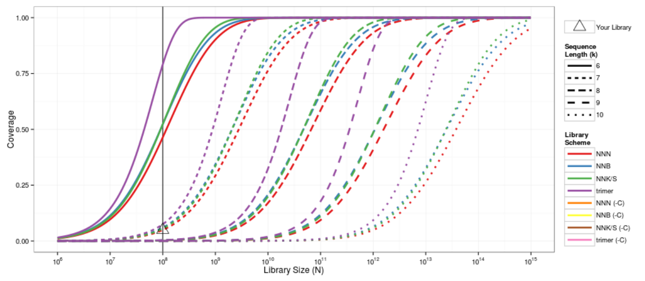


**Figure 6.** Plots of expected coverage for an NNN library with peptide length seven and a library size of $1 \times 10^8$

*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.

*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).

## TECHNOLOGY

PeLiCa is a Shiny application (3). 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. In this way, the comuptation of peptide library properties can be done on the fly, with a simple point-and-click interface and without any use of programming.

## 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.

## 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.

## FUNDING

This work was not funded by a specific source.

## REFERENCES

1. R Core Team R: A Language and Environment for Statistical Computing R Foundation for Statistical Computing Vienna, Austria (2013).
2. Hofmann,H., Hare,E., and ggobi Foundation peptider: Evaluation of diversity in nucleotide libraries (2013) R package version 0.1.2.
3. RStudio Inc. shiny: Web Application Framework for R (2014) R package version 0.10.0.

## List of Figures

## SUPPLEMENTARY MATERIAL

# Biomathematical Description of Synthetic Peptide Libraries

Timo Sieber[1], Eric Hare[2], Heike Hofmann[2,*], Martin Trepel[3]

**1** Department of Oncology and Hematology/University Medical Center Hamburg-Eppendorf, Hamburg, Germany

**2** Department of Statistics, Iowa State University, Ames, IA, USA

**3** Department of Hematology and Oncology/Augsburg Medical Center/Interdisciplinary Cancer Center, Augsburg, Germany

* Corresponding author

E-mail: hofmann@mail.iastate.edu (HH)

## Abstract

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.We are presenting a mathematical framework that allows us to derive several properties of libraries, such as the expected number of different peptides, the library's coverage and its relative efficiency. This helps to describe libraries in enough detail for researchers to plan new experiments in a more informed manner. In particular, these values allow us to answer -in a probabilistic fashion- the question of whether a specific library does indeed contain one of the 'best' possible peptides. The framework is implemented in a web-interface based on two packages, discreteRV and peptider, to the statistical software environment R.

## Introduction

Since the year 2000 we see on average more than 500 publications a year that are based on the use of peptide libraries[1]. This serves as a good measure to reflect on the importance of peptide libraries in a wide spectrum of biological applications ranging from the identification of protein interaction sites [e.g. 1] and the development of enzyme inhibitors [e.g. 2] to identification of peptides that mediate cell type specific gene delivery by viral vector systems [e.g. 3]. In all of these applications, chemically synthesized random oligonucleotides are introduced into plasmids encoding structural proteins of bacteriophages [4]

[1]PubMed query April 2014 on "peptide library"

or viruses, such as adeno-associated viruses [3], adenoviruses [5] or retroviruses [6]. Plasmids are then ligated and transformed into bacteria to generate a plasmid library, which in turn 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; 2]. The success of this method is highly dependent on the diversity of the initial pool of peptides, as the chance to identify the "best possible" sequence, or even a suitable sequence, is directly correlated with the number and diversity of the peptides in the library used for the screening procedure. A cheap, simple and powerful way to investigate, if the production of a library was successful is the Quick-Quality-Control (QQC) [7,8]. In short, library material is pooled and used in a single sanger sequencing run to uncover undesired imbalances in the ratios of inserted bases as well as production errors like primer-dimer insertions etc., which might lead to a reduced library diversity.

Determining the diversity of a library is problematic, though, as the number of distinct peptides, which we will refer to as *peptide diversity*, cannot be measured easily. Direct measurements are generally impracticable: even though next-generation sequencing is now widely accessible, the sheer size of current libraries [e.g. $2 \times 10^{10}$ clones; 9] makes the use of this technique for counting purposes prohibitive due to the time and financial effort associated with the very high sequencing depth required for a sufficient sequencing coverage. Other approaches of measuring library diversity in the literature include DeGraaf et al. [10], who estimate diversity of their phage decapeptide display library from the distribution of single amino acids and dipeptides in a sample. Rodi et al. define *functional diversity* as a measure of the distribution of peptides encoded in the library [11,12]. Both methods, functional diversity and peptide diversity, give valuable distributional information about peptide libraries. A library with an even distribution of sequence frequencies is advantageous, as all peptides enter the selection process in comparable numbers. This supports a swift and successful selection of a suitable peptide. However, peptides that match the selection criteria can be gradually enriched during the selection process, even if they are vastly underrepresented in the initial library. A limitation of functional diversity is that it is a theoretical measure based purely on the library scheme.Functional diversity therefore does not represent the actual number of distinct peptides in a library, which increases with growing size independently of its scheme.

Therefore, many researchers estimate diversity at the level of the plasmid library by counting successfully transformed bacterial colonies [e.g. 13–15]. 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 all steps of the library production to make the number of bacterial colonies a valid qualifier for the peptide library [16]. The number of bacterial colonies on its own is of limited value, as the relevant metric is the number of distinct peptides in the library. However, the two measures are correlated and the number of bacterial colonies can be used to estimate peptide diversity. Peptide diversity of the library is always lower than colony number, due to the possibility that different bacterial clones encode identical peptides. This is caused by several clones containing identical peptide encoding DNA and/or by clones harboring distinct DNA sequences that encode the same peptide due to the degenerate nature of the genetic code: amino acids are encoded by up to six distinct codons; multiple DNA sequences can therefore describe the same peptide. This has the effect that, for instance, a pool of randomised codon DNA sequences of length seven has a nominal diversity of $64^7$ (64 codons; $4.4 \times 10^{12}$) while it encodes only $23^7$ (20 amino acids and three stop codons; $3.4 \times 10^9$) distinct amino acid sequences. Further, stop codons in the random nucleotide sequence prematurely terminate the peptide and can cause dysfunctional proteins in display systems [17, 18]. Libraries are therefore often encoded by limited subsets of the standard 64 codons to at least partially counteract both effects [as also discussed in 19]. 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) are 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. Apart from the mentioned, a number of further encoding schemes exist. These are primarily developed in the framework of saturation mutagenesis, another area in which randomisation libraries are used. Special attention in saturation mutagenesis received the MAX randomization [20], the 22c trick [21] and the "small-intelligent libraries" [22]. However, as these techniques are not suited to produce long stretches [i.e. five or more amino acid positions, 23] of randomized sequences, they are not used for the production of peptide libraries.

One approach to overcome the problematic stemming from the degenerate nature of the genetic code is common to both peptide libraries and saturation mutagenesis and consists of libraries in which the ratio of the number of codons for each amino acid is one. From here on, we will refer to these libraries as 20/20 libraries (20 codons for 20 amino acids). 20/20 libraries also allow a complete avoidance of stop codons, which have been shown to increase functional diversity in phage display [24].

The most common method to produce such peptide libraries is the trimer approach. In trimer libraries [25] oligonucleotides are synthesised by assembling pre-fabricated trinucleotide phosphoramidites or trimers. An alternative to the trimer approach to generate 20/20 libraries is the ProxiMAX system [23].

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 [26]. However, random integration of odd numbers of cysteines has repeatedly been shown to inhibit the generation of peptide bearing phages [27]. 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 adeno-associated virus (AAV) vectors [28–31]. 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 [32]. 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 [33].

With regard to the aforementioned factors, we will determine peptide diversity by using the number of bacterial clones, but consider effects of encoding schemes and stop codons. For the purpose of discussing diversity, we will regard cysteine-containing peptides as non-functional unless otherwise mentioned. A complete discussion of diversity of libraries treating cysteines as valid or invalid can be found at our website PeLiCa [available at `http://www.pelica.org`]. 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 [29, 34]. 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.

Determining the peptide diversity is a mathematically taxing problem that becomes ever more challenging with increasing peptide length. In particular, Monte Carlo simulation is not practical for this purpose. There are two primary limitations:

1. For library sizes above about $10^8$, the speed of the simulation even on modern hardware is prohibitive without the use of massively parallel hardware.

2. Small probabilities (such as we deal with for rare peptides in a library) cannot be accurately

estimated by Monte Carlo methods without oversampling. Oversampling does further increase the complexity of the simulation by increasing the number of runs that need to be made (see 1).

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 *expected coverage* and *relative efficiency* to allow a more 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?"

Our framework allows to determine the peptide diversity of large peptide libraries by combining quantitative information about the number of clones with qualitative information about biological, statistical and encoding effects. This in turn facilitates a deeper understanding and allows for a more informed planning of new, optimized libraries. To make the framework easily accessible, we generated a user-friendly web-interface called PeLiCa, which allows the user to determine all of these factors for libraries of sizes up to $9.9 \times 10^{25}$ bacterial clones, using different encoding schemes (including custom-designed schemes and those that consider cysteine viable) and peptide lengths.

## Methods

### 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 GLUE-IT software [available at `http://guinevere.otago.ac.nz/stats.html`; 35] generates values for diversity and coverage for protein libraries with up to six modified codons per protein. GLUE-IT was designed for another purpose and does not allow evaluation of cysteines as disruptive, but it can also be used to gain some information for peptide libraries with short peptides. However, it is no longer sufficient 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. 27, 29, 36].

In our approach to develop a mathematical framework we consider only peptide libraries that are based on synthetic randomized oligonucleiotides. This asserts, from a statistical point of view, that all DNA sequences inserted into the library plasmids are completely randomised and can be observed multiple times.

We discuss three measures of library quality: *peptide diversity* defined – as stated before – as the number of distinct peptides in a library, *expected coverage*, describing the expected 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. The terms *diversity* and *completeness* used by Firth and Patrick [35] for saturation mutagenesis experiments are equivalent to the concepts peptide diversity and expected coverage we use here for peptide libraries.

We investigate these measures for a set of different encoding schemes: NNN-C, NNK/S-C, NNB-C, and 20/20-C. The -C indicates that we will exclude cysteines from consideration. Note that the 20/20-C notation refers to libraries that are composed of only the 19 valid amino acid codons and do not include the codon for cystein or any of the stop codons. We will first discuss library properties for libraries with equal codon representation, such as we see in 20/20 libraries, and then extend the situation to other library schemes.

## Libraries with equal codon representations

An easily tractable case for determining diversity is the setting in which all 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 20/20 libraries in which every amino acid is represented by one codon. 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 (note that this is also true, when each amino acid is represented by the same number of codons). Denote the number of all different possible peptides in the library by $b$, the size, measured as the number of bacterial colonies, of the library by $N$.

Let us denote the *diversity of this library*, as measured by the number of different peptides, as $Z = Z_{N,b}$. The number of different peptides, $Z$, that can actually be achieved in the library is the primary point of interest. In practice, the value of $Z$ will differ from library to library, but we can determine an expected value of library diversity, $E[Z]$, and its corresponding variance $Var[Z]$ as outlined below [see also 37].

**Theorem 1.** *For a library of size $N$ chosen from a scheme with $b$ different peptides, which are assumed* *to be all equally likely, the expected value and the variance of the number of different peptides $Z_{N,b}$ in the library is given as:*

$$E[Z_{N,b}] = b\left(1 - (1 - b^{-1})^N\right) \approx b(1 - e^{-N/b}). \tag{1}$$

$$\begin{aligned} Var[Z_{N,b}] &= b\left[(1 - b^{-1})^N - (1 - 2b^{-1})^N\right] - b^2\left[(1 - b^{-1})^{2N} - (1 - 2b^{-1})^N\right] \\ &\approx b(e^{-N/b} - e^{-2N/b}) - Ne^{-2(N-1)/b}. \end{aligned} \tag{2}$$

*The approximation becomes more accurate as values of $b$ and $N$ increase. For values of $b$ and $N$ above 50 the approximation is already correct to within 1% of the exact value. The relative standard deviation, or the square root of the variance divided by the mean, is negligibly small for most libraries. The proof and a more detailed discussion of the approximation error can be found in the supplementary material.*

In investigating DNA diversity in site saturation mutagenesis libraries, other groups [38, 39] obtained the same result for expected diversity as Theorem 1 based on a Poisson approximation. While this approach is usable for an analysis at the DNA level or 20/20 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 depends on the 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 $6^7 = 279,936$ 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 only 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. 35, 40] 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.

## 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. Other schemes work similarly, a class partitioning of NNN-C (S1 Table) and NNB-C (S2 Table) is given in the supplementary material. According to the degree of codon redundancy and functionality NNK and NNS are equivalent, and we can distinguish four classes of aa based on a modified NNK/S scheme, in which cysteine is excluded from the set of valid amino acids (Table 1). Amino acids are given in single letter code. Size $s$ defines the number of different amino

**Table 1. NNK/S-C 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 |

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 aa classes is therefore 3. Stop codons as well as cysteines are treated as non-viable 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: In a first step we are only interested in whether the outcome is a *valid sequence*, defined to be the case when there is no element of aa class Z in the sequence. Valid sequences are therefore those that are expected to be functional in the biological system. 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-C 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*, $\ell$, 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. Let $\mathcal{L}$ represent the total number of valid aa classes in the given encoding scheme. Then $\mathcal{L}^k$ is the total number of peptide classes in a library with peptides of length $k$. If this is performed for an exemplary library of dipeptide sequences, we have a set of nine different peptide classes as shown in Table 2. The peptide class (first line) is defined by the aa class memberships of their codons as defined for NNK/S-C 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.

Within each of the $\mathcal{L}^k = 9$ peptide classes, all peptides have an equal number of oligonucleotide sequence representations. This compares to the $19^2 = 361$ possibilities that must be taken into account without the use of peptide classes in the dipeptide case.

**Table 2. All NNK/S-C peptide sequences of length two partitioned according to peptide classes.**

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

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 $s_A, s_B$, and $s_C$ represent the number of different amino acids 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. Here, $n_A, n_B$, and $n_C$ refer to the number of elements from each of the aa classes $A$, $B$, and $C$ that make up the peptide sequence. The sum of $n_A, n_B$, and $n_C$ then adds up to the total length of the sequence.

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

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

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-C 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

$$p = \#\text{ peptides} \cdot \#\text{ oligonucleotides}/v^k. \tag{3}$$

## 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 are represented by the same number of oligonucleotide sequences. Assuming $b_i$ different peptides in peptide class $i$ are theoretically possible, we have, according to theorem 1, an expected diversity given by the number of different peptides as $b_i(1 - e^{-Np_i/b_i})$, resulting in an overall expected number of different peptides in the library and associated variance of

$$D(N,k) = \sum_{i=1}^{\mathcal{L}^k} b_i(1 - e^{-Np_i/b_i}). \tag{4}$$

$$\sigma_D^2 = N\ell(1-\ell) + \sum_{i=1}^{\mathcal{L}^k} \sigma^2(Z_{Np_i,b_i}), \tag{5}$$

where $\ell$ is the initial loss of the library scheme for peptides of length $k$. A simulation-based discussion of this result and the precision of its approximation can be found in section "Simulation Results" of the supplementary material.

## Results

### Expected coverage and relative efficiency

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 and associated variance is defined as*

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

$$\sigma_C^2 = \sigma_D^2/19^{2k}.$$

*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 $N = (\#\ \text{viable amino acids})^k$.*

Fig. 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 the expected coverage for a given library size $N$. Additionally, the used encoding scheme has a profound effect on expected coverage, with 20/20-C libraries being far superior to the other schemes [see also 16,21,41,42]. The line corresponding to 'maximum' represents an ideal situation, in which no initial loss or redundancy occurs, such that at a size of $N < b$, there are $N$ distinct peptides represented, for a coverage of $N/b$. Once the library size exceeds the total possible number of peptides $b$, coverage stays at 1. Increasing library size always improves coverage until 100% coverage is reached. However, the added value gained from increasing library size decreases with increasing total size.

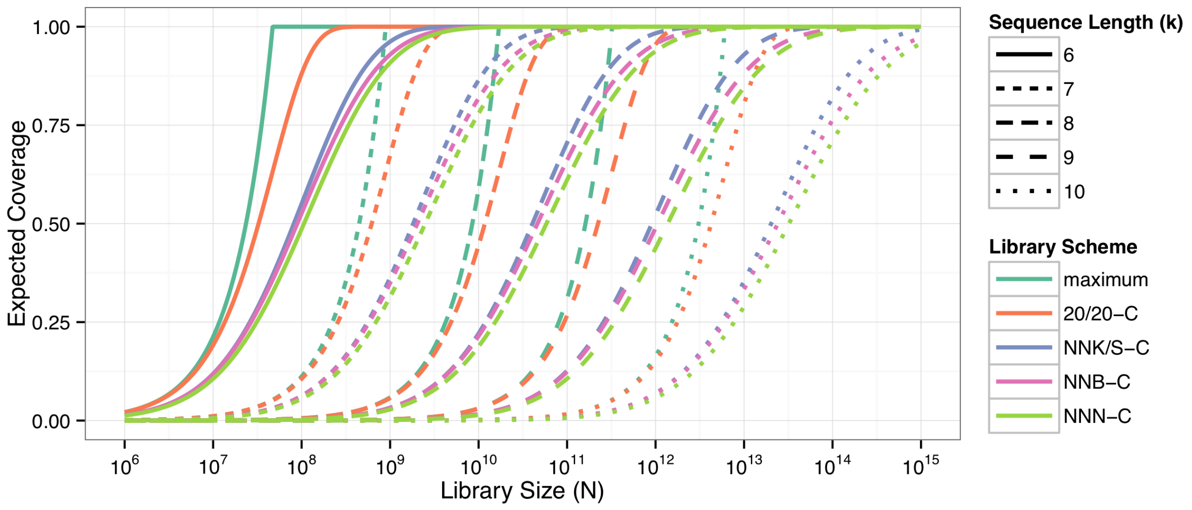


**Fig. 1. Overview of expected coverage for $k$-peptide libraries of different sizes $N$ with the different encoding schemes (NNN-C, NNB-C, NNK/S-C, and 20/20-C).** The 'maximum' line represents the best-case scenario of coverage of a library, in which no peptide appears twice (until the upper limit of all possible peptides is reached at $b = 19^k$, at which point duplication is unavoidable).

We therefore introduce relative efficiency of a library to measure the value returned for a library of a particular size and a specified scheme:

**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:*

$$R(N,k) = D(N,k)/N.$$
$$\sigma_R^2 = \sigma_D^2/N^2.$$

*This makes relative efficiency a number between 0 and 1. A relative efficiency of 1 indicates that all*

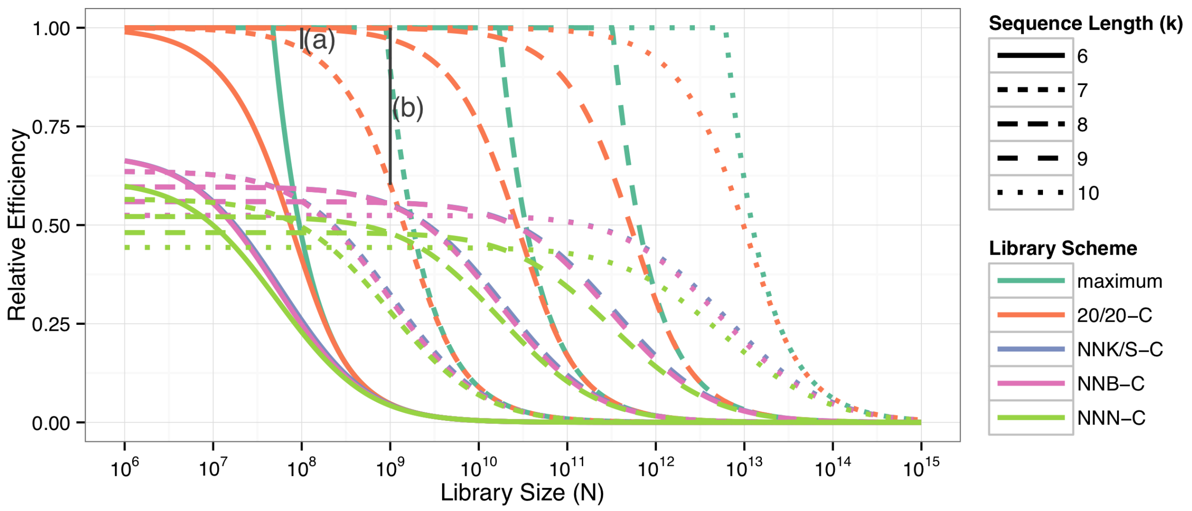


**Fig. 2. Overview of relative efficiency for $k$-peptide libraries (6 to 10) of sizes $N$ from $10^6$ to $10^{15}$.** Relative efficiency decreases with an increased number of oligonucleotides in the library and longer peptide sequencesdue to the larger initial loss.

*peptide sequences in the library are unique and no sequence is found more than once. If the relative efficiency is close to 0 the level of redundant peptide sequences is high. A relative efficiency of 0.5 means that we expect half of all peptide sequences in a library to be valid and unique.*

Fig. 2 gives an overview of relative efficiency of $k$-peptide libraries of various sizes. In contrast to an ideal situation or in a 20/20-C library, libraries encoded by NNK/S-C, NNB-C and NNN-C schemes suffer from an initial loss due to sequences containing aa class Z codons. This limits their maximal relative efficiency depending on encoding scheme and peptide length $k$. With increasing library size, relative efficiency decreases due to increasing effects of redundancy. 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 redundancy makes up for less than 10% in heptapeptide 20/20 libraries (see (a) in Fig. 2). As peptide libraries increase, the problem grows exponentially. In heptapeptide libraries of size $10^9$, the loss due to redundancy (see (b) in Fig. 2) is 39.9%.

## Inclusion Probabilities

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 [42], it might not always be reasonable to aim for full coverage to ensure that the one 'best' sequence 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 highly similar 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. 14, 29, 43]. 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. In a next step we assess the probability that any related sequence from an appropriately specified sequence neighborhood around it is included.

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 20/20 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 [see also 20]. Fig. 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. Side-by side boxplots show the inclusion probabilities of all peptide sequences for each peptide length $k$ from 6 to 10 and library sizes $N$ between $10^8$ and $10^{12}$. The colored boxes contain the middle 50% of all possible peptide sequences. 20/20-C libraries (shown in pink) do not have any variability associated with the inclusion probability, indicating that all peptide sequences have an equal chance to be part of the library. NNN-C libraries have the largest variability associated with them, while NNK/S-C libraries have the smallest (after 20/20-C libraries).

The high variability introduced by schemes with varying codons per amino acid ratios causes libraries to be biased towards peptides with a high number of possible encodings at the cost of rare ones. This makes the chance of success in selections strongly dependent on the question, if the *a priori* unknown "best" peptide has many possible encodings or not. Therefore, the inclusion probability for some peptides is maximal in biased schemes like NNN-C and exceeds that achievable with 20/20-C encoding (see "Examples for inclusion probability" and the S5 table in the supplement). However, for about 75% of all possible peptides the highest inclusion probability is reached when an unbiased coding scheme like 20/20-C is used (see Fig. 3).

## 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 [44], 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 neighborhoods of a lower degree than $s$.

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

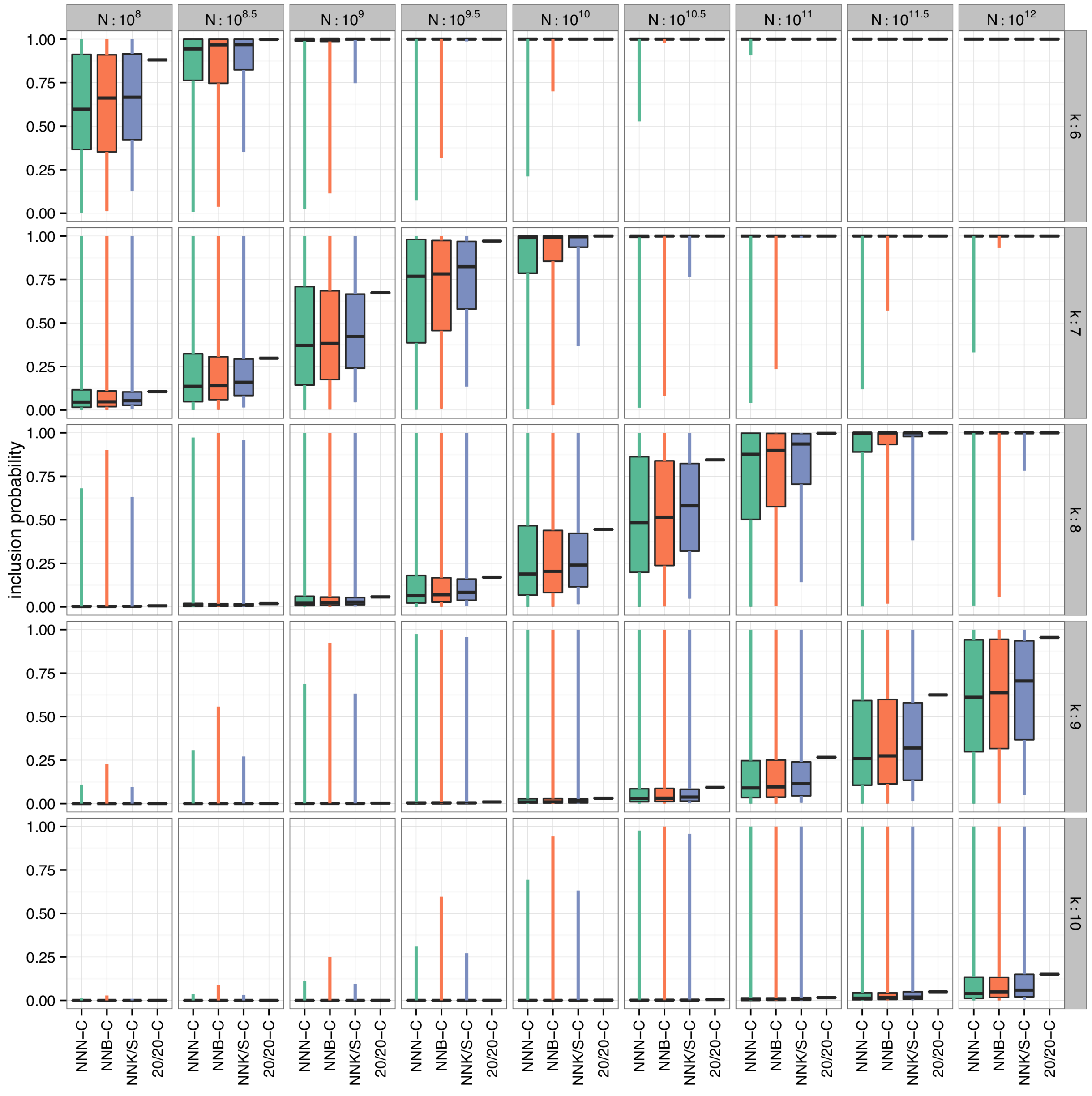


**Fig. 3. Overview of the inclusion probabilities for peptide sequences of lengths 6 to 10 (in rows) in libraries of sizes between $10^8$ to $10^{12}$ (in columns) for different encoding schemes (as side-by-side boxplots).** The boxes contain the middle 50 percent of inclusion probabilities for all peptide sequences of length $k$ in each of the schemes. The vertical lines extend to minimum and maximum of the inclusion probabilities. 20/20-C libraries do not have any variability in the inclusion probabilities, because all sequences are equally likely. NNN-C libraries generally show the largest variability (as seen in the extent of the boxes) in probabilities, followed by NNB-C and NNK/S-C. Simultaneously, median inclusion probabilities increase from NNN-C to 20/20-C libraries for all combinations of peptide lengths and library sizes.

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).

Fig. 4 gives an overview of the probabilities of including one of the sequences in the first degree neighborhood of the best peptide sequence of length $k = 7$. For an NNK/S-C library of size one billion ($N = 10^9$), we have a minimum chance of about 30% (worst case scenario) that one of the sequences of the first degree neighborhood around the best heptapeptide sequence is included. This chance increases to close to 100% for more than 75% of all peptide sequences. Taking a one degree neighborhood of peptide sequences into account has roughly the same effect on inclusion probabilities as considering sequences of a shorter length $(k - 1)$ or using a library of more than ten times the size. Note that a switch from best sequence to first degree neighborhoods of the best sequence does not change the effect that library schemes have on inclusion probabilities except for libraries, which show a higher variability in inclusion probabilities.

For individual sequences we can calculate the probability of including any of its $d$ degree neighbors (for $d = 1, 2$) based on the BLOSUM80 matrix, for an example see S5 table.

In particular for longer peptide sequences, higher degree neighbors might play a significant role in the analysis of results. While theoretically feasible, practically neighborhoods of higher order can only be derived -due to computational limitations- for a limited set of peptide sequences rather than the whole library.

## Discussion

Peptide library selection is a powerful technology used in a wide variety of biological systems. For optimum exploitation of this technique, it is necessary to understand the properties of the peptide libraries. Currently however, the possibilities to functionally describe a peptide library are rather limited. Several publications exist that focus on mathematical descriptions of saturation mutagenesis libraries used in protein evolution [16, 37, 45, 46, among others]. While saturation mutagenesis and peptide library display are similar in

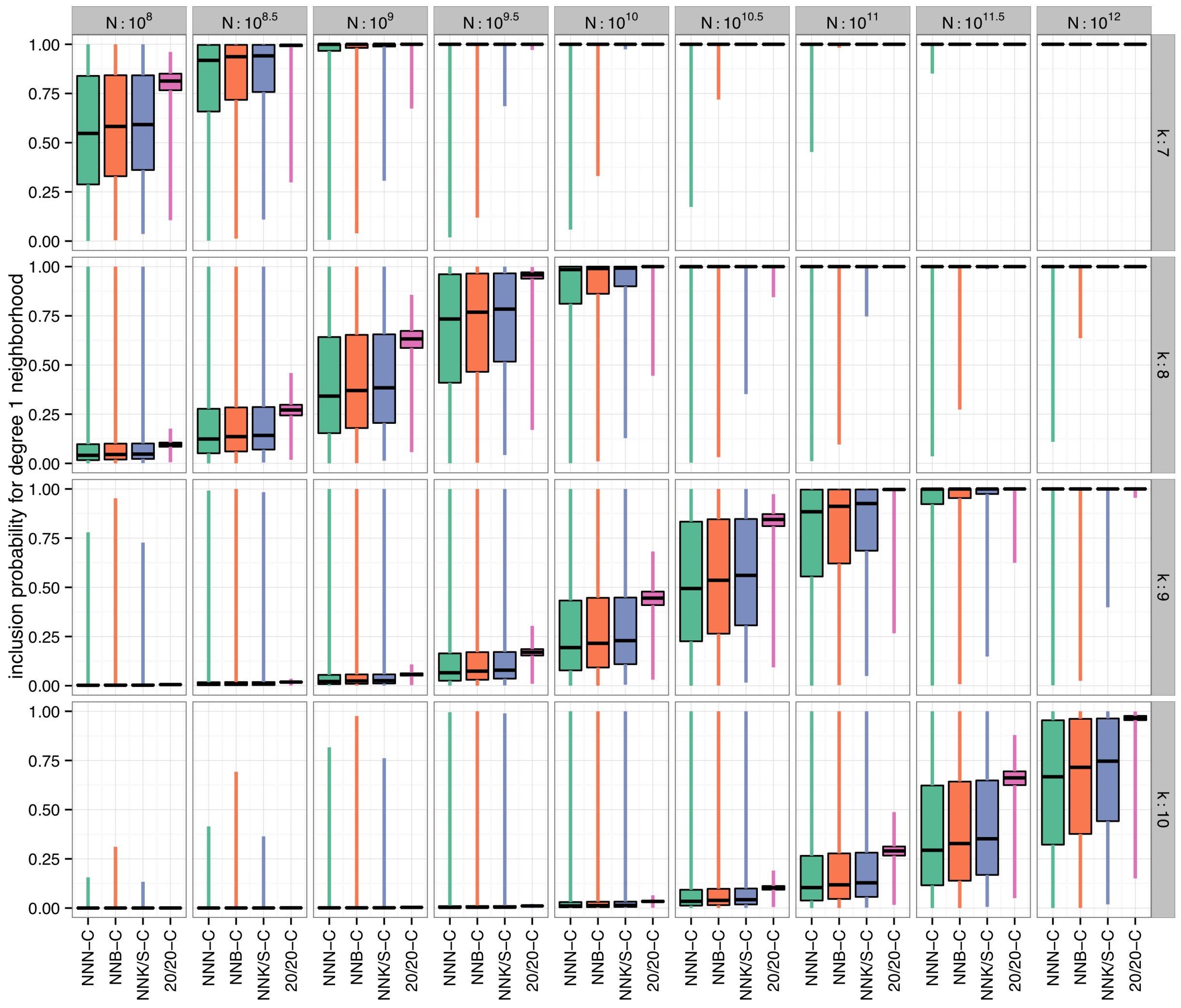


**Fig. 4. Side-by-side boxplots of the probabilities that at least one of the sequences belonging to the first degree neighborhood of the best sequence is included in libraries of different sizes (columns) and different lengths of peptides (rows).** Best and worst case probabilities depend on the number of encodings for a sequence and the exchangeability of the amino acids it consists of.

many aspects, they differ in the fact that in the first generally only low numbers of isolated positions are randomized while in the second often long randomized peptides are used. This causes differences in the techniques available for randomization and, especially, in the number of possible sequences and thereby in the mathematical complexity. Therefore, researchers designing new peptide libraries have to choose key parameters like peptide length, encoding scheme, and target diversity without a possibility to adequately quantify the effects of their decisions. Available qualifiers like functional diversity and number of bacterial colonies offer some degree of information, but are unsuited to compare the properties of different libraries in detail. We present a mathematical framework to determine the number of distinct peptides and to calculate the estimated coverage and relative efficiency. These properties are implemented in the web-based tool PeLiCa (`http://www.pelica.org`) and enable researchers to quantify and compare their libraries in far greater detail, which in particular allows for a more informed planning of new libraries and projects. Researchers can use the preset library schemes in PeLiCa as well as define new ones. The core of our approach is to classify peptides according to the redundancy 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 most 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. For very small library sizes a simulation study is included in the supplementary material, which shows the accuracy of the theoretical framework in practice. For somewhat larger library sizes, the validity of our approach was successfully confirmed by direct comparison with GLUE-IT [35]. GLUE-IT determines protein diversity and coverage for small libraries of individual proteins with mutations in up to 6, in general non-consecutive, amino acid positions ("saturation mutagenesis generated protein libraries"). Though the biological setting is different from the peptide libraries discussed here, GLUE-IT can be used to analyse a limited set of peptide libraries with very short randomised inserts ($k = 1$ to 6; Cysteines defined as valid; comparison in supplementary material). In reverse, our approach and website can also be used to investigate saturation mutagenesis libraries.

In this publication, we limit our examples to peptides of 6 to 10 amino acids in length, as shorter peptides are rarely used and the use of longer peptides - even for very large current libraries ($N$ up to

$2 \times 10^{10}$) - results in an expected coverage close to zero. The relative efficiency in these cases stays close to its possible maximum defined by peptide length and encoding scheme (Figs. 1 and 2). The losses in efficiency are strongly dominated by the initial loss and a relative efficiency $R$ (defined in definition 2) captures the ratio of the number of viable codons and all codons in the scheme.

The most fundamental information about a peptide library is the number of encoded peptides. However, determining this value is difficult, as it is not only influenced by the number of clones generated in library production, but also by other factors. Our framework is able to determine a value for the peptide diversity from the number of bacterial colonies by figuring in statistical and encoding effects as well as prominent biological factors (stop codons and cysteines). The negative influence of these factors has already been discussed in the past [e.g. 19–23, 27, 41, 47–49], however, our system now allows a quantification of their effects. Based on peptide diversity we calculate the relative efficiency and the expected library coverage. The standard deviations for all three measures are negligibly small for reasonably sized libraries. For example an NNK-C heptapeptide library of $N = 10^8$ has a peptide diversity of $5.6 \times 10^7 \pm 9.6 \times 10^3$, an expected coverage of $6.3\% \pm 1.1 \times 10^{-5}\%$ and a relative efficiency of $56.3\% \pm 9.6 \times 10^{-5}\%$ (For more detail see supplementary material).

Information on coverage is important to put libraries and selection results in perspective [see also 35]. In the above example (NNK-C library; $k = 7$; $N = 10^8$) only about 6% of all possible sequences are covered. Therefore, it is not likely that the most prominent sequence selected from this library is in fact the best possible heptapeptide. Besides that, two identical selections using independent libraries might result in identification of two completely different sets of selected peptides. This situation can be improved by either increasing the library size, which is often restricted by technical limitations, or by changing to a more favourable library design [see also 16, 21, 41, 42]. Of the encoding schemes investigated here, 20/20-C is the most beneficial regarding expected coverage and relative efficiency, as it avoids the initial loss and suffers less from redundancy effects. It also prevents the bias for amino acids with a high number of codons shown by other schemes. As generating 20/20 libraries with the trimer technique is still rather expensive, the majority of current applications uses libraries with other encoding schemes. However, there are alternative techniques to trimer to reach a ratio of one codon per amino acid, like the MAX randomization [20], the "small-intelligent libraries" [22] and the ProxiMAX randomization [23]. Of these only ProxiMAX is suited to produce the longer randomized sequences needed for most peptide library applications [23].

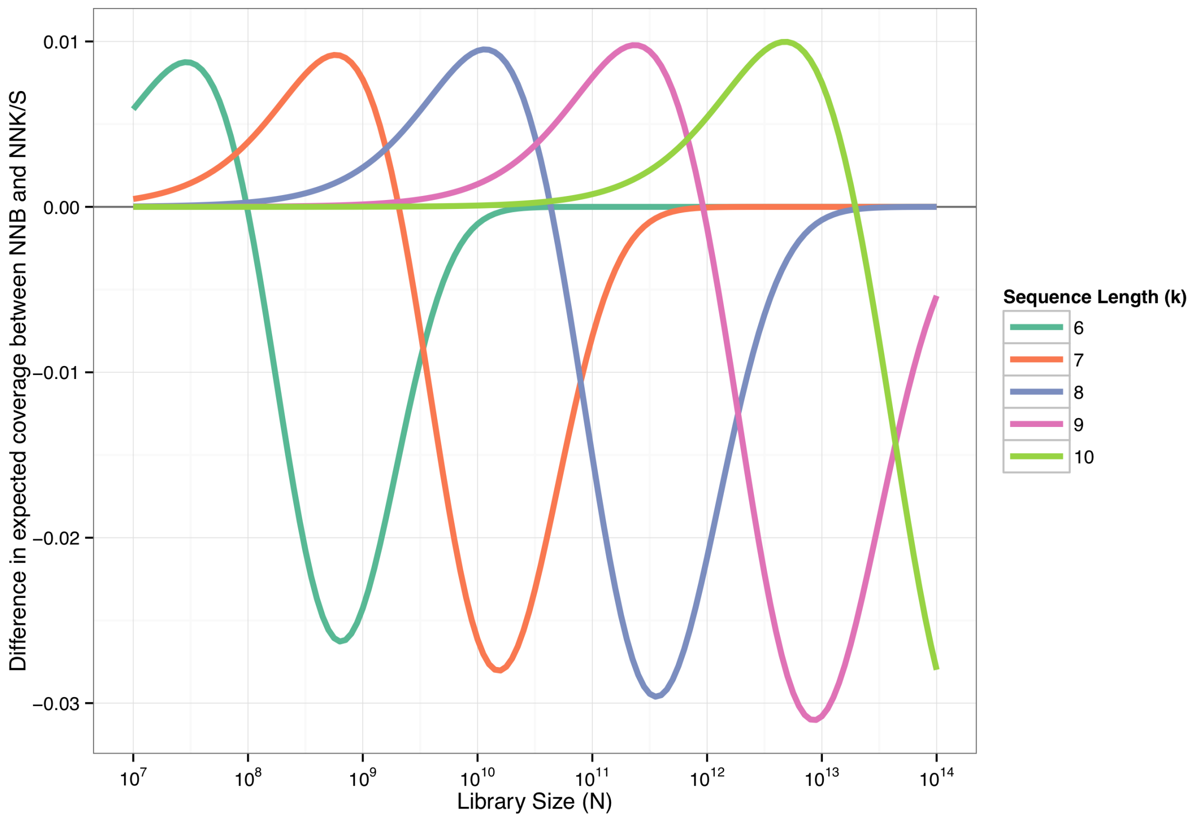


**Fig. 5. Difference in expected coverage between NNB and NNK/S libraries (with cysteines).** Initially, NNB libraries have a slight advantage in expected coverage over NNK/S libraries. Once a coverage of about 50% is reached, this pattern reverses and NNK/S libraries have a highere expected coverage. For very large libraries the difference in coverage is again, approaching zero (when libraries under both schemes have a coverage of almost 100%).

When comparing different library schemes regarding expected coverage and relative efficiency, NNK/S-C and NNB-C are very similar and preferable to NNN-C (see Figs. 1 and 2). NNK/S-C has a slight advantage over NNB-C in peptide diversity, expected coverage, and relative efficiency. If cysteines are considered as viable, however, NNB encoding has a minor advantage over NNK/S for libraries with a low expected coverage (Fig. 5). The initial advantage in expected coverage of NNB over NNK/S is due to the smaller initial loss of NNB: out of 48 codons, 47 are valid (corresponding to a 97.8% of valid codons), leading to a loss of $1 - (47/48)^k$, whereas NNK/S has 31 valid codons out of 32 (corresponding to a 96.9% of valid codons), leading to a (slightly) higher loss of $1 - (31/32)^k$. When peptide sequences including cysteine are also considered as invalid (in NNK/S-C and NNB-C schemes), the advantage of the initial loss disappears, because then an equal percentage of 93.75% of all codons are valid under either scheme.

NNK and NNS are mathematically identical but differ biologically due to different codon preferences of the host organisms. In *E. coli* and especially in *S. cerevisiae*, codon usage suggests that NNK may generally be the better option [19], while in human cells NNS codons are preferred. Another important design factor is the peptide length, as an elongation by one amino acid increases the number of possible peptides by a factor of 19 (20/20-C) to 23 (NNN with cysteines). When planning a new library, one should therefore consider the biological demands on peptide length on the one hand and the achievable coverage on the other. For all discussed encodings except 20/20 or 20/20-C, peptide length does not only influence the coverage but also the absolute number of viable peptides, as the chance that disruptive codons (stop codons and cysteines if relevant in the system) are included, increases with length. In fact, there is an optimal length that maximizes peptide diversity and relative efficiency for any given library size $N$ (Fig. 2). For example, for a non-20/20-C library of size $N = 10^8$ a peptide length of $k = 8$ is optimal in the sense, that its relative efficiency is larger than for libraries of peptide lengths 7 or 9. Therefore peptide diversity of a library of 8-peptides is also maximal.

Even extremely large libraries rarely exceed $N = 10^{10}$, using peptides longer than 9 to 10 amino acids therefore leads to a reduced peptide diversity in non-20/20-C libraries. In the case of an NNK-C library of 10 billion sequences about 40% less viable peptides are contained if a length of 18 amino acids is used instead of the optimal 9.

A high coverage is not always feasible due to limited library size and biological restraints on peptide length. Therefore, the chances that the "best" peptide is included in the library are often slim. However, peptides whose sequences are close to ideal may exist and perform similarly well [42]. By calculating the chance that at least one such peptide is contained, it is possible to better evaluate if a specific library is likely to produce high performing peptides. This chance depends on the used encoding scheme and the sequence of the "best" peptide. As this sequence is unknown beforehand, we define a best and worst case and determine a probability range for different library designs (see Fig. 4). The degree of variability is by far smallest for 20/20-C libraries indicating that such libraries should show the most reliable performance over different selections. With the worst case scenario as the most relevant qualifier, 20/20-C is again the best scheme followed by NNK/S-C, NNB-C and NNN-C, with differences spanning several orders of magnitude. About $5 \times 10^9$ sequences are needed in a heptapeptide 20/20-C library for a 99.5% chance that even in the worst case at least one top peptide from the first degree neighborhood is part of the library. With NNN-C about 2000-fold more ($10^{12}$) sequences are necessary.

In summary, our mathematical framework and its implementation at the web-interface PeLiCa offer evaluation parameters that allow an in-depth analysis of peptide libraries. This promotes a better understanding of library dynamics and enables a more informed design process. With the help of this mathematical framework libraries can be optimised directly for the requirements of the experiment and for the technical feasibility in a given setting. Therefore, our work contributes to improved peptide libraries which, in turn, will impact the success of viral and phage display systems in a multitude of scientific applications.

The authors would like to thank both anonymous reviewers for their constructive comments.

## References

1. Rodi DJ, Janes RW, Sanganee HJ, Holton RA, Wallace B, 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: 197–203.

2. Lu G, Zheng M, Zhu Y, Sha M, Wu Y, 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: 650–662.

3. Müller OJ, Kaul F, Weitzman MD, Pasqualini R, Arap W, Kleinschmidt JA, et al. (2003) Random peptide libraries displayed on adeno-associated virus to select for targeted gene therapy vectors. Nat Biotechnol 21: 1040–1046.

4. Binder M, Müller F, Jackst A, L'echenne B, Pantic M, Bacher U, et al. (2011) B-cell receptor epitope recognition correlates with the clinical course of chronic lymphocytic leukemia. Cancer 117: 1891–1900.

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

6. Bupp K, Roth MJ (2003) Targeting a retroviral vector in the absence of a known cell-targeting ligand. Hum Gene Ther 14: 1557–1564.

7. Sanchis J, Fernández L, Carballeira JD, Drone J, Gumulya Y, Höbenreich H, et al. (2008) Improved PCR method for the creation of saturation mutagenesis libraries in directed evolution: application to difficult-to-amplify templates. Appl Microbiol Biotechnol 81: 387–397.

8. Bougioukou DJ, Kille S, Taglieber A, Reetz MT (2009) Directed evolution of an enantioselective enoate-reductase: Testing the utility of iterative saturation mutagenesis. Advanced Synthesis & Catalysis 351: 3287–3305.

9. Deshayes K, Schaffer ML, Skelton NJ, Nakamura GR, Kadkhodayan S, Sidhu SS (2002) Rapid identification of small binding motifs with high-throughput phage display: discovery of peptidic antagonists of IGF-1 function. Chem Biol 9: 495–505.

10. DeGraaf ME, Miceli RM, Mott JE, Fischer HD (1993) Biochemical diversity in a phage display library of random decapeptides. Gene 128: 13–17.

11. Rodi DJ, Soares AS, Makowski L (2002) Quantitative assessment of peptide sequence diversity in M13 combinatorial peptide phage display libraries. J Mol Biol 322: 1039–1052.

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

13. Noren KA, Noren CJ (2001) Construction of high-complexity combinatorial phage display peptide libraries. Methods 23: 169–178.

14. Michelfelder S, Kohlschütter J, Skorupa A, Pfennings S, Müller OJ, Kleinschmidt JA, et al. (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: e5122.

15. Maersch S, Huber A, Büning H, Hallek M, Perabo L (2010) Optimization of stealth adeno-associated virus vectors by randomization of immunogenic epitopes. Virology 397: 167–175.

16. Denault M, Pelletier JN (2007) Protein library design and screening: working out the probabilities. Methods Mol Biol 352: 127–154.

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

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

19. Patrick WM, Firth AE (2005) Strategies and computational tools for improving randomized protein libraries. Biomol Eng 22: 105–112.

20. Hughes MD, Nagel DA, Santos AF, Sutherland AJ, Hine AV (2003) Removing the redundancy from randomised gene libraries. J Mol Biol 331: 973–979.

21. Kille S, Acevedo-Rocha CG, Parra LP, Zhang ZG, Opperman DJ, Reetz MT, et al. (2013) Reducing codon redundancy and screening effort of combinatorial protein libraries created by saturation mutagenesis. ACS Synth Biol 2: 83–92.

22. Tang L, Gao H, Zhu X, Wang X, Zhou M, Jiang R (2012) Construction of "small-intelligent" focused mutagenesis libraries using well-designed combinatorial degenerate primers. BioTechniques 52: 149–158.

23. Ashraf M, Frigotto L, Smith ME, Patel S, Hughes MD, Poole AJ, et al. (2013) ProxiMAX randomization: a new technology for non-degenerate saturation mutagenesis of contiguous codons. Biochem Soc Trans 41: 1189–1194.

24. Krumpe LR, Schumacher KM, McMahon JB, Makowski L, Mori T (2007) Trinucleotide cassettes increase diversity of T7 phage-displayed peptide library. BMC Biotechnol 7: 65.

25. Ono A, Matsuda A, Zhao J, Santi DV (1995) The synthesis of blocked triplet-phosphoramidites and their use in mutagenesis. Nucleic Acids Res 23: 4677–4682.

26. McConnell SJ, Kendall ML, Reilly TM, Hoess RH (1994) Constrained peptide libraries as a tool for finding mimotopes. Gene 151: 115–118.

27. Fukunaga K, 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.

28. Waterkamp DA, Müller OJ, Ying Y, Trepel M, Kleinschmidt JA (2006) Isolation of targeted AAV2 vectors from novel virus display libraries. J Gene Med 8: 1307–1319.

29. Naumer M, Ying Y, Michelfelder S, Reuter A, Trepel M, Müller OJ, et al. (2012) Development and validation of novel AAV2 random libraries displaying peptides of diverse lengths and at diverse capsid positions. Hum Gene Ther 23: 492–507.

30. Perabo L, Goldnau D, White K, Endell J, Boucas J, Humme S, et al. (2006) Heparan sulfate proteoglycan binding properties of adeno-associated virus retargeting mutants and consequences for their in vivo tropism. J Virol 80: 7265–7269.

31. Varadi K, Michelfelder S, Korff T, Hecker M, Trepel M, Katus HA, et al. (2012) Novel random peptide libraries displayed on AAV serotype 9 for selection of endothelial cell-directed gene transfer vectors. Gene Ther 19: 800–809.

32. Xie Q, Bu W, Bhatia S, Hare J, Somasundaram T, Azzi A, et al. (2002) The atomic structure of adeno-associated virus (AAV-2), a vector for human gene therapy. Proc Natl Acad Sci USA 99: 10405–10410.

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

34. Girod A, Ried M, Wobus C, Lahm H, Leike K, Kleinschmidt J, et al. (1999) Genetic capsid modifications allow efficient re-targeting of adeno-associated virus type 2. Nat Med 5: 1052–1056.

35. Firth AE, Patrick WM (2008) GLUE-IT and PEDELAA: new programmes for analyzing protein diversity in randomized libraries. Nucleic Acids Research 36: W281–285.

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

37. Kong Y (2009) Calculating complexity of large randomized libraries. J Theor Biol 259: 641–645.

38. Patrick WM, Firth AE, Blackburn JM (2003) User-friendly algorithms for estimating completeness and diversity in randomized proteinencoding libraries. Protein Engineering 16: 451-457.

39. Bosley AD, Ostermeier M (2005) Mathematical expressions useful in the construction, description and evaluation of protein libraries. Biomol Eng 22: 57–61.

40. Scott JK, Smith GP (1990) Searching for peptide ligands with an epitope library. Science 249: 386–390.

41. Reetz MT, Kahakeaw D, Lohmer R (2008) Addressing the numbers problem in directed evolution. Chembiochem 9: 1797–1804.

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

43. Michelfelder S, Lee MK, deLima Hahn E, Wilmes T, Kaul F, Müller O, et al. (2007) Vectors selected from adeno-associated viral display peptide libraries for leukemia cell-targeted cytotoxic gene therapy. Exp Hematol 35: 1766–1776.

44. Henikoff S, Henikoff JG (1992) Amino acid substitution matrices from protein blocks. Proceedings of the National Academy of Sciences 89: 10915-10919.

45. Volles MJ, Lansbury PT (2005) A computer program for the estimation of protein and nucleic acid sequence diversity in random point mutagenesis libraries. Nucleic Acids Res 33: 3667–3677.

46. Nov Y (2014) Probabilistic methods in directed evolution: library size, mutation rate, and diversity. Methods Mol Biol 1179: 261–278.

47. Hoebenreich S, Zilly FE, Acevedo-Rocha CG, Zilly M, Reetz MT (2014) Speeding up Directed Evolution: Combining the Advantages of Solid-Phase Combinatorial Gene Synthesis with Statistically Guided Reduction of Screening Effort. ACS Synth Biol .

48. Neuner P, Cortese R, Monaci P (1998) Codon-based mutagenesis using dimer-phosphoramidites. Nucleic Acids Res 26: 1223–1227.

49. Gaytan P, Roldan-Salgado A (2013) Elimination of redundant and stop codons during the chemical synthesis of degenerate oligonucleotides. Combinatorial testing on the chromophore region of the red fluorescent protein mKate. ACS Synth Biol 2: 453–462.

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

# List of Figures

# List of Tables

# Biomathematical Description of Synthetic Peptide Libraries – Supplementary Material

## Considerations to Theorem 1

### Proof of Theorem 1

Theorem 1 gives the expected value and the variance of the number of different peptides from a clone library of size $N$ as:

$$\begin{aligned}
E[Z_N] &= b\left(1-(1-b^{-1})^N\right) \approx b\left(1-e^{-N/b}\right). \\
\mathrm{Var}[Z_N] &= b\left[\left(1-b^{-1}\right)^N - \left(1-2b^{-1}\right)^N\right] - b^2\left[\left(1-b^{-1}\right)^{2N} - \left(1-2b^{-1}\right)^N\right] \\
&\approx b(e^{-N/b} - e^{-2N/b}) - Ne^{-2(N-1)/b}.
\end{aligned}$$

*Proof.* We will discuss the proof in a stepwise approach:

Let $Z_i$ be the number of selected peptides after $i$ subsequent draws, $1 \le i \le N$. After the first draw, i.e. $i = 1$, we know that there is a single peptide sequence $i$ the 'library', therefore $Z_1 = 1$.

Drawing one more peptide sequence opens up two possibilities: we either increase the number of different peptides in the library by one, or we draw a peptide sequence that we have already seen before. The probability to observe a new sequence depends on how many peptide sequences we have already observed ($Z_i$), and how many are still left ($b - Z_i$).

For draw $i+1$ we then get the following probabilities to either observe a new peptide, i.e. $Z_{i+1} = Z_i + 1$, or a previously drawn peptide, in which case the number of different peptides does not change, i.e. $Z_{i+1} = Z_i$:

$$\begin{aligned}
P(Z_{i+1} = Z_i) &= Z_i/b \\
P(Z_{i+1} = Z_i + 1) &= (b - Z_i)/b = (1 - Z_i/b)
\end{aligned}$$

For the expected number of selected peptides the above equations lead us to:

$$\begin{aligned} E[Z_{i+1}] &= E[Z_i]^2/b + (1 - E[Z_i]/b) \cdot (E[Z_i] + 1) = \\ &= 1 - E[Z_i]/b + E[Z_i] = 1 + (1 - 1/b)E[Z_i]. \end{aligned}$$

Since $Z_1 = 1$, we get for the $i$th draw an expected value of

$$\begin{aligned} E[Z_i] &= \sum_{k=0}^{i-1} (1 - 1/b)^k = b\left(1 - (1 - 1/b)^i\right) && (6) \\ &\approx b\left(1 - e^{-i/b}\right). && (7) \end{aligned}$$

The approximation holds for large values of $b$, and the expected diversity of a library of size $N$..

In deriving the variance, Let again $Z_{i+1}$ be the random variable for the number of different peptides at draw $i + 1$, with $Z_1 = 1$. Consider the conditional random variable $Z_{i+1} \mid Z_i$. Expected value and variance are easily derived from first principles:

$$\begin{aligned} E\left[Z_{i+1} \mid Z_i\right] &= b^{-1} Z_i \cdot Z_i + (1 - b^{-1} Z_i) \cdot (Z_i + 1) = 1 + (1 - b^{-1}) Z_i \\ E\left[Z_{i+1}^2 \mid Z_i\right] &= b^{-1} Z_i \cdot Z_i^2 + (1 - b^{-1} Z_i) \cdot (Z_i + 1)^2 \\ &= Z_i^2 (1 - 2b^{-1}) + Z_i (2 - b^{-1}) + 1 \\ \mathrm{Var}\left[Z_{i+1} \mid Z_i\right] &= E\left[Z_{i+1}^2 \mid Z_i\right] - E\left[Z_{i+1} \mid Z_i\right]^2 \\ &= b^{-1} Z_i (1 - b^{-1} Z_i) \end{aligned}$$

This leads to a recursive formula for the variance of $Z_i$:

$$\begin{aligned} \mathrm{Var}(Z_{i+1}) &= E\left[\mathrm{Var}(Z_{i+1} \mid Z_i)\right] + \mathrm{Var}\left[E(Z_{i+1} \mid Z_i)\right] \\ &= E\left[b^{-1} Z_i (1 - b^{-1} Z_i)\right] + \mathrm{Var}\left[1 + (1 - b^{-1}) Z_i\right] \\ &= b^{-1} E\left[Z_i\right] - b^{-2} E\left[Z_i^2\right] + (1 - b^{-1})^2 \mathrm{Var}\left[Z_i\right] \\ &= b^{-1} E\left[Z_i\right] - b^{-2} \mathrm{Var}\left[Z_i^2\right] - b^{-2} E\left[Z_i\right]^2 + (1 - b^{-1})^2 \mathrm{Var}\left[Z_i\right] \\ &= b^{-1} E\left[Z_i\right] - b^{-2} E\left[Z_i\right]^2 + \left(1 - 2b^{-1}\right) \mathrm{Var}\left[Z_i\right] \end{aligned}$$

Using the exact result for expected values in (6), the variance of $Z_{i+1}$ can be written as

$$\mathrm{Var}(Z_{i+1}) = \left(1-b^{-1}\right)^{i} - \left(1-b^{-1}\right)^{2i} + \left(1-2b^{-1}\right)\mathrm{Var}(Z_i)$$

The recursion is resolved as

$$\mathrm{Var}(Z_{i+1}) = \sum_{j=0}^{i} a_j c^{i-j} + c^i \mathrm{Var}(Z_1) \overset{\mathrm{Var}(Z_1)=0}{=} \sum_{j=0}^{i} a_j c^{i-j},$$

with $a_j = (1-b^{-1})^j - (1-b^{-1})^{2j} = x^j - (x^2)^j$ and $c = 1-2b^{-1}$.

The above summation leads to two geometric series for an explicit expression of the variance:

$$\begin{aligned}
\mathrm{Var}(Z_{i+1}) &= c^i \sum_{j=0}^{i} a_j \left(c^{-1}\right)^j \\
&= c^i \left[\sum_{j=0}^{i} \left(c^{-1}x\right)^j - \sum_{j=0}^{i} \left(c^{-1}x^2\right)^j\right] \\
&= \frac{x^{i+1}-c^{i+1}}{x-c} - \frac{(x^2)^{i+1}-c^{i+1}}{x^2-c}.
\end{aligned}$$

This expression can be simplified somewhat, because the denominators are simple fractions in $b$:

$$\begin{aligned}
x - c &= 1 - \frac{1}{b} - \left(1 - \frac{2}{b}\right) = \frac{1}{b} \\
x^2 - c &= 1 - \frac{2}{b} + \frac{1}{b^2} - \left(1 - \frac{2}{b}\right) = \frac{1}{b^2}.
\end{aligned}$$

$$\begin{aligned}
\mathrm{Var}(Z_{i+1}) &= b(x^{i+1}-c^{i+1}) - b^2((x^2)^{i+1}-c^{i+1}) \\
&= b\left[(1-b^{-1})^{i+1} - (1-2b^{-1})^{i+1}\right] - b^2\left[(1-b^{-1})^{2(i+1)} - (1-2b^{-1})^{i+1}\right]
\end{aligned}$$

A direct calculation of the variance as stated above is generally not possible due to limited precision, which leads to numerically unstable and, unfortunately, rather unpredictable results. Instead, we make use of a Taylor approximation of the second term, which yields in a first order approximation

$$(1-b^{-1})^{2(i+1)} - (1-2b^{-1})^{i+1} = b^{-2}(i+1)\cdot\left(1-2b^{-1}\right)^{i} + \mathcal{O}(ib^{-2}).$$

At this point, we can, again, safely use the approximations of the exponential function and get in summary for the diversity $Z_N$:

$$\begin{aligned} \mathrm{Var}[Z_N] &= b\left[\left(1-b^{-1}\right)^N - \left(1-2b^{-1}\right)^N\right] \\ &\quad -b^2\left[\left(1-b^{-1}\right)^{2N} - \left(1-2b^{-1}\right)^N\right] \\ &\approx b(e^{-N/b} - e^{-2N/b}) - Ne^{-2(N-1)/b} \end{aligned}$$

□

## Approximation Error in Theorem 1

The question is how close the approximation of the expected value in theorem 1 is:

$$(1-1/b)^N = e^{-N/b} + \mathcal{O}\left(\min\left(\frac{1}{b}e^{-N/b-\epsilon}, \frac{N}{b^2}\right)\right). \tag{8}$$

*Proof.* In Dörrie's elementary proof of problem 12 [50] (limits for sequences $\lim_{x\to\infty}(1+1/x)^x$ and $\lim_{x\to\infty}(1+1/x)^{x+1}$) we find the following inequality: for $x > 0$ and $m$ such that $1 \pm x/m > 0$ it holds, that

$$(1-x^2/m)e^x \leq (1+x/m)^m \leq e^x.$$

We can directly apply this to our situation with $x := -N/b$ and $m = N$. Then $1 \pm x/m = 1 \pm 1/b > 0$, which is true for $b > 1$ – this we can safely assume.

The above inequality gives us

$$(1-N/b^2)e^{-N/b} \leq (1-1/b)^N \leq e^{-N/b}.$$

This means that the error in equation (8) has an upper bound given by $Nb^{-2}e^{-N/b} = 1/b \cdot N/b \cdot e^{-N/b}$.

It is easy to see that the expression $N/b\ e^{-N/b}$ is positive with a maximum of $e^{-1}$ at $N = b$. Both for large and small values of $N/b$ this function goes rapidly to zero: For small values of $N/b$ the expression $N/b\ e^{-N/b} = \mathcal{O}(N/b)$, while for large values of $N/b$ the expression $N/b\ e^{-N/b} = \mathcal{O}(e^{-N/b-\epsilon})$ for some small $\epsilon > 0$.

Upper and lower bound combined give equation (8). □

## Peptide Library Schemes

According to their multiplicity and functionality we distinguish six classes of amino acids based on an NNN-C scheme (see S1 Table) and five classes of amino acids based on an NNB-C scheme (see S2 Table).

**S1 Table. NNN-C Library Scheme.**

| class | amino acids | size $s$ | # nucleotides c |
|---|---|---|---|
| A | S, L, R | 3 | 6 |
| B | A, G, P, T, V | 5 | 4 |
| C | I | 1 | 3 |
| D | D, E, F, H, K, N, Q, Y | 8 | 2 |
| E | M, W | 2 | 1 |
| Z | cysteine C, stop tags `TAG, TGA, TAA` | 2 | 2.5 |

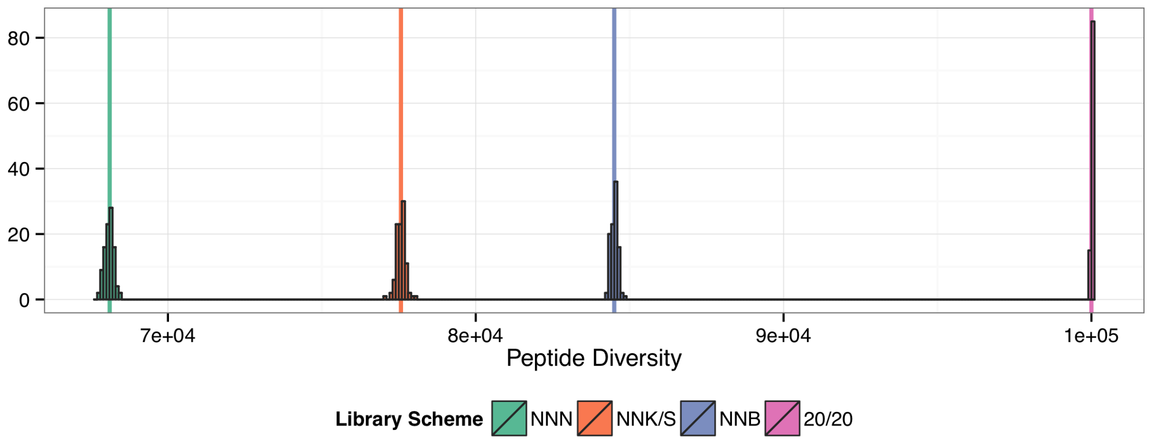
**S2 Table. NNB-C Library Scheme.**

| class | amino acids | size $s$ | # nucleotides c |
|---|---|---|---|
| A | S | 1 | 5 |
| B | L, R | 2 | 4 |
| C | A, G, P, T, V | 5 | 3 |
| D | D, F, H, I, N, Y | 6 | 2 |
| E | E, K, M, Q, W | 5 | 1 |
| Z | cysteine C, stop tag `TAG` | 2 | 1.5 |

## Simulation Results

A simulation scenario is used to validate results from the theoretical framework in practice.

We draw a peptide sequence of length $k$ from amino acids that are represented in their frequency by the number of codons specified by the respective library scheme. For the example of a NNK/S library scheme, this means amino acids S, L, and R are selected with probability 3/32, amino acids A, G, P, T, and V are selected with probability 2/32, while all other amino acids and a stop codon are selected with probability 1/32. For $k = 7$ a sample of these peptide sequences might look like: `KFRTVIR, RR*DISY, *LWAEPP, YAYEN*S, RMRQFWP, LYHPVIT, VNMMRHS, SEGGGRG, NYS*RT*, RA*LTAL`. A library is made up of a sample of $N$ of these sequences. Peptide diversity of the library is then determined by first removing all invalid peptide sequences (the ones that include the stop codon *), and secondly removing all duplicate peptide sequences. The number of peptide sequences left gives the diversity of the library.



**S1 Fig.. Simulation results: 100 libraries of size $10^5$ were sampled from each of the library schemes NNN, NNK/S, NNB, and 20/20, and the number of unique peptide sequences in each library was determined.** The resulting sampled peptide diversity is displayed in four histograms - one for each scheme. The vertical lines correspond to expected peptide diversity in each of these scenarios as given by equation (4).

The simulation was set up to sample $k$-peptide libraries of sizes between $10^4$ and $10^6$, for schemes NNN, NNB, NNK/S, and 20/20. $k$ is chosen as a length between 7 and 10. For each of these scenarios, 100 libraries are sampled, diversity is determined for each one of them. The results are shown in the S1 Fig. and S3 Table. The S1 Fig. shows histograms of sampled library diversity of 8-peptide libraries of size $N = 10^5$ under all four library schemes. NNN results in the lowest diversity number, while 20/20 libraries are almost perfect – only two libraries had two duplicate peptides, while another 13 libraries had a single duplicate. Here, the NNB scheme has a slightly higher diversity than the NNK/S scheme, because of the smaller initial loss under NNB and the relatively small library size ($N = 10^5$), which is much lower than half the number of possible peptides, at which point diversity would tip in favor of NNK/S.

The S3 and S4 Tables give an overview of observed versus expected library diversity and the variance for each of the scenarios. It is obvious, that both expected and observed values are extremely close to each other.

**S3 Table. Observed library diversity compared to expected library diversity for different library sizes $N$ under various library schemes.**

| $N$ | $k$ | NNN | | NNK/S | | NNB | | 20/20 | |
|---|---|---|---|---|---|---|---|---|---|
| | | Exp. | Obs. | Exp. | Obs. | Exp. | Obs. | Exp. | Obs. |
| $10^4$ | 7 | 7145.67 | 7144.12 | 8007.12 | 8004.38 | 8629.59 | 8624.94 | 9999.96 | 9999.93 |
| | 8 | 6810.81 | 6816.74 | 7756.99 | 7754.60 | 8449.92 | 8446.89 | 10000.00 | 10000.00 |
| | 9 | 6491.55 | 6492.26 | 7514.59 | 7507.55 | 8273.89 | 8274.90 | 10000.00 | 10000.00 |
| | 10 | 6187.26 | 6186.22 | 7279.76 | 7269.77 | 8101.51 | 8097.81 | 10000.00 | 10000.00 |
| $10^{4.5}$ | 7 | 22596.05 | 22593.02 | 25320.19 | 25315.05 | 27288.56 | 27289.27 | 31622.61 | 31621.65 |
| | 8 | 21537.77 | 21535.01 | 24529.90 | 24530.59 | 26721.14 | 26711.42 | 31622.98 | 31622.00 |
| | 9 | 20528.24 | 20534.84 | 23763.39 | 23777.29 | 26164.51 | 26161.31 | 31623.00 | 31622.00 |
| | 10 | 19565.98 | 19568.83 | 23020.79 | 23014.38 | 25619.42 | 25617.92 | 31623.00 | 31622.00 |
| $10^5$ | 7 | 71447.47 | 71412.17 | 80061.71 | 80054.17 | 86285.28 | 86278.78 | 99996.09 | 99996.17 |
| | 8 | 68107.53 | 68090.65 | 77569.38 | 77575.83 | 84498.58 | 84502.26 | 99999.80 | 99999.83 |
| | 9 | 64915.52 | 64897.07 | 75145.89 | 75151.60 | 82738.83 | 82720.42 | 99999.99 | 100000.00 |
| | 10 | 61872.63 | 61868.48 | 72797.61 | 72800.58 | 81015.14 | 81022.08 | 100000.00 | 100000.00 |
| $10^{5.5}$ | 7 | 225867.04 | 225885.04 | 253105.56 | 253125.86 | 272777.22 | 272777.49 | 316188.94 | 316188.60 |
| | 8 | 215371.07 | 215345.48 | 245291.95 | 245264.41 | 267203.41 | 267229.60 | 316226.05 | 316225.18 |
| | 9 | 205280.81 | 205238.30 | 237632.11 | 237628.82 | 261643.07 | 261643.24 | 316227.90 | 316226.88 |
| | 10 | 195658.58 | 195648.39 | 230206.42 | 230193.84 | 256192.54 | 256150.85 | 316228.00 | 316227.00 |
| $10^6$ | 7 | 713556.98 | 713548.28 | 799671.48 | 799695.85 | 861788.73 | 861781.77 | 999609.48 | 999612.41 |
| | 8 | 681022.51 | 681031.98 | 775639.16 | 775606.04 | 844923.26 | 844897.72 | 999980.47 | 999979.07 |
| | 9 | 649152.14 | 649150.04 | 751455.76 | 751392.16 | 827384.63 | 827344.54 | 999999.02 | 999999.10 |
| | 10 | 618726.15 | 618751.93 | 727975.96 | 727960.55 | 810151.20 | 810162.17 | 999999.95 | 999999.97 |

Observed library diversity is based on averages from 100 simulation runs.

## Examples for inclusion probability

The S5 Table gives the inclusion probabilities for some example sequences from a heptapeptide library of size 100 Million under common library schemes. For comparison, inclusion probabilities of the sequence itself (denoted by degree 0) are given as well as the probabilities to detect at least one sequence from the degree-1 and degree-2 neighborhoods, respectively.

While the probabilities to detect individual sequences are highly dependent on the specific sequence and the library scheme, inclusion probabilities quickly grow as first and second degree neighborhoods (based on positive BLOSUM80 scores) are regarded.

The exemplary peptides in the S5 Table cover the maximal range of inclusion probabilities. While the amino acids S, L and R are represented by 6 codons each in NNN-C (probability of 6/64), M is only encoded by ATG (probability of 1/64). The inclusion probability of `SLRSLRS` is therefore much higher than that of `MMMMMMM` if the NNN encoding scheme is used. However, with a 20/20-C encoding scheme in which each amino acid has only one representation, all individual peptides have the same inclusion probability

**Table S4. Variances of peptide diversity for different library sizes $N$ under four library schemes.**

| $N$ | $k$ | NNN |  | NNK/S |  | NNB |  | 20/20 |  |
|---|---|---|---|---|---|---|---|---|---|
|  |  | Exp. | Obs. | Exp. | Obs. | Exp. | Obs. | Exp. | Obs. |
| $10^4$ | 7 | 2054.60 | 2026.43 | 1595.72 | 1537.19 | 1197.52 | 1069.21 | 0.04 | 0.09 |
|  | 8 | 2189.10 | 2192.64 | 1739.90 | 1670.48 | 1326.80 | 1314.22 | 0.00 | 0.00 |
|  | 9 | 2296.53 | 2338.82 | 1867.68 | 1682.03 | 1447.17 | 1317.71 | 0.00 | 0.00 |
|  | 10 | 2380.04 | 2088.50 | 1980.27 | 2213.82 | 1559.06 | 1831.37 | 0.00 | 0.00 |
| $10^{4.5}$ | 7 | 6465.33 | 6539.01 | 5046.88 | 4242.51 | 3754.64 | 4489.71 | 0.39 | 0.27 |
|  | 8 | 6885.84 | 7526.29 | 5502.11 | 5992.65 | 4158.98 | 3783.98 | 0.00 | 0.00 |
|  | 9 | 7221.22 | 7623.67 | 5906.17 | 5303.88 | 4535.29 | 4762.18 | 0.00 | 0.00 |
|  | 10 | 7480.99 | 6893.42 | 6262.20 | 5791.51 | 4884.81 | 4564.72 | 0.00 | 0.00 |
| $10^5$ | 7 | 20419.13 | 19807.40 | 15966.72 | 15161.66 | 11844.98 | 11374.46 | 3.91 | 4.30 |
|  | 8 | 21738.28 | 19810.29 | 17399.49 | 19313.56 | 13115.06 | 12104.80 | 0.00 | 0.18 |
|  | 9 | 22794.27 | 20270.23 | 18676.84 | 19134.65 | 14300.66 | 18030.45 | 0.00 | 0.00 |
|  | 10 | 23611.41 | 23954.90 | 19802.69 | 16990.43 | 15401.61 | 18344.03 | 0.00 | 0.00 |
| $10^{5.5}$ | 7 | 64606.21 | 49532.42 | 50562.89 | 44056.85 | 37492.17 | 41781.20 | 39.07 | 39.64 |
|  | 8 | 68709.44 | 84265.04 | 55026.21 | 49145.92 | 41438.66 | 39691.54 | 1.31 | 2.09 |
|  | 9 | 72040.93 | 66343.75 | 59061.65 | 64609.97 | 45181.64 | 35111.19 | 0.00 | 0.13 |
|  | 10 | 74620.47 | 86011.65 | 62621.65 | 57106.12 | 48658.79 | 37666.31 | 0.00 | 0.00 |
| $10^6$ | 7 | 204953.80 | 230626.37 | 160601.24 | 117657.22 | 119305.32 | 101954.83 | 390.20 | 470.24 |
|  | 8 | 217279.94 | 209733.96 | 174049.48 | 133598.16 | 131038.29 | 126377.19 | 17.48 | 23.36 |
|  | 9 | 227774.21 | 215835.33 | 186771.60 | 176358.32 | 142836.53 | 130808.86 | 0.00 | 0.98 |
|  | 10 | 235925.15 | 201751.50 | 198027.05 | 245600.27 | 153827.10 | 115739.96 | 0.00 | 0.03 |

Expected variances are based on equation (5), observed variances are based on 100 simulation runs each.

(probability for each amino acid is 1/19). This has the same effect on the neighborhoods of first and second degree. Because `SLRSLRS` is higly probable under NNN-C, its first and second neighborhoods are large. This makes it very likely that one of these sequences is found in an NNN-C library. The sequence `MMMMMMM` is highly unlikely, because it is only encoded once among the $64^7$ peptides of an NNN-C scheme of heptapeptides. This implies that its first and second degree neighborhoods are also very small (and therefore fairly unlikely to be found) in an NNN-C library. Under a 20/20-C scheme these probabilities reverse, and neighborhood sizes are completely determined by the exchangeability of amino acids: amino acids `S, L`, and `R` are not as exchangeable (based on positive BLOSUM80 scores) as `M`. `M` can be exchanged by amino acids `I, L`, and `V`, whereas amino acids `S, R` only have two possibilities for an exchange each, resulting in correspondingly smaller degree 1 and 2 neighborhoods for `SLRSLRS` than `MMMMMMM`.

**Table S5. Inclusion probabilities of specific heptapeptide sequences (degree 0) and first and second degree neighborhoods (degree 1 and degree 2) under different library schemes in a library of size 100 Million DNA Sequences.**

| peptide | degree | NNN-C | NNB-C | NNK/S-C | 20/20-C |
|---|---|---|---|---|---|
| `SLRSLRS` | 0 | 1.0000 | 0.9998 | 1.0000 | 0.1058 |
| | 1 | 1.0000 | 1.0000 | 1.0000 | 0.8507 |
| | 2 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| `NEAREST` | 0 | 0.1690 | 0.0918 | 0.1518 | 0.1058 |
| | 1 | 0.9099 | 0.7650 | 0.8823 | 0.7912 |
| | 2 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| `HENNING` | 0 | 0.0153 | 0.0254 | 0.0091 | 0.1058 |
| | 1 | 0.1775 | 0.2931 | 0.1282 | 0.7388 |
| | 2 | 0.7112 | 0.8930 | 0.6065 | 0.9996 |
| `MMMMMMM` | 0 | 0.0000 | 0.0003 | 0.0046 | 0.1058 |
| | 1 | 0.0037 | 0.0170 | 0.1785 | 0.9147 |
| | 2 | 0.1361 | 0.3765 | 0.9741 | 1.0000 |

## Validation

The intersecting properties of GLUE-IT and PeLiCa offer the possibility to directly compare values for peptide diversity and coverage for the following libraries: Peptide length: 6; Encoding schemes: NNN, NNB, NNK and NNS with variable library sizes; cysteines regarded as viable. Both PeLiCa and GLUE-IT offer identical values for libraries of sizes 10 million, 100 million and 1 billion as indicated in the S6 and S7 Tables.

**Table S6. PeLiCa vs GLUE-IT comparison of peptide diversity/expected number of distinct amino acid variants in hexapeptides.**

| | Library Size ($N$) | Ten Million ($10^7$) | Hundred Million ($10^8$) | One Billion ($10^9$) |
|---|---|---|---|---|
| Scheme | Software | | | |
| NNN | GLUE-IT | 6.206 e+06 | 2.970 e+07 | 5.770 e+07 |
| | PeLiCa | 6.2058e+06 | 2.9696e+07 | 5.7705e+07 |
| NNB | GLUE-IT | 7.296 e+06 | 3.345 e+07 | 5.961 e+07 |
| | PeLiCa | 7.2925e+06 | 3.3447e+07 | 5.9612e+07 |
| NNK/S | GLUE-IT | 6.918 e+06 | 3.347 e+07 | 6.116 e+07 |
| | PeLiCa | 6.9179e+06 | 3.3468e+07 | 6.1165e+07 |

**Table S7. PeLiCa vs GLUE-IT comparison of Coverage/Completeness of hexapeptides under different library schemes and library sizes.**

| | Library Size ($N$) | Ten Million ($10^7$) | Hundred Million ($10^8$) | One Billion ($10^9$) |
|---|---|---|---|---|
| Scheme | Software | | | |
| NNN | GLUE-IT | 0.09697 | 0.1140 | 0.1081 |
| | PeLiCa | 0.09697 | 0.11400 | 0.10809 |
| NNB | GLUE-IT | 0.4640 | 0.5226 | 0.5229 |
| | PeLiCa | 0.46401 | 0.52261 | 0.52294 |
| NNK/S | GLUE-IT | 0.9016 | 0.9314 | 0.9557 |
| | PeLiCa | 0.90163 | 0.93144 | 0.9557 |