A Framework and Application for Efficient Analysis of Peptide Libraries

Eric Riemer Hare

October 19, 2013

1 Introduction

Peptide libraries have immensely important biological and medical applications[REF]. However, analysis of properties of these libraries is a challenging task. With 20 potential amino acids and a peptide length k, there are 20^k distinct peptides that can be encoded. The situation is further complicated by the fact that different sets of codons can encode the same amino acid. This phenomenon is illustrated in Figure 1. As this wheel represents the possibilities for only one of the k positions in the peptide sequence, its clear that this problem becomes exponentially more complex as the peptide length increases.

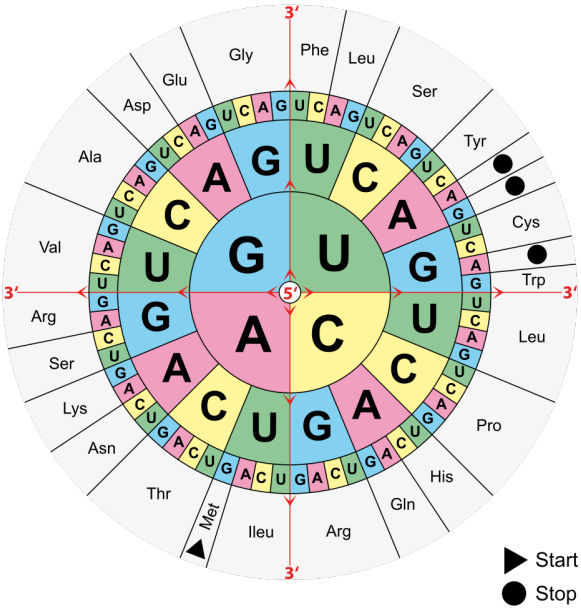

Figure 1: Codon wheel illustrating the sequences of codons leading to each amino acid, as well as start and stop tags.

In order to assess the quality of a particular library of peptides, some building blocks are needed in order to proceed. Specifically, treating the presence of each amino acid in a peptide as independent of any other amino acid in that sequence, we can define the joint distribution:

$$P(A_1 = a_1, A_2 = a_2, ..., A_k = a_k) = P(A_1 = a_1)P(A_2 = a_2) \cdots P(A_k = a_k)$$

A_1 through A_k represent random variables associated with the amino acid at each of positions one through k in the peptide sequence. The need for a clean and consise representation of jointly discrete random variables is clear. I will now introduce the software-based building blocks used to represent peptides.

2 Backend

There are two major components that represent the backend of the application and analysis. The modular nature of these components are convenient for extensions to this work, both in the analysis of peptide libraries, and any other application in which jointly discrete random variables are useful as a representation of particular phenomena.

2.1 discreteRV

Dr. Andreas Buja, professor of Statistics at the University of Pennsylvania, created a set of R functions implementing discrete random variables[REF]. These functions were compiled and documented in a package called discreteRV. discreteRV is available for download on the Comprehensive R Archive Network (CRAN)[REF].

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

2.1.1 Probabilities

The centerpiece of discreteRV is a set of functions to create and manipulate discrete random variables. A list of these functions and brief discriptions of their functionality is available in table **??**. A random variable is defined through the use of the make.RV function. make.RV accepts a vector of probabilities and a vector of outcome values, and returns an RV object.

R Example 2.1

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

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