Quick Permutation Test for rapid filtering of n-gram data

Piotr Sobczyk, Paweł Mackiewicz and Michał Burdukiewicz

June 2016

1 Abstract

Lorem ipsum dolor sit amet, est ad doctus eligendi scriptorem. Mel erat falli ut. Feugiat legendos adipisci vix at, usu at laoreet argumentum suscipiantur. An eos adhuc aliquip scriptorem, te adhuc dolor liberavisse sea. Ponderum vivendum te nec, id agam brute disputando mei.

2 Introduction

N-grams (k-mers) are vectors of n characters derived from input sequences, widely used in genomics, transcriptomics and proteomics. Despite the continuous interest in the sequence analysis, there are only a few tools tailored for comparative n-gram studies. Furthermore, the volume of n-gram data is usually very large, making its analysis in **R** especially challenging.

The CRAN package *biogram* (Burdukiewicz *et al.* 2015) facilitates incorporating n-gram data in the **R** workflows. Aside from the efficient extraction and storage of n-grams, the package offers also a feature selection method designed specifically for this type of data. QuiPT (Quick Permutation Test) uses several filtering criteria such as information gain (mutual information) to choose significant n-grams. To speed up the computation and allow precise estimation of small p-values, QuiPT uses analytically derived distributions instead of a large number of permutations. In addition to this, *biogram* contains tools designed for reducing the dimensionality of the amino acid alphabet (Murphy *et al.* 2000), further scaling down the feature space.

To illustrate the usage of n-gram data in the analysis of biological sequences,
we present two case studies performed solely in **R**. The first,
prediction of amyloids, short proteins associated with the number of clinical
disorders as Alzheimer's or Creutzfeldt-Jakob's diseases (Fändrich 2012), employs random forests (Wright & Ziegler 2015) trained on n-grams. The second, detection
of signal peptides orchestrating an extracellular transport of proteins, utilizes more complicated probabilistic framework (Hidden semi-Markov model,) but still uses n-gram data for training.

3 Methods

The standard approach for filtering k-mer features is a permutation test (Fisher 1935) , which verifies if a presence or an absence of a specific k-mer is related to the label of the sequence. We propose new framework called QuiPT, an exact version of permutation test, which is more accurate, faster and as customizable, in terms of measure of dependence we use, as permutation test itself.

3.1 Permutation test

Let us consider target $y = (y_1, ..., y_n)$ and matrix of features $X \in M_{n \times p}$. More specifically, as we focus on a method for filtering k-mer, we shall focus on one specific feature $x = (x_1, ..., x_n)$. When x origins from positioned k-mer analysis then it is binary. In this paper we assume that y is also binary.

The standard approach [cite] is to define some measure $C(x, y)$ of dependence between x and y. Then in permutation test we reshuffle feature and target indpenedently large number of times and compute approximate p-value for how extreme is the dependency between x and y measured with $C(\cdot, \cdot)$.

$$p_{value} \approx \frac{\sum_i^m C(s_i(x), s_i(y)) > C(x, y)}{m}$$

where m is number of permutations, and s_i denotes shuffling in i-th permutation.

3.2 Quick Permutation Test - QuiPT

In this secion we shall outline how one can avoid performing huge number of permutations and get exact p-values. The main idea of this approach was previously described in http://www.cytel.co/hs-fs/hub/1670/file-2600444346-pdf/lib/pubs/exact-inference-for-categorical-data.pdf and we shall briefly state what is that idea based on, and we show how we managed to speed it up.

Let us consider the contingency table for y and x:

target\feature	1	0
1	$n_{1,1}$	$n_{1,0}$
0	$n_{0,1}$	$n_{0,0}$

If probability that target equals 1 is p and probability that feature equals 1 is q and feature and target are independant then each of them has the following probabilities

$$P((Target, Feature) = (1, 1)) = p \cdot q$$

$$P((Target, Feature) = (1, 0)) = p \cdot (1 - q)$$

$$P((Target, Feature) = (0, 1)) = (1 - p) \cdot q$$

$$P((Target, Feature) = (0, 0)) = (1 - p) \cdot (1 - q)$$

This means that a target-feature can be described as multinomial distribution.

$$\binom{n}{n_{1,1}} (p{\cdot}q)^{n_{1,1}} \binom{n - n_{1,1}}{n_{1,0}} (p{\cdot}(1{-}q))^{n_{1,0}} \binom{n - n_{1,1} - n_{1,0}}{n_{0,1}} ((1{-}p){\cdot}q)^{n_{0,1}} \binom{n - n_{1,1} - n_{1,0} - n_{0,1}}{n_{0,0}} ((1{-}p){\cdot}(1{-}q))^{n_{0,0}}$$

with additional restriction that $n_{1,\cdot} = n_{1,1} + n_{1,0}$ and $n_{\cdot,1} = n_{1,1} + n_{0,1}$ are known and fixed as they describe the number of „ones" for target and feature respectively.

The introduction of these restrictions is not an unnecessary complication, but in fact simplifies computations, because we can describe whole confusion matrix using only $n_{1,1}$.

Observe that $n_{1,1}$ is from range $[0, min(n_{\cdot,1}, n_{1,\cdot})]$. So we get probability of certain contingency table as conditional distribution, as impose restrictions on two parameters $n_{\cdot,1}$ and $n_{1,\cdot}$. We can compute IG for each possible value of $n_{1,1}$ and finally we get distribution of Information Gain under hypothesis that target and feature are independant.

Having exact distribution allows us to perform permutation test much quicker as we no longer need large number of replications. Furthermore, by using exact test we will get precise values of even in tails of statistic distribution, which was not guaranteed with random permutations. In fact, suppose we want to perform test with $\alpha = 10^{-8}$, which is not uncommon value when we adjust for multiple testing - more on that later. Even for huge number of permuations $m = 10^8$ standard deviation of permutation test estimate $\frac{p(1-p)}{m}$ is roughly equal true p-value itself.

In the context of k-mer data we can speed up our algorithm even further. Note that since target y is common for testing all k-mer features, test statistics depends only on number of positive cases in feature $n_{\cdot,1}$. Though we test millions of features, there are just few distributions that we need to compute, as usually number of positives in k-mer is small. We take advantage of this fact, and therefore complexity of our algorithm is roughly equal $n * p$.

Furthermore, as we deal with sparse vectors, our implementation in R is optimized by using libraries **bit**, which speeds up computing entropy and information gain, and **slam**, for sparse encoding of data. Thanks to those details **biogram** works with even very large data sets, being limited only by available RAM.

??? GIVE SOME EXAMPLE OF BIG DATA HANDLED ????

Size	matrix [bytes]	slam [bytes]
1e+01	1000	1032
1e+04	80200	1032
1e+09	8000200	1032
1e+16	800000200	1032

Caption: Size of data set (a matrix with the number of records equal to the size) in the memory. matrix is a standard R matrix, while slam is the size of the object created using the slam package.

3.2.1 Relationship with Fisher's exact test

Fisher's exact test is a test for independance in contingency table (mostly 2×2). From derivation provided in [citation Lehmann], it becomes obvious that what we did in previous section is providing a framework for heuristics in two-tailed Fisher's exact test, for which there is no right solution.

In fact, QuiPT can give different results than Fisher's exact test as implemented in R.

```
library(biogram)
n11 <- 15
n10 <- 0
n01 <- 30
n00 <- 15

target_feature <- create_feature_target(n11 = n11, n01 = n01, n10 = n10, n00 = n00)
m <- matrix(c(n11,  n10,  n01,  n00), ncol = 2)
fisher.test(x = m, alternative = "t")
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  m
## p-value = 0.01281
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  1.499432      Inf
## sample estimates:
## odds ratio
##        Inf
```

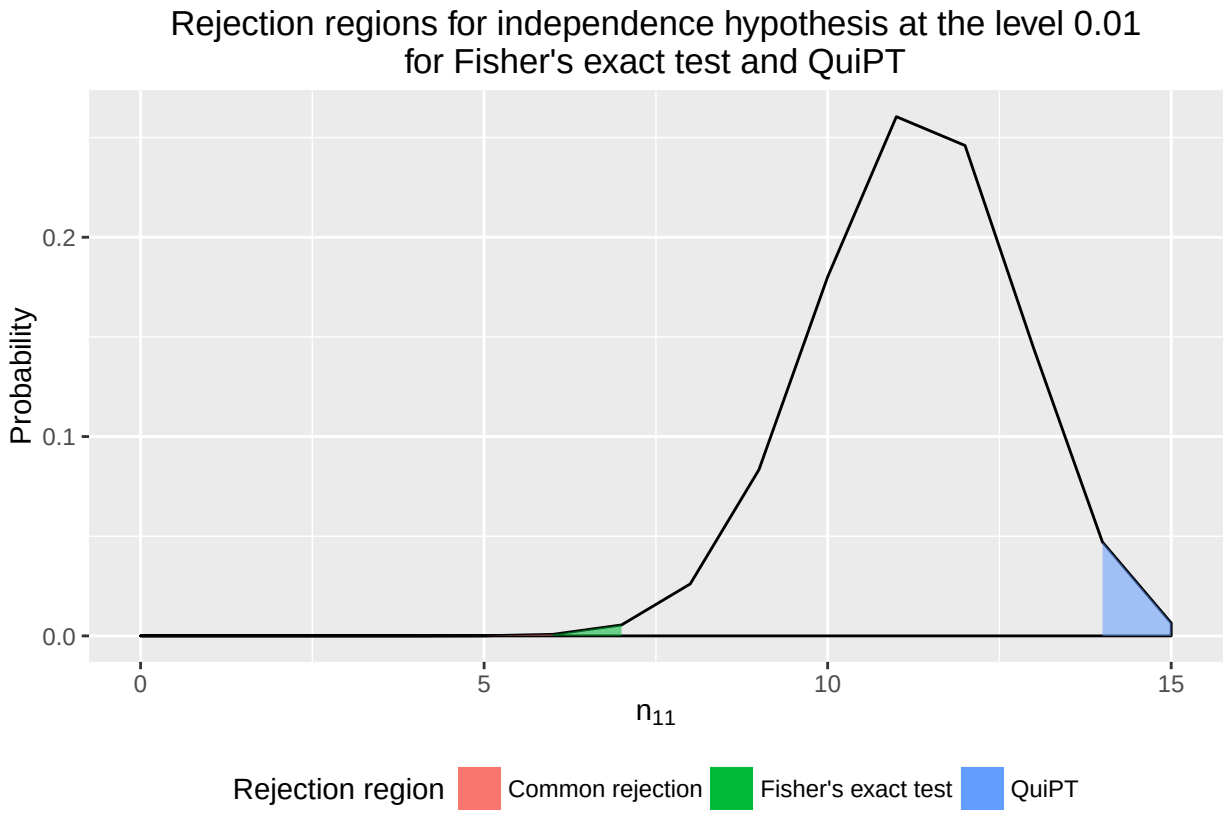


Figure 1: $n = 60, n_{1\bullet} = 15, n_{\cdot 1} = 45$

```
test_features(target_feature[, 1], cbind(target_feature[, 2], target_feature[, 2]))
```

```
##    feature1    feature2
## 0.007322474 0.007322474
```

Where does the difference come from?

For $n_{11} = 7$ we have higher probability of occurence (0.0054897) smaller than one for $n_{11} = 15$ - 0.0064832, which means that Fisher's test is more likely to reject 7 than 15. However Information Gain for $n_{11} = 15$ equal 0.0849495 is higher than Information Gain for $n_{11} = 7$ equal 0.0654349 which means that QuiPT would be more likely to reject it.

3.3 Criteria

3.3.1 Information Gain

3.3.1.0.1 Definition - entropy

For a discrete random variable we define entropy by:

$$H(X) = -\sum^{m} p_j log p_j$$

For a Bernoulli r.v. we get a simplified expression

$$H(X) = -p \cdot log(p) - (1-p) \cdot log(1-p)$$

3.3.1.0.2 Definition - conditional entropy

For two discrete r.v. X, Y we define average conditional entropy by:

$$H(Y|X) = \sum_j P(X = v_j)H(Y|X = v_j)$$

If X and Y are Bernoulli's then:

$$H(Y|X) = q \cdot H(Y|X = 1) + (1 - q) \cdot H(Y|X = 0)$$

3.3.1.0.3 Definition - Information gain

$$IG(Y|X) = H(Y) - H(Y|X)$$

3.3.2 χ^2

3.3.3 Kullback-Leibler

3.3.4 Bhattacharyya distance

3.3.5 Mahalanobis distance

3.4 Encoding distance

The encoding distance, as computed per the calc_ed function, between a and b is defined as the minimum number of amino acids that have to be moved between subgroups of encoding to make a identical to b (the order of subgroups in the encoding and amino acids in a group is unimportant).

The encoding distance may be further normalized by a factor reflecting how much moving amino acids between groups altered mean group properties.

Table 3: Encoding A

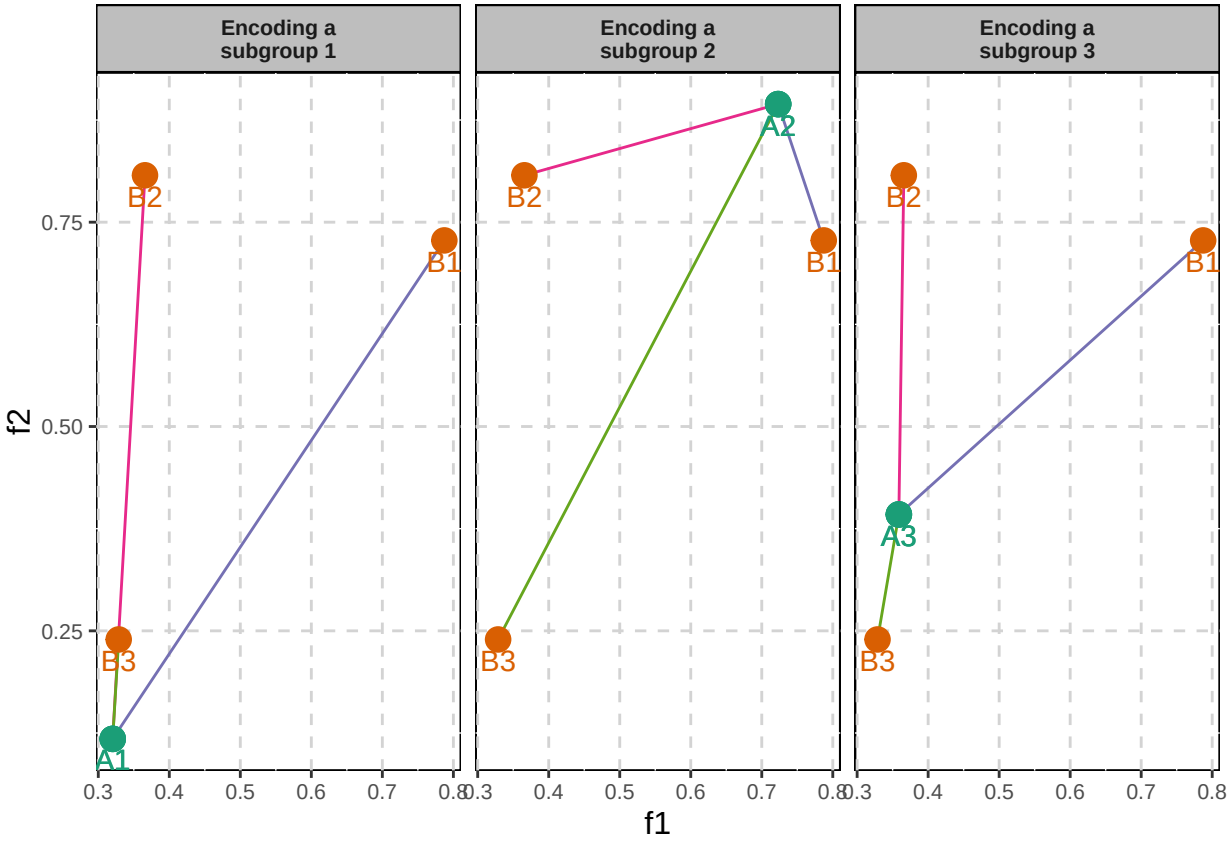
ID	Groups
1	P
2	F, I, W, Y
3	A, C, D, E, G, H, K, L, M, N, Q, R, S, T, V

Table 4: Encoding B

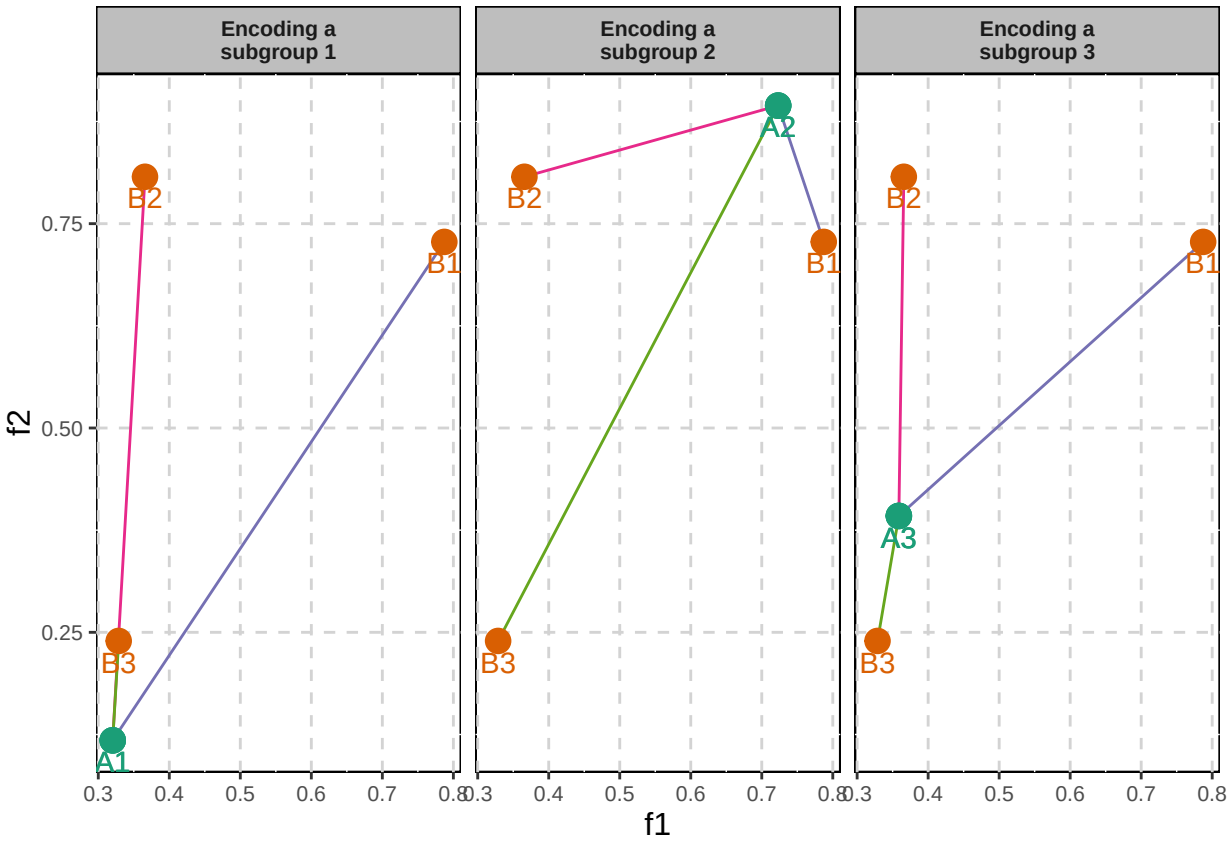
ID	Groups
1	F, R, W, Y
2	C, I, L, T, V
3	A, D, E, G, H, K, M, N, P, Q, S

The encoding distance between both encodings is 7.

3.5 Normalized encoding distance



The figure above represents the distances between groups of encoding **a** (green dots) and groups of encoding **b** (red dots). The position of the dot defined by mean values of specific properties of all amino acids belonging to the group.



	Enc a, group 1	Enc a, group 2	Enc a, group 3
Enc b, group 1	0.7681436	0.1786769	0.5441119
Enc b, group 2	0.6910472	0.3677947	0.4150182
Enc b, group 3	0.1219397	0.7645517	0.1558620

For each group in encoding **a**, we choose the minimum distance d_i between the group and any group belonging to the encoding **b**. The normalization factor is equal to the sum of minimum distances for each group in **a**.

$$n_f = \sum_{i \in 1,2,3} \min\left(d_i\right)$$

In the case depicted above, n_f is equal to the sum of 0.1219, 0.1787, 0.1559: 0.4565.

4 Simulations

Compute how drastic must be the difference of fractions in both groups.
Simulation study (motif length, sequence lenth, alphabet length) - sensitivity.

References

Burdukiewicz, M., Sobczyk, P. & Lauber, C. (2015). *Biogram: Analysis of biological sequences using n-grams.* Retrieved from http://CRAN.R-project.org/package=biogram

Fändrich, M. (2012). Oligomeric Intermediates in Amyloid Formation: Structure Determination and

Mechanisms of Toxicity. *Journal of Molecular Biology*, **421**, 427–440. Retrieved July 24, 2015, from http://www.sciencedirect.com/science/article/pii/S0022283612000277

Fisher, R. (1935). *The design of experiments*. Oliver; Boyd. Retrieved from https://books.google.pl/books?id=-EsNAQAAIAAJ

Murphy, L.R., Wallqvist, A. & Levy, R.M. (2000). Simplified amino acid alphabets for protein fold recognition and implications for folding. *Protein Engineering*, **13**, 149–152. Retrieved January 24, 2016, from http://peds.oxfordjournals.org/content/13/3/149

Wright, M.N. & Ziegler, A. (2015). Ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *arXiv:1508.04409 [stat]*. Retrieved March 6, 2016, from http://arxiv.org/abs/1508.04409