Computing the tRNA pairing index (TPI) for multiple organisms
Semesterarbeit

Author(s):
Zaugg, Gerry

Publication Date:
2003

Permanent Link:
https://doi.org/10.3929/ethz-a-004696377

Rights / License:
In Copyright - Non-Commercial Use Permitted
Computing the tRNA Pairing Index (TPI) for Multiple Organisms

Semesterarbeit
Summer Semester 2003
ETH Zürich, October 17, 2003

Gerry Zaugg

Prof. Gaston H. Gonnet
Advisor: Peter von Rohr
Abstract

The tRNA Pairing Index (TPI) has been developed to measure the autocorrelation in tRNA usage being independent of codon, tRNA, or amino acid bias. In this paper, the TPI is computed and analysed for the six organisms *Bacillus subtilis, Caenorhabditis elegans, Escherichia coli, Schizosaccharomyces pombe, Homo sapiens* and *Saccharomyces cerevisiae*. Beside the already implemented exact recursive method, a Monte Carlo approach has been implemented. The problem of twin exclusion is discussed. Outliers with respect to computing time are analysed. Several approximations of the skewed probability distribution of the number of changes for a given set of tRNAs are compared. The correlations between TPI values of amino acids over all sequences of an organism are estimated. All computing was done in the DARWIN system.
Contents

1 Introduction ........................................... 1

2 Computing the TPI .................................. 2
  2.1 Method ......................................... 2
    2.1.1 Recursive Computation ...................... 2
    2.1.2 Monte Carlo Simulation .................... 2
  2.2 Results ........................................ 3

3 TPI of Multiple Organisms ......................... 4
  3.1 Method ........................................ 4
  3.2 Results ........................................ 4

4 Twin exclusion .................................... 7
  4.1 Method ........................................ 7
  4.2 Results ........................................ 7

5 Outliers ............................................ 9
  5.1 Method ........................................ 9
  5.2 Results ....................................... 9

6 Skewness .......................................... 12
  6.1 Method ....................................... 12
  6.2 Results ...................................... 15

7 Correlation ........................................ 19
  7.1 Method ....................................... 19
  7.2 Results ...................................... 19

8 Conclusions ....................................... 22

9 Acknowledgements .................................. 24

10 References ........................................ 25
1 Introduction

In messenger RNA (mRNA) sequences coding for a particular protein, effects of autocorrelation emerge. Autocorrelation means that the occurrence of a given codon in the sequence depends on the previous codon coding for the same amino acid; they are correlated. It has been hypothesized that this autocorrelation influences the re-usage of tRNA molecules.

Imagine you have two sorts of balls, \( n \) red ones and \( n \) green ones, representing two sorts of tRNAs transferring the same amino acid. There are \( n + n \) boxes in a row and you have to place one ball in each box. A pair is defined as two adjacent balls. You’re interested in the number of changes, where a change is a pair of differently colored balls. It is a combinatorial problem to check all possible arrangements and to determine the expected number of changes. If you put the red balls in the first \( n \) boxes and the green balls in the remaining ones, there will only be one single change, where the last red and the first green ball get in touch. This is an example for a highly autocorrelated sequence, since all pairs (except one) have the same color and the occurrence of a ball “guarantees” that its neighbours will have the same color with high certainty.

Alternating red and green balls leads to the maximal number of changes. This is an example for a sequence with negative autocorrelation.

The TPI counts the number of changes in a sequence and compares the outcome to the expected number of changes. If the value is smaller than the expected number, the sequence is positively autocorrelated, otherwise negatively. The TPI lies in the interval \([-1, 1]\), where \(+1\) indicates high (positive) autocorrelation and \(-1\) means negative autocorrelation.

There is a biological background for this index: To increase the translation efficiency in the ribosomes, DNA sequences may have the tendency to reuse the same codon again because the coding tRNA will be around for some time and can be recycled to avoid waiting until another tRNA finds its way to the ribosom. By reusing the same codon, the trend to reuse the same tRNAs increases, the number of changes decreases and the TPI increases.

This paper is divided into several sections. Every section concentrates on one particular aspect of the TPI problem. Section 2 discusses the two approaches to calculate the TPI. In Section 3 we compute the TPI for multiple organisms and interpret the results. Section 4 discusses twin exclusion. In Section 5 we analyse outliers. Section 6 deals with the skewness of the probability distribution of the number of changes for a particular amino acid. Finally, Section 8 concludes the previous results and insights.
2 Computing the TPI

There are two fundamentally different ways to compute the TPI: An exact recursive method and a Monte Carlo approach. We will explain both methods and highlight their advantages and weaknesses.

2.1 Method

2.1.1 Recursive Computation

In DARWIN [1], the exact recursive method is already implemented in the function `ComputeTPI`. A DNA sequence is passed as an argument and the TPI will be returned. The optimized kernel function `TPIDistr` acts as the core component of the procedure. It computes the probability distribution of the number of changes for a given set of tRNAs coding for an amino acid. All these distributions are convoluted into a single distribution. The actual number of changes determines the cumulative probability which is spread over the interval $[-1, 1]$. If there are more changes than expected, the TPI is negative, otherwise it is positive. `TPIDistr` checks all possible tRNA arrangements and counts the number of changes. Therefore, the generated probability distribution is exact.

Until now, all computing was done in the yeast database, where the maximum number of tRNAs transferring a amino acid is four. Considering this constraint, the kernel function was designed and optimized for at most four arguments. Unfortunately, organisms like *E.coli* or *S.pombe* contain amino acids that are transferred by five tRNAs. The calculation will fail for these data sets. It is beyond the scope of this project to rewrite the kernel function to adapt it to the new circumstances. Instead, we implemented a Monte Carlo approach.

2.1.2 Monte Carlo Simulation

According to the Monte Carlo tradition, we generate random permutations of the given tRNA sequence. For every permutation, we count the number of changes for all amino acids and add them up to get the total number of changes. The values are analysed and the cumulative probability is computed. Again, the cumulative probability is spread over the interval $[-1, 1]$. The number of generated permutations is an important parameter. More permutations mean more operations but higher accuracy, fewer permutations lead to faster termination of the computation but lower accuracy. There is always a tradeoff between computing time and precision.
<table>
<thead>
<tr>
<th>Permutations</th>
<th>Average error</th>
<th>Average computing time (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.0166735</td>
<td>3.13029</td>
</tr>
<tr>
<td>2000</td>
<td>0.0122644</td>
<td>6.22666</td>
</tr>
<tr>
<td>3000</td>
<td>0.0103834</td>
<td>9.35892</td>
</tr>
<tr>
<td>4000</td>
<td>0.0092495</td>
<td>12.42070</td>
</tr>
<tr>
<td>5000</td>
<td>0.0086503</td>
<td>15.58180</td>
</tr>
</tbody>
</table>

Table 1: Departure from the exact value depending on different number of permutations

Table 1 shows the average error and the average execution time per entry as a function of the number of permutations. Throughout the paper, a default value of **1000 permutations** was chosen. The error is acceptably small and the computing time lies within reasonable boundaries. With a constant number of permutations, longer sequences tend to generate bigger errors. This dependency is neglectable since we restrict the maximal length of a sequence to **2500 codons**. Longer sequences are excluded for reasons of excessive computing time. The fraction of ignored entries is less than 1% for all organisms.

2.2 Results

The recursive method is exact and fast, but it is based on a model that is hardly adaptable to new circumstances. Furthermore, there is the drawback that TPIDistr only allows the computation of four tRNAs. The Monte Carlo approach is reliable and robust but noisy, computationally intensive and approximate. Its strength lies in the ability to incorporate easily additional constraints. We will appreciate this capability while analysing twin exclusion (Section 4).

If possible, we use the exact method because it saves time and increases the accuracy of the results.
3 TPI of Multiple Organisms

We compute the TPI for the multicellular organisms *H. sapiens* and *C. elegans*, the yeasts *S. pombe* and *S. cerevisiae* and the bacteria *B. subtilis* and *E. coli*.

3.1 Method

We used the recursive method to compute the TPI for *H. sapiens* and *S. cerevisiae*. For the other organisms, we applied the Monte Carlo simulation. The TPI is calculated separately for every entry in the database. The individual values are added up and averaged. The resulting TPI is a measure for the autocorrelation in the entire organism. We plot the distribution of the cumulated TPIs over the interval [−1, 1] in a histogram to visualize the tendencies already inherently present in the computed results.

3.2 Results

<table>
<thead>
<tr>
<th>Organism</th>
<th>TPI Monte Carlo</th>
<th>TPI exact recursive</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>H. sapiens</em></td>
<td>0.308249</td>
<td>0.309598</td>
</tr>
<tr>
<td><em>S. cerevisiae</em></td>
<td>0.12757</td>
<td>0.129301</td>
</tr>
<tr>
<td><em>S. pombe</em></td>
<td>0.0860732</td>
<td>—</td>
</tr>
<tr>
<td><em>C. elegans</em></td>
<td>0.214542</td>
<td>—</td>
</tr>
<tr>
<td><em>B. subtilis</em></td>
<td>0.0408884</td>
<td>—</td>
</tr>
<tr>
<td><em>E. coli</em></td>
<td>0.190126</td>
<td>—</td>
</tr>
</tbody>
</table>

Table 2: TPI for several organisms

A comparison between exact and approximated TPI shows that the Monte Carlo error is marginal (Table 2). The TPI is positive for all organisms so they are all positively autocorrelated. This observation supports the biological hypothesis that high autocorrelation leads to higher translation efficiency. If there would be no signs of autocorrelation, this would indicate that nature doesn’t care about tRNA reuse. In a darwinian world like ours where only the fittest life-forms survive, the reuse of codons seems to pay off.

The most complex organism (by means of physical diversity) *H. sapiens* shows the highest TPI value. We conclude that high complexity of an organism downcasts to a high TPI. Regarding the fact that *C. elegans* also has a high
TPI supports this hypothesis. The measures of the two yeasts are more positive than the TPI of the “simple” bacteria \textit{B. subtilis} that has the smallest value. 

But our assumption doesn’t hold for \textit{E.coli} which has a higher TPI than the evolutionary higher developed yeasts \textit{S.cerevisiae} and \textit{S.pombe}.

Therefore, structural complexity is not the criteria to explain these results. It is interesting to see that the two yeasts have similar TPIs whereas the two bacteria differ seriously.

One may think that the number of entries in the databases influence the computation, but \textit{E.coli} has the least number of entries but not the smallest TPI.

There is no obvious criteria that separates highly from poorly autocorrelated organisms. The histograms in Figure 1-3 clarify the already observed tendencies in Table 2. There are big shifts to the positive side in \textit{H.sapiens} and \textit{C.elegans} and a balanced distribution in \textit{B.subtilis}. For reasons of consistency, all graphics stem from Monte Carlo simulation.

Figure 1: Distributions of the cumulated TPIs for \textbf{H.sapiens} (left) and \textbf{C.elegans} (right)
Figure 2: Distributions of the cumulated TPIs for \textit{S.cerevisiae} (left) and \textit{S.pombe} (right)

Figure 3: Distributions of the cumulated TPIs for \textit{B.subtilis} (left) and \textit{E.coli} (right)
4 Twin exclusion

The tRNA molecules match the codons of the mRNA strand in the ribosomes and append the transported amino acid to the growing protein. Two consecutive codons that are bound by the same tRNA are called twins. There are biological reasons to exclude them from computation. The hope is to get a more accurate picture of autocorrelation in the sequences.

4.1 Method

The inclusion of twin exclusion enforces a different TPI computation. This is a situation where the easy adaption of the Monte Carlo method pays off. It is hardly possible to change completely the entire recursive model but it is simple to exclude all twins by modifying the Monte Carlo approach. Excluding twins means to go through the permuted sequences before counting the changes (say from left to right) and to break up the twins by deleting one partner (by convention the right hand sided; compare Table 3). All the other steps are identical.

<table>
<thead>
<tr>
<th>Original Sequence</th>
<th>Marking twins</th>
<th>Deleting partner to the right</th>
<th>“Clean” sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAABB</td>
<td>BAABB</td>
<td>BAB</td>
<td>BAB</td>
</tr>
<tr>
<td>AAABBBB</td>
<td>AAAABBBB</td>
<td>AABB</td>
<td>AABB</td>
</tr>
</tbody>
</table>

Table 3: Twin exclusion

Schraudolph and Gonnet propose in their paper [3] that it is not allowed to permute the sequences as a whole, but to permute the tRNAs within the occupied positions of the particular amino acid. Our results show that is makes no difference whether or not to follow this assessment. Only the relative changes within a set of tRNAs are counted and they are independent of the absolute positions.

4.2 Results

The twin exclusion was done for all organisms and the outcome highly corresponds to the observations made by Schraudolph and Gonnet in [3]: The differences in TPI computation with and without twins are negligible. Table 4 shows that there are no significant shifts.

By deleting the twins, the number of tRNAs and therefore the expected
<table>
<thead>
<tr>
<th>Organism</th>
<th>TPI with twins</th>
<th>TPI with twin exclusion</th>
<th>$\text{TPI}_{\text{excl}} - \text{TPI}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H.sapiens$</td>
<td>0.308264</td>
<td>0.30863</td>
<td>+0.000366</td>
</tr>
<tr>
<td>$S.cerevisiae$</td>
<td>0.126163</td>
<td>0.126128</td>
<td>−0.000035</td>
</tr>
<tr>
<td>$S.pombe$</td>
<td>0.0860831</td>
<td>0.0858865</td>
<td>−0.0001966</td>
</tr>
<tr>
<td>$C.elegans$</td>
<td>0.215029</td>
<td>0.215049</td>
<td>+0.00002</td>
</tr>
<tr>
<td>$B.subtilis$</td>
<td>0.0405878</td>
<td>0.0407011</td>
<td>+0.0001133</td>
</tr>
<tr>
<td>$E.coli$</td>
<td>0.189517</td>
<td>0.190321</td>
<td>+0.000804</td>
</tr>
</tbody>
</table>

Table 4: Consequences of twin exclusion

number of changes decrease. The actual number of changes in a given sequence is constant since the deleted tRNAs don’t affect the changes: They are identical to the remaining partner. Hence, the number of changes for a given sequence increases relatively to the expected number of changes and we assume that the TPI decreases in all organisms when applying twin exclusion. This hypothesis holds for four organisms but the differences are very small. The reason why the effect is not clearer lies in the facts that (1) the Monte Carlo simulation is approximate and (2) the number of twins is very small compared to the average length of the sequences (Table 5). There are

<table>
<thead>
<tr>
<th>Organism</th>
<th>Average number of twins</th>
<th>Average length (in codons)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H.sapiens$</td>
<td>21.08</td>
<td>763.77</td>
</tr>
<tr>
<td>$C.elegans$</td>
<td>16.97</td>
<td>634.93</td>
</tr>
<tr>
<td>$S.cerevisiae$</td>
<td>17.85</td>
<td>678.42</td>
</tr>
<tr>
<td>$S.pombe$</td>
<td>16.91</td>
<td>685.63</td>
</tr>
<tr>
<td>$B.subtilis$</td>
<td>10.13</td>
<td>435.79</td>
</tr>
<tr>
<td>$E.coli$</td>
<td>12.14</td>
<td>471.50</td>
</tr>
</tbody>
</table>

Table 5: Average number of twins and average sequence length

no shifts because the number of twins is too small to make any difference.
5 Outliers

Using the exact method, there are remarkable differences in computing time for entries with similar lengths (Table 6). We want to analyse these sequences and find the reasons for the differences.

<table>
<thead>
<tr>
<th>Organism</th>
<th>Entry</th>
<th>Length (in codons)</th>
<th>Computing time (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>S. cerevisiae</em></td>
<td>680</td>
<td>1609</td>
<td>1835.9</td>
</tr>
<tr>
<td></td>
<td>1044</td>
<td>1612</td>
<td>16.9</td>
</tr>
<tr>
<td><em>S. cerevisiae</em></td>
<td>1374</td>
<td>1802</td>
<td>2327.7</td>
</tr>
<tr>
<td></td>
<td>1500</td>
<td>1796</td>
<td>35.3</td>
</tr>
<tr>
<td><em>H. sapiens</em></td>
<td>10351</td>
<td>2442</td>
<td>1779.28</td>
</tr>
<tr>
<td></td>
<td>8251</td>
<td>2442</td>
<td>97.40</td>
</tr>
<tr>
<td><em>H. sapiens</em></td>
<td>10473</td>
<td>2108</td>
<td>1280.44</td>
</tr>
<tr>
<td></td>
<td>7398</td>
<td>2106</td>
<td>37.28</td>
</tr>
</tbody>
</table>

Table 6: Long running entries and sequences with similar lengths but short computing time.

5.1 Method

In the Monte Carlo approach, the computing time is proportional to the number of permutations and the length of the sequences. There is no dependency on other criterias. As a consequence, outliers only occur in recursive computation.

We know that the kernel function TPIDistr requires the main part of the computing time. Because TPIDistr operates on the tRNAs, we take into account two quantities to demystify the outliers: (1) The distribution of the number of tRNAs and (2) the number of changes for all amino acids.

5.2 Results

We compare the number of changes for the entries 680 and 1044.

There are two amino acids that drop-out in entry 680: Serine (16) and to a less degree Threonine (17). All the other values lie within a certain range.

The tRNA distribution shows that there is a huge number of tRNAs coding for Serine and Threonine. More tRNAs result in higher computing time since TPIDistr has to deal with deeper recursion. This is the reason for the overwhelming execution time.
Figure 4: **Number of changes** for entries 680 and 1044

Figure 5: **tRNA distribution** for entries 680 and 1044

The number of changes serves as an indicator for interesting tRNA arrangements since increasing number of tRNAs go in hand with increasing number of changes.

We analysed a total of four outliers (Table 6) and these characteristics arised
in all of them.

The tRNA distribution also shows causes for short computing time. Fast sequences often suffer from putting the tRNAs on amino acids that are translated by only one tRNA (compare amino acids 3 and 4 of entry 1044). Regarding the number of changes, these tRNAs are wasted since no changes occur and no TPDistr evaluation takes place. Another reason for fast convergence are unbalanced sets of tRNAs where the number of changes is limited to twice the sum of the smallest values (compare amino acid 10 of entry 1044).

Suppose you have a constant number of entities (tRNAs) that you distribute among a group of tRNAs coding for an amino acid. How does the distribution of the entities influences the computing time? Figure 6 demonstrates

![Figure 6: Spreading 400 elements over a set of four tRNAs (computing time in brackets)](image)

that extremely balanced and extremely unbalanced tRNA distributions lead to short computing time. The highest values achieve those distributions that form a “stair” and have (almost) constant differences between the individual tRNAs.
6 Skewness

The probability distribution of the number of changes computed by the kernel function \textsc{TPIDistr} is not symmetric but skewed and therefore doesn’t fit a Gaussian distribution very well (Figure 8). For reasons of simplicity, the number of changes are assumed to follow a Gaussian distribution. To determine the influence of approximating the probability distribution by a Gaussian distribution, we tested several alternative distributions.

![Skewed probability distribution](image)

Figure 7: Skewed probability distribution: \textsc{TPIDistr}(15,19)

6.1 Method

Fernandez and Steel [4] define the following class of skewed distributions, indexed by a scalar $\gamma \in (0, \infty)$:

$$p(x \mid \gamma) = \frac{2}{\gamma + \frac{1}{\gamma}} \left\{ f \left( \frac{x}{\gamma} \right) I_{[0,\infty)}(x) + f(\gamma x) I_{(-\infty,0)}(x) \right\}$$

where $f$ is a unimodal probability distribution that is symmetric around 0 and $I$ is an index that separates the positive and the negative range.

The basic idea is to introduce a scalar factor $\gamma$ that breaks the symmetry of the distribution. $p(x \mid \gamma)$ retains its mode at 0 but loses the symmetry whenever $\gamma \neq 1$.

An important term in our considerations is

$$\frac{P(x \geq 0 \mid \gamma)}{P(x < 0 \mid \gamma)} = \gamma^2$$
Figure 8: Skewed standard Gaussian distribution

that allows the calculation of the parameter $\gamma$ by just cumulating the probabilities on both sides of the mode, dividing them and taking the square root. TPDistr provides the necessary probabilities to proceed. All we have to do is to find the mode of the distribution and then to cumulate the probabilities to its left and to its right. The mode is the maximal probability of the distribution. $\gamma$ can then be computed using the formula above.

\[
P(x \geq 0 \mid \gamma) = \sum_{i=\text{mode}+1}^{N} \Pr(Changes = i) + \frac{1}{2} \Pr(Changes = \text{mode})
\]

\[
P(x < 0 \mid \gamma) = \sum_{i=0}^{\text{mode}-1} \Pr(Changes = i) + \frac{1}{2} \Pr(Changes = \text{mode})
\]

where $N$ is the maximal number of changes and $\Pr(\ldots)$ are the probabilities according to TPDistr.

If there is no clear mode but two probabilities that are equally likely (with respect to a maximal difference $\epsilon$), the formulas change to

\[
P(x \geq 0 \mid \gamma) = \sum_{i=\text{mode}+1}^{N} \Pr(Changes = i)
\]

\[
P(x < 0 \mid \gamma) = \sum_{i=0}^{\text{mode}} \Pr(Changes = i)
\]

where $\text{mode}$ and $\text{mode} + 1$ are the two modes.

It has to be said that there are exceptions where the computation of $\gamma$ will
fail because there are kinks in the probability distribution that make the location of the mode fail. These artefacts only arise in distributions with a small number of changes.

The following considerations are based on the analysis of three examples. All computing is done upon their probability distributions. The results are just observations and the chosen examples may be not representative. The

Figure 9: Example 1: TPIDistr(40, 50, 60, 70), 4 arguments, stepwise

Figure 10: Example 2: TPIDistr(50, 50, 52), 3 arguments, balanced

Figure 11: Example 3: TPIDistr(10, 20), 2 arguments, unbalanced

samples have 2, 3 and 4 arguments and the arguments are unbalanced, balanced and stepwise (Figure 9-11). The aim is to cover a wide area of possible

14
phenomena and therefore to increase the reliability of the results. We will try to fit several common distributions to the probability distributions of the examples. The errors are measured by a least squares approach.

$$\text{minimize} \sum_{i=0}^{N} (\text{cum}_i - \Phi(i))^2$$

where

$$\text{cum}_i = \sum_{j=0}^{i} \text{Pr}(\text{Changes} = j)$$

$N$ is the maximal number of changes and $\Phi$ is a z-transformed cumulative reference distribution. If the reference distribution contains additional parameters that have to be optimized, a maximum likelihood estimation is used.

$$\text{minimize over } a_1..a_M \sum_{i=0}^{N} (\text{cum}_i - \Phi(i; a_1..a_M))^2$$

### 6.2 Results

We fit four distributions to the probability distributions of the examples:

1. Gaussian distribution
2. Skewed Gaussian distribution
3. Skewed Student’s t distribution
4. Binomial distribution

<table>
<thead>
<tr>
<th>TPIDistr arguments</th>
<th>$\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(40,50,60,70)</td>
<td>0.9762</td>
</tr>
<tr>
<td>(50,50,52)</td>
<td>0.9825</td>
</tr>
<tr>
<td>(10,20)</td>
<td>0.8128</td>
</tr>
</tbody>
</table>

Table 7: $\gamma$-factor for all examples

It holds that $\gamma < 1$ for all examples, which means that the centre of mass lies to the left of the mode (Table 7). Comparing the Gaussian distribution to its skewed variant shows that the introduction of skewness results in a reduction of the error (Table 8).
The probability distributions of the examples are characterized by heavier tails converse to those of the Gaussian distribution. It makes sense to take this fact into account. The Student’s t distribution allows an adaptation of the tail but there is an additional parameter $\nu$ to deal with. For $\nu \to \infty$, the t distribution converges to the Gaussian distribution. The maximum likelihood estimation is used to determine the parameter $\nu$. The skewed Student’s t distribution improves the precision, but the improvement is only minor (Table 9) and is most likely not worth the additional complexity of estimating the optimal $\nu$.

The main drawback in all previous considerations is the fact that all distributions are continuous whereas the approximated probability distribution is discrete. There are only whole-numbered changes. To meet this constraint, we make use of the binomial distribution which is symmetric but discrete.

$$P(n \mid N) = \binom{N}{n} p^n (1 - p)^{N-n}$$

All parameters can easily be computed: $N$ is the maximal number of changes, the probability $p$ for a change can be calculated by dividing the expected number of changes by $N$:

$$E[Changes] = \sum_{i=1}^{N} Pr(Changes = i) \cdot i$$

$$p = \frac{E[Changes]}{N}$$

Table 8: Errors of the approximation

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>0.00950569</td>
<td>0.01095585</td>
<td>0.02786155</td>
</tr>
<tr>
<td>Skewed Gaussian</td>
<td>0.00250289</td>
<td>0.00514557</td>
<td>0.00791045</td>
</tr>
</tbody>
</table>

Table 9: Errors of the approximation
For the least squares approach, the formula changes to

\[
\text{minimize } \sum_{i=0}^{N} (\text{cum}_i - \text{binom}_i)^2
\]

where

\[
\text{binom}_i = \sum_{j=0}^{i} P(j \mid N)
\]

The verification yields a much better approximation than the other distributions. The reason for that lies in the observation that the skewness of the probability distributions is only small which is reflected by \(\gamma\) values close to 1 in Table 7. The gain of using a discrete distribution is bigger than the symmetry loss. The conversion from continuous to discrete distributions has more impact on the result than the transition from symmetric to asymmetric distributions.

The skewed Gaussian distribution is always better than the simple Gaussian distribution. The skewed Student’s t distributions is marginally better than the skewed Gaussian distribution. The cost for the small improvement is an additional parameter. It isn’t worth the extra effort. The binominal distribution beats all the other distributions in all examples by far (Table 10).
<table>
<thead>
<tr>
<th>Distribution</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example 1</td>
</tr>
<tr>
<td>Gaussian</td>
<td>0.00950569</td>
</tr>
<tr>
<td>Skewed Gaussian</td>
<td>0.00250289</td>
</tr>
<tr>
<td>Skewed Student</td>
<td>0.00250208</td>
</tr>
<tr>
<td>Binomial</td>
<td>0.00016102</td>
</tr>
</tbody>
</table>

Table 10: Errors of the approximation
7 Correlation

Regarding the fact that the effects of autocorrelation emerge all over the place in the analysis of DNA sequences, the question arises whether there are signs of correlation between the TPIs of different amino acids across sequence boundaries. Whereas we investigated the autocorrelation in sets of tRNAs coding for an amino acid, we now study the correlation of TPI values between amino acids over entire databases.

7.1 Method

First we compute the TPI for all amino acids in a sequence. The values of all entries will be related to each other by a correlation matrix which then reveals the hidden correlation.

7.2 Results

The resulting correlation matrices are humbling: There is no serious correlation detectable between the TPIs of the amino acids. They seem to coexist without relating to each other, as the example of *H. sapiens* shows.

\[
\begin{pmatrix}
1.00 & 0.03 & 0.00 & 0.03 & 0.02 & 0.03 & 0.01 & 0.02 & 0.00 & 0.03 & 0.02 & 0.00 \\
0.03 & 1.00 & 0.02 & 0.01 & 0.03 & 0.03 & 0.01 & 0.02 & 0.03 & 0.04 & 0.04 & 0.00 \\
0.00 & 0.02 & 1.00 & 0.04 & 0.03 & -0.00 & 0.07 & 0.02 & 0.03 & -0.00 & 0.01 & 0.01 \\
0.03 & 0.01 & 0.04 & 1.00 & 0.04 & 0.01 & 0.07 & 0.01 & 0.04 & 0.02 & 0.01 & 0.01 \\
0.02 & 0.03 & 0.03 & 0.04 & 1.00 & 0.02 & -0.02 & 0.01 & 0.02 & 0.01 & 0.03 & 0.00 \\
0.03 & 0.03 & -0.00 & 0.01 & 0.02 & 1.00 & -0.01 & 0.00 & 0.03 & 0.03 & 0.01 & -0.00 \\
0.01 & 0.01 & 0.07 & 0.07 & -0.02 & -0.01 & 1.00 & 0.01 & 0.04 & -0.01 & 0.01 & -0.00 \\
0.02 & 0.02 & 0.02 & 0.01 & 0.01 & 0.00 & 0.01 & 1.00 & 0.02 & 0.01 & 0.01 & 0.02 \\
0.00 & 0.03 & 0.03 & 0.04 & 0.02 & 0.03 & 0.04 & 0.02 & 1.00 & 0.01 & 0.01 & 0.00 \\
0.03 & 0.04 & -0.00 & 0.02 & 0.01 & 0.03 & -0.01 & 0.01 & 0.01 & 1.00 & -0.00 & 0.01 \\
0.02 & 0.04 & 0.01 & 0.01 & 0.03 & 0.01 & 0.01 & 0.01 & -0.00 & 1.00 & 0.01 & 0.01 \\
0.00 & 0.00 & 0.01 & 0.01 & 0.00 & -0.00 & -0.00 & 0.02 & 0.00 & 0.01 & 0.01 & 1.00 \\
0.02 & 0.02 & 0.02 & 0.03 & 0.04 & 0.01 & 0.02 & 0.02 & -0.01 & 0.02 & 0.01 & 0.01 \\
\end{pmatrix}
\]

There are only 13 amino acids as all those are excluded that are transferred by one tRNA. Computing the TPI for these amino acids is superfluous since there are no changes and hence no autocorrelation.

All other organisms show the same results as above.

The reason for this outcome could be caused by effects of smearing. Every DNA sequence in a database consists of functional and non-functional parts. We expect that the TPIs differ according to the stretch they are measuring. By mixing the different parts, the tendency to equalize the TPIs of the amino acids arise. Intervals with high TPI are combined with stretches of low TPI. This effect could be responsible for the very low correlation.

For membrane proteins, the tool **TMHMM** [5] allows the breakdown of entire sequences into their functional and non-functional parts. The non-functional stretches are additionally divided into parts that lie inside or outside the organelle.
A set of cell wall sequences builds the data on which the computation is based. The results show that the coding parts are 22-29 DNA bases long. The small number of bases doesn’t allow reliable calculations. Concatenation of coding parts of different sequences leads to wrong results. There are several entries which have more than one functional part. Out of the 102 sequences, there are 7 that have more than 5 of them. In these sequences, the functional stretches and the non-functional parts in-

![TMHMM posterior probabilities for FKS1](image)

Figure 13: Example for a sequence with multiple functional parts: FKS1 inside and outside the organelle are concatenated and the TPIs are computed separately. It holds that the TPIs of functional and non-functional parts differ but the

<table>
<thead>
<tr>
<th>Sequence ID</th>
<th>TPI functional stretch</th>
<th>TPI non-functional inside organelle</th>
<th>TPI non-functional outside organelle</th>
</tr>
</thead>
<tbody>
<tr>
<td>FKS1</td>
<td>-0.7251</td>
<td>0.8732</td>
<td>-0.5304</td>
</tr>
<tr>
<td>CHS1</td>
<td>0.0404</td>
<td>-0.5931</td>
<td>-0.3141</td>
</tr>
<tr>
<td>CHS2</td>
<td>-0.5832</td>
<td>0.7400</td>
<td>-0.5624</td>
</tr>
<tr>
<td>CHS3</td>
<td>0.1837</td>
<td>0.8104</td>
<td>-0.0508</td>
</tr>
<tr>
<td>ECM3</td>
<td>-0.9493</td>
<td>-0.5981</td>
<td>0.3094</td>
</tr>
<tr>
<td>ECM27</td>
<td>0.8787</td>
<td>0.5430</td>
<td>-0.4751</td>
</tr>
<tr>
<td>ECM39</td>
<td>0.1787</td>
<td>0.1256</td>
<td>-0.9066</td>
</tr>
</tbody>
</table>

Table 11: Functional and non-functional TPIs for several proteins underlying patterns are not clear.

\[
\text{Corr} = \begin{pmatrix}
corr_{f,f} & corr_{f,in} & corr_{f,out} \\
corr_{in,f} & corr_{in,in} & corr_{in,out} \\
corr_{out,f} & corr_{out,in} & corr_{out,out}
\end{pmatrix} = \begin{pmatrix}
1.0000 & 0.1589 & -0.3537 \\
0.1589 & 1.0000 & -0.3989 \\
-0.3537 & -0.3989 & 1.0000
\end{pmatrix}
\]
where $f$ are the functional parts, $in$ are the non-functional parts inside the organelle and $out$ are the non-functional parts outside the organelle. The correlation matrix shows that there is positive correlation between the TPIs of the functional stretches and the TPIs of the non-functional inside parts. Furthermore, there is negative correlation between the TPIs of the functional stretches and the TPIs of the non-functional outside parts. But there are no obvious structures or regularities.
8 Conclusions

We have seen the functionality of the two TPI evaluation approaches Recursive Computation and Monte Carlo Simulation. The former is exact and fast, the latter is expensive but easily adaptable to new circumstances. For pure TPI computation, the exact method defeats Monte Carlo by far but as soon as changes take place, Monte Carlo is of invaluable importance. It would be a helpful improvement for future works to extend the kernel function TPIDistr to more than four arguments.

The TPI computation for multiple organisms points out serious differences in the measures. There is no easy separation rule for highly and poorly autocorrelated organisms. A clear classification and characterization seems to be a non-trivial task. Further studies with more data sets are necessary to reveal the hidden patterns combining groups of organisms and characteristic TPI values.

Taking into account the exclusion of twins, the observations made by Schraudolph and Gonnet in [3] indicating that there is no remarkable departure from the evaluation with twins are also correct for the other organisms. The number of twins is too small to cause a shift in the TPI. We think these results prove that twin exclusion doesn’t matter for TPI computation.

We characterized outliers and found reasons for their weird behavior. The high number of tRNAs of a single amino acid leads to a deep recursion within the kernel function TPIDistr and is responsible for the enormous computing time. We have seen that under the assumption of constant tRNAs, not balanced nor unbalanced, but constantly staged groups of tRNAs lead to long execution time.

The probability distribution of changes suffer from skewness.

On the basis of three examples, we tested several distributions to approximate the probability distribution. We introduced a class of skewed distributions that leads to a remarkable improvement. The replacement of the Gaussian distribution by a Student’s t distribution brings a further refinement. Because the computation of the additional parameter $\nu$ that controls the tail is expensive and the gain is small, we recommend to not use the Student’s t distribution. The final step from a continuous asymmetric to a discrete symmetric distribution and the analysis of the binomial distribution yields another advancement. The calculation of the two parameters $p$ and $N$ is trivial and the results outperform the other approximations, no matter how distinct the skewness is.

In the last chapter we analysed the correlation between TPIs of amino acids across sequence boundaries. There are no signs of correlation. The breakdown of membrane proteins into their functional and non-functional parts...
revealed different TPIs but no clear patterns. We think it would be worth going ahead in this direction.

This Semesterarbeit touched a wide area in the field of TPI computation. The analysis didn’t explore the subject in-depth, but our observations shed light on many aspects of the problem. Some of the paths we took ended in no man’s land, others confirmed our considerations.
9 Acknowledgements

I would like to thank Peter von Rohr for his support, his critical comments and his help that lead me out of the darkness into the light a hundred times. I know it wasn’t always easy to work with me. His quietness and gentleness are amazing.
My thanks go to him.
10 References

References


