Measures of Codon Bias in Yeast, the tRNA Pairing Index and Possible DNA Repair Mechanisms*

Markus T. Friberg[1], Pedro Gonnet[1], Yves Barral[2], Nicol N. Schraudolph[3 4], and Gaston H. Gonnet[1]

[1] Institute of Computational Science, ETH Zurich, 8092 Zurich, Switzerland
[2] Institute of Biochemistry, Department of Biology, ETH Zurich, Switzerland
[3] Statistical Machine Learning, National ICT Australia, Canberra ACT 2601, Australia
[4] RSISE, Australian National University, Canberra ACT 0200, Australia

Abstract. Protein translation is a rapid and accurate process, which has been optimized by evolution. Recently, it has been shown that tRNA reusage influences translation speed. We present the tRNA Pairing Index (TPI), a novel index to measure the degree of tRNA reusage in any gene. We describe two variants of the index, how to combine various such indices to a single one and an efficient algorithm for their computation. A statistical analysis of gene expression groups indicate that cell cycle genes have high TPI. This result is independent of other biases like GC content and codon bias. Furthermore, we find an additional unexpected codon bias that seems related to a context sensitive DNA repair.

1 Introduction

Protein translation is a rapid and accurate process, despite the need to discriminate between many possible incoming and competing tRNAs. One can assume that the process has been optimized by evolution. It has been shown that tRNA availability is both a limiting step and a regulatory parameter during translation [1, 2]. Recently, through an experiment with synthesized GFP genes, it was shown that tRNA reusage (codon order) influences translation speed in yeast [3]. Here we describe the tRNA Pairing Index (TPI), an index that measures the degree of tRNA reusage in any gene.

By a statistical analysis of the TPI and gene expression, we show that genes that change their expression level rapidly (and thus require the most rapid translation) have a (statistically significant) higher TPI. Specifically, genes involved in cell cycle and DNA damage have a high TPI. These genes are regulated in the most dynamic manner, i.e. they are most rapidly turned on and off in response to intra- or extra-cellular activities.

The TPI distribution over all yeast coding sequences is biased towards positive values, indicating that there is a general tendency of tRNA reusage in the yeast genome.

Codon bias has been extensively studied previously [4–10]. However, to the best of our knowledge, the problem of measuring tRNA reusage in a gene has not been addressed before. The general analysis of codon autocorrelation suffers from the bias that may be induced by different base frequencies in different parts of the genome. It is

* in: P. Bucher and B. Moret (eds), Proceedings of the 6th Workshop on Algorithms in Bioinformatics (WABI), vol. 4175 of Lecture Notes in Bioinformatics, Springer Verlag, Berlin 2006.

known that some parts of the genome are GC-rich while other parts are GC-poor. Such long-stretched biases induce an autocorrelation in the codons, which could be significant. Our first version of the TPI can measure autocorrelation without being affected by this kind of bias.

2 Methods

The TPI is an index which is computed for each protein and measures the autocorrelation (positive or negative) of its codons. Depending on how the background distribution is chosen, it is possible to make TPI completely independent of the frequencies of the amino acids, tRNAs, codons or bases, so that it will not suffer from any of the common sources of bias.

We measure the autocorrelation independently of everything else by analyzing the usage of tRNA in each amino acid of a protein as a combinatorial problem on symbols. For example, suppose that we are considering an amino acid which occurs 7 times in the protein in question and can be translated by two different tRNAs, A and B (e.g. 3 A's and 4 B's). We will extract the tRNAs from our sequence and represent them as a sequence of 7 symbols, e.g. AABABBB.

Highly autocorrelated cases are AAABBBB and BBBBAAA. A highly negatively autocorrelated case is BABABAB. This autocorrelation can be quantified by the number of identical pairs in the sequence or, conversely, by the number of changes C as we read from left to right. Notice that for a sequence of length n, the number of identical pairs plus the number of changes is $n - 1$. The mathematics is completely analogous for the number of pairs or number of changes. We call these breaks in the sequences changes, with the thought that if a tRNA molecule is doing the translation for one particular amino acid, when these breaks happen, this tRNA will have to be changed for another molecule. The first two examples have 1 change each, the last example has 6 changes. The TPI measures how high the actual number of pairs are, or how low C is, compared to all possible permutations of the sequence of tRNAs.

We present two different background distributions: one (TPI_1) based on codon frequencies given by the actual gene/genome under study, i.e. all possible orders considered equally likely (2.1) and another one (TPI_2) based on variable codon frequencies extracted from the entire genome (2.3).

2.1 TPI_1: Constant Codon Frequencies

Computation of the Probability of the Number of Changes. We will now describe the function to compute the probability and cumulative distribution of a given number of changes x.

It is easy to observe that the probability of the number of changes $C(x, n_1, n_2, ..., n_k)$ does not depend on what the symbols are, but rather on how many symbols there are of each kind $(n_1, n_2, ..., n_k)$. C is a (symmetric) function of the number of each kind of different symbols. It is difficult to write a recursion based on C, so instead we will base its computation on another function, called C_r, which does the recursive part of the computation. $C_r(x, n_1, n_2, ..., n_k)$ assumes that we are not at the beginning of the

sequence, but rather that the last symbol observed is known (Fig. 1). To identify this known symbol (all symbols are otherwise equivalent), we will make it the first of the arguments. Our function C_r assumes that it is called with a symbol of the first class preceding the rest of the symbols (Fig. 1). We explain C_r for $k = 2$ symbols in detail.

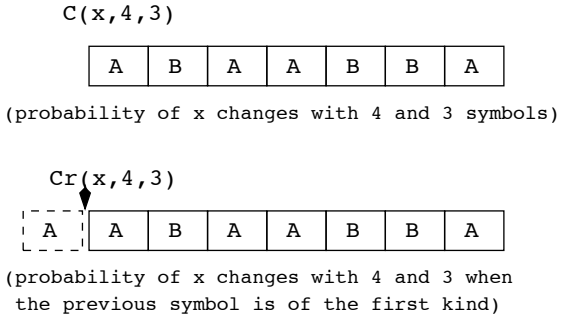

Fig. 1. C and C_r

$$C_r(x, n_1, n_2) = \begin{cases} 0 & \text{if } n_1 < 0 \text{ or } n_2 < 0 \text{ or } x < 0 \text{ or } x > n_1 + n_2 \\ 1 & \text{if } n_1 = n_2 = 0 \ (x \text{ must be } 0) \\ \frac{1}{n_1+n_2}(n_1 Cr(x, n_1 - 1, n_2) + n_2 Cr(x - 1, n_2 - 1, n_1)) \end{cases} \quad (1)$$

The first symbol is either from the class of n_1 (no change) or from the class of n_2, in which case the preceding symbol now is of the second class and we invert the arguments: $Cr(x - 1, n_2 - 1, n1)$.

The extension of this function to higher k is simple. Supplementary material (http://www.biorecipes.com/TPI/appendix/Cr.M) shows a production quality version of this procedure which takes into account more refined border conditions. $C(x, n_1, n_2)$ can be expressed in two forms in terms of C_r. First, if we allow an arbitrary number of symbols we use

$$C(x, n_1, n_2) = C_r(x + 1, [0, n_1, n_2]) \quad (2)$$

i.e. we create an artificial first symbol (of which we have 0 left) and allow for one more change. Else we can expand based on the first symbol:

$$C(x, n_1, n_2) = \frac{n_1}{n_1 + n_2} C_r(x, n_1 - 1, n_2) + \frac{n_2}{n_1 + n_2} C_r(x, n_2 - 1, n_1) \quad (3)$$

The code for C_r as written above, is exponential. We can use dynamic programming, or we could use something equivalent to *option remember* in Maple [11] to make it polynomial in the product of the n_i.

To estimate how rare a given number of changes is, we need to compute its cumulative distribution. Since the distribution is over the integers, we will take the cumulative distribution which adds one half of the probability at the point.

$$C_{cum}(x, n_1, n_2, ..., n_k) = \sum_{i=0}^{x-1} C(i, n_1, n_2, ..., n_k) + \frac{1}{2} C(x, n_1, n_2, ..., n_k) \quad (4)$$

Our TPI is $1 - 2C_{cum}$, which is more intuitive to use than C_{cum}.

Expected Values and Moments of the Number of Changes. The expected value of the number of changes can be expressed in terms of the symmetric functions S_1 and S_2 on the arguments:

$$\mu_1' = \sum_{i=1}^{\infty} i \times C(i, n_1, n_2, ..., n_k) = \frac{S_1^2 - S_2}{S_1} \quad \text{where} \quad S_i = \sum_{j=1}^{k} n_j^i \quad (5)$$

The derivation of this formula is not trivial in its general form (for an arbitrary k). However if we observe that all the probabilities are sums of binomial coefficients, then we can conclude that the result (expected value or higher moments) must be a polynomial expression divided an appropriate descending factorial. Since all the moments are symmetric in all the arguments, the moments must be functions of the symmetric polynomials derived from the n_i. Hence by symbolic interpolation we can determine all the moments in a much easier (and safer) way. Of interest are the expected value and the variance. This is because we will attempt a normal approximation to the distribution.

$$\mu_2 = \sum_{i=1}^{\infty} (i - \mu_1')^2 C(i, n_1, n_2, ..., n_k) = \frac{S_1 S_2 - S_1^3 - 2S_1 S_3 + S_2 S_1^2 + S_2^2}{S_1^2 (S_1 - 1)} \quad (6)$$

Unfortunately, despite the simplicity of the formulas resulting from this approach, they do not resolve our problem completely. The normal approximation gives a good approximation of the cumulative distribution around the average (for large values of S_1) and very good approximations when $\min(n_i)$ is high. However, it gives poor approximations at the tails when some of the n_i are small, which is an important case.

Computing the Distribution of C in Practice. The recursion in C_r, although simple, swaps its arguments, which makes it almost impossible to handle with the standard techniques. Even dynamic programming becomes very difficult to express. In this section we find a mechanism to rewrite the recursion in a way that the argument order is maintained.

Since the function is totally symmetric in its arguments (and C_r is totally symmetric in its arguments but the first) we can sort the arguments in increasing order guaranteeing a time of $O(n_1 n_2 .... n_k)$. This makes the recursion marginally acceptable for real problems (for yeast $k \leq 4$ and for most other genomes $k \leq 5$). This ordering is partly ruined by the swapping of arguments in the recursion (1). Each recursive call to C_r uses a different argument as second argument.

To resolve this problem we find recursions which (while maybe more complicated) do not jumble the arguments. We can illustrate this by doing the transformation on the simplest recursion, $k = 2$. For further simplicity, we will use the auxiliary function $H(x, n_1, n_2) = C_r(x, n_1, n_2)\binom{n_1+n_2}{n_1}$. As expected, the recursion on $H(x, n_1, n_2)$ is significantly simpler.

$$H(x, n_1, n_2) = H(x, n_1 - 1, n_2) + H(x - 1, n_2 - 1, n_1) \tag{7}$$

We now apply this formula to the shifted arguments

$$-H(x, n_1, n_2 - 1) = -H(x, n_1 - 1, n_2 - 1) - H(x - 1, n_2 - 2, n_1) \tag{8}$$

$$H(x - 1, n_2 - 1, n_1) = H(x - 1, n_2 - 2, n_1) + H(x - 2, n_1 - 1, n_2 - 1) \tag{9}$$

Adding these three equations results in

$$\begin{aligned} H(x, n_1, n_2) = {} & H(x, n_1 - 1, n_2) + H(x, n_1, n_2 - 1) - H(x, n_1 - 1, n_2 - 1) + \\ & H(x - 2, n_1 - 1, n_2 - 1) \end{aligned} \tag{10}$$

Notice that we have managed to obtain a recursion for which all the arguments (n_1, n_2) are in the same order. The new recursion with four terms instead of two is a bit more complicated, but this is an insignificant cost when we observe that in this form it is easy to write a recursive program to compute it. The computation can be done over the space of $n_1 x n_2$ for increasing x, having to keep two copies of the older H.

Transformations for up to $k = 5$ were obtained by doing a Knuth-Bendix style elimination procedure among all shifts of the basic recurrence. This was done in Maple and required some careful and extensive manipulations. Table 1 shows the summary of the results.

k	terms	eq. used	max shift x	max shift $n_1, n_2, ...$
2	4	3	-2	-1
3	12	37	-3	-1
4	32	657	-4	-1
5	80	19125	-5	-1

Table 1. recursions

In the supplementary material (http://www.biorecipes.com/TPI/appendix/recursions) we show the recursions for $k = 2$ to $k = 5$. With these recursions it was possible to write a C program that can compute all the TPI values for a genome like yeast in about 6 hours. Previous attempts failed after weeks of computing in very large machines.

Analytic Solution for two Symbols. The case with two symbols can be resolved explicitly (unfortunately, we were not able to find closed forms for higher k, and conjecture that no simple forms exist).

Theorem:

$$H(x, n_1, n_2) \;=\; \binom{n_1}{\lfloor \frac{x}{2} \rfloor}\binom{n_2 - 1}{\lfloor \frac{x-1}{2} \rfloor} \tag{11}$$

This is easily proven by plugging the recursion that defines $H(x, n_1, n_2)$ and separating the case when x is even and when x is odd. For example if x is even then $x = 2w$ and the recursion becomes:

$$\binom{n_1}{w}\binom{n_2 - 1}{w - 1} = \binom{n_1 - 1}{w}\binom{n_2 - 1}{w - 1} + \binom{n_2 - 1}{w - 1}\binom{n_1 - 1}{w - 1} \tag{12}$$

By removing the common factor $\binom{n_2 - 1}{w - 1}$ we get

$$\binom{n_1}{w} = \binom{n_1 - 1}{w}\binom{n_1 - 1}{w - 1} \tag{13}$$

which is a well-known identity of binomial coefficients [12].

2.2 Scale of the TPI

The previous subsection showed how to compute the cumulative distribution which is needed to compute the TPI. These formulas were applied to each individual amino acid. Now comes the question of how to express, with a single index, the joint distribution for all amino acids.

We use convolution of the cumulative distributions, since the measure is the same for each individual amino acid. To facilitate our understanding, we use $\text{TPI}_1 = 1 - 2C_{cum}$ to scale this convolved cumulative distribution to $-1..1$, so that we can talk about negative TPI (more tRNA changes than expected) and positive TPI (fewer changes than expected).

2.3 TPI_2: Variable Codon Frequencies

Some highly expressed genes (e.g. YGR192C) use basically only one tRNA for each amino acid. Although the number of tRNA changes will be practically zero, so will the distribution of possible changes. Hence, the TPI_1 as computed above will be neutral (close to 0) for these genes. This is not desirable, since these highly expressed genes are highly optimized, and have almost the maximum tRNA reusage possible. In some sense, TPI_1 is too independent of the codon usage distribution.

The problem occurs because we have assumed that the choice of codon in a gene is constant, and only consider different orderings of these codons. To resolve this problem, we suggest an alternative TPI_2, where only the choice of amino acids (not codons) is fixed. The test statistic is the same as for TPI_1 (number of tRNA changes). However, the background distribution is estimated from the set of all possible genes resulting in the same protein, where the contribution from each gene is proportional to the probabilities of its codons according to the global codon frequencies in the genome.

For the TPI_2 we will assume that the codon frequency for each amino acid is given. It can be the global distribution or some localized (extracted from a group of genes)

distribution. Contrary to the case for the TPI_1, we have an efficient method of computing the exact distribution (although not a closed form).

We use a similar notation, $C(x, n, \mathbf{P})$ will give us the probability that we find x changes in a sequence of n symbols which appear with probabilities $\mathbf{P} = (p_1, p_2, ..., p_k)$. Similarly $C_i(x, n, \mathbf{P})$ will denote the probability of x changes among the next n symbols when the last symbol is the ith.

$$C(x, n, \mathbf{P}) = \sum_{i=1}^{k} p_i C_i(x, n-1, \mathbf{P}) \tag{14}$$

$$C_i(x, 0, \mathbf{P}) = \delta_{x0} \tag{15}$$

$$C_i(x, n, \mathbf{P}) = p_i C_i(x, n-1, \mathbf{P}) + \sum_{k \neq i} p_k C_k(x-1, n-1, \mathbf{P}) \tag{16}$$

This recursion is fundamentally simpler than the ones for TPI_1, as it is in two variables ($\mathbf{P}$ is a constant). Furthermore, we can compute the functions for increasing n requiring space $O(nk)$ and time $O(n^2k)$. The computation of TPI_2 for all yeast genes is done in approximately 10 minutes on a modern desktop computer.

The moments of $C(x, n, \mathbf{P})$ have simple expressions, but in this case we can compute the exact distribution for all practical cases, so there is no point in using them.

3 Results

In a recent experiment with synthesized GFP genes, it was shown that genes with maximum tRNA reusage were translated faster than genes with minimum tRNA reusage [3]. To complement that biochemistry experiment of one gene, a bioinformatics analysis is provided here. We examine whether different gene groups have higher or lower average TPI than what can be expected by chance. Gene groups from expression connection (http://db.yeastgenome.org/cgi-bin/expression/expressionConnection.pl) were used to evaluate TPI_1 and TPI_2 (Table 2). For comparison, we also included the CAI index [5].

Overall, TPI_1 does not show a clear correlation with gene expression. The exception is for cell cycle genes, a group that requires fast translation. Cell cycle genes that increase expression at least 10-fold have an average TPI_1 of 0.365, which is significant with p-value 0.013. This is an indication that there is a selection for high tRNA reusage in cell cycle genes.

TPI_2, which is also correlated with CAI, shows a clear correlation with several gene expression groups. Genes from the cell cycle, zinc levels, DNA damage, diauxic shift and glucose limitation experiments have a significantly high average TPI_2. This is an additional indication for high tRNA reusage in cell cycle genes.

Experiment	avg(TPI_1)	p(TPI_1)	avg(TPI_2)	p(TPI_2)	avg(CAI)	p(CAI)
glucose limitation 5x	0.142	0.463	0.512	0.0108	0.240	0.0285
sporulation 20x	0.028	0.942	-0.045	0.9831	0.140	0.9987
diauxic shift 10x	0.215	0.222	0.428	0.0062	0.209	0.0583
DNA damage 10x	0.048	0.861	0.466	0.00001	0.223	0.0015
alpha-factor over time 80x	0.144	0.337	-0.037	0.9992	0.138	1.0000
alpha-factor (var. conc.) 20x	0.192	0.148	0.173	0.207	0.157	0.8808
stress response 50x	0.163	0.251	0.236	0.0267	0.174	0.3849
histone depletion 50x	0.140	0.440	0.207	0.1681	0.170	0.4894
zinc levels 10x	0.198	0.236	0.461	0.0015	0.246	0.0017
phosphate pathway 10x	0.328	0.048	0.215	0.2410	0.180	0.3233
cell cycle 5x	0.175	0.153	0.310	0.00017	0.194	0.0152
cell cycle 10x	0.365	0.013	0.465	0.0026	0.21	0.0900

Table 2. Average TPI's (scale: -1..1) and codon adaptation index (CAI) for groups of genes that are up-regulated in different experiments. Consider as an example the first row: average TPI's and CAI were computed for the group of genes with at least 5-fold up-regulation when subjected to glucose limitation. The same was done for 100000 groups of (equally many) randomly picked genes, and p-values were computed by comparing the real average TPI's and CAI to their respective distribution of random values.

4 Pairwise Codon Bias Suggests Context-Dependent DNA Repair in *Saccharomyces cerevisiae*

We observe a very significant bias in the selection of synonymous codons in the coding DNA of *S. cerevisiae*. The bias appears in DNA, its reverse complement and in each amino acid individually which strongly suggests that it is neither a translational nor RNA effect, but a feature of the DNA. Isolating this bias from other possible sources, we conjecture that this is caused by a context dependent DNA mismatch correction mechanism.

Given the complete set of coding sequences of the *S. cerevisiae* genome (release 36 of the *S. cerevisiae* genome from EMBL [13]), we can calculate the observed frequencies of amino acids ($f^a_{obs}(A)$) and codons ($f^c_{obs}(C)$) as well as the observed pairwise amino acid ($f^{aa}_{obs}(A_1, A_2)$) and codon frequencies ($f^{cc}_{obs}(C_1, C_2)$). Using f^{aa}_{obs} and f^c_{obs}, we can calculate the *expected* pairwise codon frequencies f^{cc}_{exp} under the hypothesis of independent codon pairing.

We compute the difference between the observed value and the expected value, in units of a standard deviation and get differences of up to 25 standard deviations. If we group the codon pairs according to the nucleotides at the frame border, that is, for a codon pair $x_1x_2x_3$ and $y_1y_2y_3$, the nucleotides x_3 and y_1, we get even more significant results (up to 83.10 standard deviations, Table 3).

A look at the most biased codon pair patterns (Table 4) quickly reveals that the reverse complement of each pattern is similarly biased. This indicates that whatever is causing the observed bias is probably not dependent on the reading-sense of the sequences, since it also appears in the reverse-complement strand in identical form.

$x_3 \mid y_1$	A	C	G	T
A	19.74	-4.68	-23.46	5.79
C	**77.90**	-11.21	**-61.09**	-19.50
G	-0.67	30.52	2.84	-29.46
T	**-83.10**	-11.20	**65.06**	31.50

Table 3. Difference in standard deviations between expected and observed counts of pairwise codons grouped by the nucleotides at the frame border.

Furthermore, if we re-compute the values in Table 3 for each amino acid with more than one codon, using the third nucleotide in each codon for the position x_3 and the observed and expected pairwise codon frequencies for each amino acid, the resulting biases (although not all pairs are present for each amino acid) match those in Table 3.

In summary, the pairwise codon bias observed seems to be caused by some mechanism that operates directionally on DNA since it is independent of the coding strand and of the individual amino acids. This excludes as possible mechanisms post-translational events.

We postulate that the bias is caused by a context sensitive DNA mismatch correction mechanism (CSCM). Although DNA replication is in general a very faithful process, errors in the form of mismatches can arise. The most common form of mismatches, and the only type we will consider here, are the purine·pyrimidine mismatches C·A and T·G. Such a mismatch can occur at any given position in the sequence. To avoid DNA packing and transcription problems, such mismatches need to be corrected to either T·A or C·G. In organisms which do not mark the original strand during DNA replication (*e.g. S. cerevisiae*), corrections could be made according to:

- a static, context insensitive correction rule, always correcting to either T·A or C·G. Such a mechanism would create an obvious T·A or C·G bias.
- a random correction scheme, which would not produce a C·G-bias, however, it would not produce the observed codon pair frequency bias either.

In the absence of a strong C·G bias we assume that a more refined context sensitive mechanism exists. We also assume that such a mechanism would produce *better* corrections than the static or random models. For our study we will define a *good* correction

Pattern	bias	Reverse	bias	Pattern	bias	Reverse	bias
`**T A**`	-83.10	self		`**C G**`	-61.09	self	
`**C A**`	77.90	`**T G**`	65.06	`*TC A**`	59.07	`**T GA*`	25.51
`**T AA*`	-70.59	`*TT A**`	-69.96	`*CC A**`	54.69	`**T GG*`	32.07
`*TT A**`	-69.96	`**T AA*`	-70.59	`**T A*A`	-52.78	`T*T A**`	-45.06
`**T G**`	65.06	`**C A**`	77.90	`**T A*G`	-52.61	`C*T A**`	-51.43
`**C AA*`	62.39	`*TT G**`	36.10	`**T AG*`	-51.57	`*CT A**`	-48.28
`*TT AA*`	-62.02	self		`C*T A**`	-51.43	`**T A*G`	-52.61

Table 4. The 20 most significantly biased patterns and their reverse complement. The indented patterns are subsets of an already listed pattern. The bias is given in standard deviations.

x/y	A	C	G	T
A	A·T	A·T	A·T	A·T
C	**C·G**	A·T	A·T	A·T
G	**C·G**	A·T	A·T	A·T
T	**C·G**	A·T	A·T	A·T

Table 5. A·C portion of the CSCM. The T·G portion is shown to be identical for optimal correction schemes.

as a correction that does not alter the primary structure of the sequence, since it is the only type of error we can quantify.

We define a CSCM as a scheme in which the mismatch correction depends on the mismatch itself and its two neighbouring nucleotides. Considering mismatches in a strand-independent (symmetric) correction scheme, we define the local context as the nucleotides upstream and downstream of the pyrimidine in the mismatch. We shall call these nucleotides x and y. We represent the CSCM as a Table in which for each x and y an A·C or T·G mismatch is corrected to either A·T or C·G (Table 5).

We define the quality of a CSCM as the percentage of correct corrections (PCC). This is the relative number of mismatch corrections which do not induce a change in the primary structure of the sequence or the insertion or deletion of a stop-codon (note that mutations to synonymous codons are accepted).

The optimal CSCM (*i.e.* the CSCM with the highest PCC of 74.18% as opposed to 67.12% for random corrections) for the *S. cerevisiae* genome is shown in Table 5. Due to its rather simple structure, we can define, for each nucleotide, *protected* and *vulnerable* contexts:

- **A** preceded by T is *vulnerable*, since after correction it would change to G.
- **C** followed by A is *protected*, since after correction it would remain a C.
- **G** preceded by T is *protected*, since after correction it would remain a G.
- **T** followed by A is *vulnerable*, since after correction it would change to C.

These four contexts are exactly the four most biased nucleotide combinations observed over the frame border (Table 3). Note that the observed bias has very little influence on the choice of the optimal CSCM: the optimal CSCM computed over the *expected* pairwise codon frequencies is identical. **The optimal CSCM could therefore create the observed codon bias.**

5 Conclusions

We have defined and presented two versions of the TPI. The first one assumes constant codon frequencies (codon shuffling). This makes it insensitive to codon bias and different GC content in different parts of the genome. Still, gene expression analysis indicates that tRNA reusage is important to cell cycle genes. The advantage of TPI_1 is that it is constructed to be independent of codon usage, GC content etc, which indicates that the observed signal indeed is due to tRNA reusage.

The second TPI version only assumes constant amino acid sequence, and uses the global codon frequencies to estimate the background distribution. The result is an index

that is more similar to the codon adaptation index (CAI). Also, overall TPI_2 p-values correlate quite well with CAI. However, for cell cycle regulated genes, the p-value is much more significant than for CAI. This is an additional indication that tRNA reusage is important for cell cycle genes. Also, the DNA damage experiments show much higher correlation with TPI than with CAI.

The two versions of TPI complement each other. TPI_1 shows that tRNA reusage exists, since it measures tRNA reusage independently of other biases. However, assuming a constant codon population in a gene reduces the measurable tRNA reusage signal. Naturally, if tRNA reusage is a way to optimize translation speed, one can assume that evolution can optimize codon bias for this purpose. By not assuming a constant codon population in a gene, TPI_2 measures tRNA reusage more accurately as it happens in the cell, which is confirmed by the TPI_2 p-values for the cell cycle genes.

In addition to the tRNA reusage bias, we observe an additional unexpected codon bias in *S. cerevisiae*. Isolating this bias from other possible sources, we conjecture that this is caused by a context dependent DNA mismatch correction mechanism.

References

1. Elf, J., Nilsson, D., Tenson, T., Ehrenberg, M.: Selective charging of tRNA isoacceptors explains patterns of codon usage. Science **300** (2003) 1718–1722
2. Dittmar, K.A., Sorensen, M.A., Elf, J., Ehrenberg, M., Pan, T.: Selective charging of tRNA isoacceptors induced by amino-acid starvation. EMBO Rep. **6** (2005) 151–157
3. Barral, Y., Faty, M., von Rohr, P., Schraudolph, N.N., Friberg, M.T., Roth, A.C., Gonnet, P., Cannarozzi, G.M., Gonnet, G.H.: tRNA recycling in translation and its influence on the dynamics of gene expression in yeast. in preparation (2006)
4. Bennetzen, J.L., Hall, B.D.: Codon selection in yeast. J Biol Chem **257** (1982) 3026–3031
5. Sharp, P.M., Li, W.H.: The codon adaptation index–a measure of directional synonymous codon usage bias, and its potential applications. Nucleic Acids Res. **15** (1987) 1281–1295
6. Wright, F.: The 'effective number of codons' used in a gene. Gene **87** (1990) 23–29
7. Boycheva, S., Chkodrov, G., Ivanov, I.: Codon pairs in the genome of *Escherichia coli*. Bioinformatics **19** (2003) 987–998
8. Carbone, A., Zinovyev, A., Kepes, F.: Codon adaptation index as a measure of dominating codon bias. Bioinformatics **19** (2003) 2005–2015
9. Greenbaum, D., Colangelo, C., Williams, K., Gerstein, M.: Comparing protein abundance and mRNA expression levels on a genomic scale. Genome Biol **4** (2003) 117
10. Friberg, M., von Rohr, P., Gonnet, G.: Limitations of codon adaptation index and other coding DNA-based features for prediction of protein expression in saccharomyces cerevisiae. Yeast **21** (2004) 1083–1093
11. Char, B.W., Geddes, K.O., Gonnet, G.H., Leong, B.L., Monagan, M.B., Watt, S.M.: Maple V Language Reference Manual. Springer-Verlag (1991)
12. Abramowitz, M., Stegun, I.: Handbook of Mathematical Functions, chapter 24.1.1. Dover, New York (1972)
13. Kulikova, T., Aldebert, P., Althorpe, N., Baker, W., Bates, K., Browne, P., van den Broek, A., Cochrane, G., Duggan, K., Eberhardt, R., Faruque, N., Garcia-Pastor, M., Harte, N., Kanz, C., Leinonen, R., Lin, Q., Lombard, V., Lopez, R., Mancuso, R., McHale, M., Nardone, F., Silventoinen, V., Stoehr, P., Stoesser, G., Tuli, M.A., Tzouvara, K., Vaughan, R., Wu, D., Zhu, W., Apweiler, R.: The EMBL nucleotide sequence database. Nucleic Acids Res. **32** (2004) D27–D30