sscu user manual (2.0.0)

Yu Sun

2016-09-22

Contents

1. Background

The central dogma is the cornerstone of modern molecular biology, as it explains the information processing flow within a biological system is in the direction of DNA to RNA to protein (crick 1970). An important step in this process is to decode the information stored in the cryptic nucleotides sequences into protein molecules.

Sixty-four tri-nucleotide units (codons) are correspond to the coding of twenty amino acids, thus almost all the amino acids has more than one matching codons. The codon choice for an amino acid is not random for most species, and it has been shown that mutation and selection are the two major forces shaping the codon usage pattern. For the majority of the gene, mutation bias is the factor determining the codon choice, but the influence from selection is increased with the genes' expression level.

Several programs and packages has been release for the codon usage analysis, such as CodonW. However, most of these programs focus on the overall usage of codons, more specificly, the mutational influence on the codons usage. Many theorical work and statistical tools has been developed to disentangle the effect of selection on codon usage, such as Sharp's S index, Akashi test, identify optimal codons by highly expressed genes and correlative method. However, no program or pipeline has been released to do the calculation, thus every codon selection study need to write individual code to perform the same task. It is not only waste a lot of time, but also have the possibility to introduce errors during the coding process. In this R package, we mainly focus on developing functions for identify selection on codon usage, and included functions for the following tests.

1.1 S index

Sharp released a mathematic formula, named as S index, to quantify the strength of selection in the synonymous codon. The method calculates the relative proportion of C- and U-ending codons in the two codon boxes, and also takes into consideration of the genomic mutation bias. The method has been proved to be an effective way to quantify and compare the codon selection among different species.

1.2 Akashi's test

The function calculate Akashi's test for translational accuracy selection on coding sequences. Akashi proposed the translational accurary theory for codon usage in 1994. The theory suggest that the optimal codons are codons that translated more accurately than the other codons, and these codons are favored in the important sites, such as the evolutionary conserved amino acid sites, whereas the less conserved amino acids sites are more tolerable to the non-optimal codons. For detailed information, see reference (Akashi H. Synonymous codon usage in Drosophila melanogaster: natural selection and translational accuracy. Genetics 1994 Mar;136(3):927-35.) and Drummond sites.

1.3 Identify optimal codons

Two major method has been used to identify optimal codons: highly expressed gene method (short as highly method) and correlative method. The highly method compare the codon usage in the highly and lowly/all genes. The basic idea is that the highly expressed genes are enriched with more optimal codons, due to the translational efficiency or accuracy reasons. The correlative method identifies the optimal codons based on the method from Hershberg & Petrov (Hershberg R, Petrov DA. 2009. General rules for optimal codon choice. Plos Genet. 5:e1001115.). This method take the whole genome into consideration, and predict the optimal codons by making the correlation between the frequency of each codon within each gene and the overall codon bias (Nc or Nc').

1.4 Other functions

Other functions include genomic gc3 calculation, optimal codon identification and proportion index for the four and six codon boxes.

Overall, the package is useful for the calculation of selective profile in codon usage studies, and it is a good complement to other major codon usage programs, such as CodonW.

2. Function overview

2.1 akashi_test

akashi_test(contingency_file=NULL)

The function calculate Akashi's test for translational accuracy selection on coding sequences.

The input of the function is a contingency file, you can find an example in the folder akashi_test in the sscu directory. You can either make the file by yourself, or by the perl script make_contingency_table.pl in the akashi_test folder. You can check the detailed usage of the perl script by reading the first few lines in the perl script. The perl script tabulates and calculate the a,b,c,d entries Drummond sites for the coding sequences, and output the four values for each amino acid for each gene, example as the contingency_file_Gvag. The input of the perl script is a folder contains the codon alignments of the genes that you are interested to calculate.

The function reads and calculates the Z value, p value and odd ratio for the Akashi's test. In addition, it calculates the conserved, variable, optimal and nonoptimal sites for the coding sequences.

2.2 genomic_gc3

genomic_gc3(genomic_cds_file)

The function calculates the genomic gc3 for an multifasta genomic CDS file. The function first concatenated all the CDS sequences in the file into one long CDS string, than calculated the gc3 from the GC3 function in seqinr package. You can also use the function to calculate the gc3 for a single gene, or a set of genes, depends what CDS sequences you put in the input file. The result can be used as input for the s_index calculation.

2.3 op_corre_codonW

op_corre_CodonW(genomic_cds_file=NULL, correspondence_file=NULL)

The function identify the optimal codons based on the correlative method from Hershberg & Petrov. This method take the whole genome into consideration, and predict the optimal codons by making the correlation between the frequency of each codon within each gene and the overall codon bias (Nc or Nc'). The input file include the correspondence analysis output file from the program CodonW (to get the Nc value), and the genomic cds file (to get the codon usage information for each gene).

For further details regard how to use CodonW, you can refer to the site http://codonw.sourceforge.net/. Note, you must input the same genomic cds file to CodonW and to the op_corre_CodonW funtion, so that the order and number of genes are consistent in the files.

2.4 op_corre_NCprime

op_corre_NCprime(genomic_cds_file=NULL, nc_file=NULL)

The function identify the optimal codons based on the correlative method from Hershberg & Petrov. This method take the whole genome into consideration, and predict the optimal codons by making the correlation between the frequency of each codon within each gene and the overall codon bias (Nc or Nc'). The input file include the output file from the program ENCprime (to get the Nc and Nc' value), and the genomic cds file (to get the codon usage information for each gene).

For further details regard how to use ENCprime, you can refer to the site https://github.com/jnovembre/ENCprime. Note, you must input the same genomic cds file to ENCprime and to the op_corre_NCprime funtion, so that the order and number of genes are consistent in the two files.

2.5 op_highly

op_highly(high_cds_file = NULL,ref_cds_file = NULL,p_cutoff = 0.05)

Optimal codons are defined as codons significantly enriched in the highly expressed genes compared to the lowly expressed genes, or other set of appropriate reference genes. In another word, these codons were favored by translational selection, which was strongest among highly expressed genes. This function calculate the optimal codon list with p-values, thus user could have a general idea of which codons were preferred by selection in the genome.

The argument high_cds_file should specific the path for the highly expressed gene dataset. It is up to the users how to define which dataset of highly expressed genes. Some studies use the expression data, or Nc value to divide genes into highly/lowly sets. Other studies use a specific dataset, such as only including the very highly expressed genes (ribosomal genes).

The argument ref_cds_file should specific the path for the lowly expressed gene dataset, or any appropriate dataset. In Sharp PM paper (Forces that influence the evolution of codon bias), he used the all gene data set as neutral reference and also get a list of optimal codons.

The argument p_cutoff set the cutoff for p values in the chi.square test. Only codons are significantly enriched in the highly expressed genes are marked with + symbol in the ouotput tables. The codons are significantly lower presented in the highly expressed genes are marked with - symbol. The codons are not significantly differently presented compared to the reference dataset are marked with NA symbol.

2.6 optimal_codon_statistics

optimal_codon_statistics(high_cds_file=NULL,ref_cds_file=NULL,p_cutoff=0.05)

If you want to know the detailed codon usage information, you can run the function optimal_codon_statistics in the package. The optimal codons are defined as codons significantly enriched in the highly expressed genes compared to the lowly expressed genes, or other set of reference genes. The function calculate the optimal codon list with p-values, thus user could have a general idea of which codons were preferred by selection in the genome.

The argument high_cds_file should specific the path for the highly expressed gene dataset. It is up to the users how to define which dataset of highly expressed genes. Some studies use the expression data, or Nc value to divide genes into highly/lowly sets. Other studies use a specific dataset, such as only including the very highly expressed genes (ribosomal genes).

The argument ref_cds_file should specific the path for the lowly expressed gene dataset, or any appropriate dataset. In Sharp PM paper (Forces that influence the evolution of codon bias), he used the all gene data set as neutral reference to infer codons used significantly in highly expressed genes.

The argument p_cutoff set the cutoff for p values in the chi.square test. Only codons are significantly enriched in the highly expressed genes are marked with + symbol in the ouotput tables. The codons are significantly lower presented in the highly expressed genes are marked with - symbol. The codons are not significantly differently presented compared to the reference dataset are marked with NA symbol.

The function also output the rscu value for the high expressed dataset and reference dataset, actual and expected number of codons, the p value of chi.square test.

2.7 proportion_index

proportion_index(high_cds_file=NULL,genomic_cds_file=NULL)

The function optimal_index is developed in this study, which estimate the relative amount of GC-ending optimal codon for the four and six codon boxes codon in a given mutational background. The function has similar mathematical formula as sscu and also take into account of background mutation rate, thus is comparable with the S index. However, since the set of GC-ending optimal codons are likely to be different among different species, the index can not be compared among different species.

The argument high_cds_file must be specified with the input filepath for the highly expressed genes. The file should be a multifasta file contains 40 highly, including elongation factor Tu, Ts, G, 50S ribosomal protein L1 to L6, L9 to L20, 30S ribosomal protein S2 to S20. This file can be generated by either directly extract these DNA sequence from genbank file, or parse by blast program. For the four amino acids (Phy, Tyr, Ile and Asn), the C-ending codons are always preferred than the U-ending codons. Thus, only these four codons were taken into account in the analyses.

The arguments, genomic_cds_file, is used to calculate the genomic mutation rate (gc3). The genomic_cds_file should be a multifasta file contains all the coding sequences in the genome, and the function use it to calculate the genomic gc3 and mutation rate.

Noted, most of the AT biased genomes do not have any GC-ending optimal codons for the four and six codon boxes, thus the function will report NA as output.

2.8 s_index

s_index(high_cds_file = NULL, genomic_cds_file = NULL, gc3 = NULL)

The function calculate the s index based on Paul Sharp's method. The method take into account of background mutation rate (in the program, two arguments genomic_cds_file and gc3, are input to calculate the mutation rate), and focus only on codons with universal translational advantages in all bacterial species (in the program, one argument high_cds_file, is input to calculate these codons). Thus the s index can be used to quantify the strength of translational selection and is comparable among different species.

The argument high_cds_file much be specified with the input filepath for the highly expressed genes. The file should be a multifasta file contains 40 highly, including elongation factor Tu, Ts, G, 50S ribosomal protein L1 to L6, L9 to L20, 30S ribosomal protein S2 to S20. This file can be generated by either directly extract these DNA sequence from genbank file, or parse by blast program. For the four amino acids (Phy, Tyr, Ile and Asn), the C-ending codons are universally preferred than the U-ending codons. Thus, only these four codons were taken into account in the analyses.

The second arguments, genomic_cds_file or gc3, is used to calculate the genomic mutation rate, and one of them must be specified. The genomic_cds_file should be a multifasta file contains all the coding sequences in the genome, and the function use it to calculate the genomic gc3 and mutation rate. If the gc3 value for the genome is known already, or calculated by genomic_gc3 function below, you can specify it in the argument gc3. If both the genomic_cds_file and gc3 arguments are specified, the function will use the genomic_cds_file to calculate mutation rate, and neglect the gc3 argument.

3 Workflow

3.1 general statistics

Here is an example of how to use these functions from the package. We will use the genomic CDS file from the Lactobacillus. kunkeei and Gardnerella vaginalis as exemple for the workflow. These files are included in the sequences folder in the package.

First, we can calculate the genomic gc3 for the L. kunkeei to have a general statistics about the genome. We show how to do it here with genomic_gc3 function:

```r
library(sscu)
genomic_gc3(system.file("sequences/L_kunkeei_genome_cds.ffn",package="sscu"))
```

```
## [1] 0.3083719
```

The genomic gc3 content is 0.308, which is a low gc genome.

Then, we can further check the codon usage pattern in detailes for L. kunkeei by using the optimal_codons function:

```
head(optimal_codon_statistics(high_cds_file=system.file("sequences/L_kunkeei_highly.ffn",package="sscu"
               ref_cds_file=system.file("sequences/L_kunkeei_genome_cds.ffn",package="sscu")))
```

```
##     codon aa rscu_high rscu_ref high_No_codon high_expect_No_codon
## TTT   TTT  F      0.66     1.10            58                   88
## TTC   TTC  F      1.34     0.90           118                   88
## TTA   TTA  L      2.58     2.54           198                   77
## TTG   TTG  L      0.56     1.09            43                   77
## TCT   TCT  S      1.10     1.04            64                   58
## TCC   TCC  S      0.12     0.38             7                   58
##     ref_No_codon ref_expect_No_codon p_value symbol
## TTT        10540                9539   0.002      -
## TTC         8538                9539   0.006      +
## TTA        16254                6399   0.898     NA
## TTG         6971                6399   0.000      -
## TCT         5263                5062   0.796     NA
## TCC         1937                5062   0.003      -
```

We can see several statistics for each codon, including RSCU values, actual and expected number of codons, the p value of chi.square test. The symbol indicates if the codon is highly (optimal codons), no difference, or lowly presented in the highly expressed genes compared to the reference gene set. Thus, you can get a list of optimal codons for the output. The output has 64 lines with each line for each codon, and we only show the first six lines for simplicity.

3.2 optimal codons

Optimal codons are defined as codons favored by translational selection, by promoting either translational efficiency or accuracy. If we want to get a list of optimal codons by the highly method:

```
op_highly(high_cds_file=system.file("sequences/L_kunkeei_highly.ffn",package="sscu"),
          ref_cds_file=system.file("sequences/L_kunkeei_genome_cds.ffn",package="sscu"))
```

```
##  [1] "AAC" "ACT" "ATC" "CGT" "CTA" "GAC" "GCT" "GGT" "GTT" "TAC" "TCA"
## [12] "TTC"
```

Now we get a list of optimal codons in L. kunkeei "AAC" "ACT" "ATC" "CGT" "CTA" "GAC" "GCT" "GGT" "GTT" "TAC" "TCA" "TTC". You can input the list to the further analysis akashi_test.

If you want to get a list of optimal codons by the correlative method, you need to input a file with Nc or Nc' information for all the genes you are interested. Now you have two options, either input the Nc file by the correspondence analysis from CodonW, or input the Nc/Nc' file by ENCprime:

```
op_corre_CodonW(genomic_cds_file=system.file("sequences/Gvag_genome_cds.ffn",package="sscu"),
                correspondence_file=system.file("correlative_test/Gvag.codonw",package="sscu"))
```

```
## Warning in cor(rank(x), rank(y)): the standard deviation is zero

## Warning in cor(rank(x), rank(y)): the standard deviation is zero

## Warning in min(correlations_nc[aa_codons[[i]]][logic_vec]): no non-missing
## arguments to min; returning Inf

## Warning in min(correlations_nc[aa_codons[[i]]][logic_vec]): no non-missing
## arguments to min; returning Inf

##  [1] "act" "cct" "gct" "ggt" "gtt" "tct" "cgt" "ttg" "att" "aag" "aat"
## [12] "cag" "cat" "gat" "tgc" "ttc"
```

```
op_corre_NCprime(genomic_cds_file=system.file("sequences/LbDelBA1_genome_cds.ffn",package="sscu"),
                 nc_file=system.file("correlative_test/LbDelBA1.NCprime",package="sscu"))
```

```
## Warning in cor(rank(x), rank(y)): the standard deviation is zero

## Warning in cor(rank(x), rank(y)): the standard deviation is zero

## Warning in cor(rank(x), rank(y)): the standard deviation is zero

## Warning in cor(rank(x), rank(y)): the standard deviation is zero

## Warning in min(correlations_nc[aa_codons[[i]]][logic_vec]): no non-missing
## arguments to min; returning Inf

## Warning in min(correlations_ncp[aa_codons[[i]]][logic_vec]): no non-missing
## arguments to min; returning Inf

##  [1] "acc" "cca" "gcc" "ggc" "gtc" "tcc" "cgc" "ctg" "atc" "aag" "aac"
## [12] "cac" "gaa" "gac" "tac" "tgc" "ttc"
```

Generally, these two test should give similar optimal codon list, but for some cases, especially for genomes with low GC content, the list could be different, for further details, read Hershberg R, Petrov DA. 2009. General rules for optimal codon choice. Plos Genet. 5:e1001115.

3.3 selective profile

Next, if we want to see the selective profile for the species, which is the major function in the package. We can use the s_index function to do the calculation. We can use either the genomic gc3 content 0.308 as we got previously:

```
s_index(gc3=0.308,
        high_cds_file=system.file("sequences/L_kunkeei_highly.ffn",package="sscu"))
```

```
## [1] 1.541898
```

or you can directly input the CDS file in the function.

```r
s_index(high_cds_file=system.file("sequences/L_kunkeei_highly.ffn",package="sscu"),
        genomic_cds_file=system.file("sequences/L_kunkeei_genome_cds.ffn",package="sscu"))
```

```
## [1] 1.540154
```

We can get the S index as 1.54 from both method. S index = 0 indicates that, in the highly expressed genes, the relative frequency of C/U-ending codons are exactly the same as the genomic gc3 (mutational pattern). S index > 0.3 indicate C-ending codons are used in significantly higher frequency than the U-ending codons, which further indicates that translational selection is affecting codon usage. In this case, S index of 1.54 suggests strong stranslational selection in L. kunkeei, which is even higher than E. coli (1.49).

Akashi's test is another way to evaluate the translational selection on codon usage. Akashi's theory suggests that the optimal codons are codons that translated more accurately than the other codons, and these codons are favored in the important sites, such as the evolutionary conserved amino acid sites, whereas the less conserved amino acids sites are more tolerable to the non-optimal codons.

```r
akashi_test(system.file("akashi_test/contingency_file_Gvag",package="sscu"))
```

```
## $Z
## [1] 4.443357
##
## $p
## [1] 4.428296e-06
##
## $odd_ratio
## [1] 1.081472
##
## $conserved_optimal_sites
## [1] 17774
##
## $conserved_non_optimal_sites
## [1] 48683
##
## $variable_optimal_sites
## [1] 13235
##
## $variable_non_optimal_sites
## [1] 54958
##
## $con_var_ratio
## [1] 0.9745428
```

3.4 a new function of evaluate selection on the four and six codon boxes

The translational selection in the two codon boxes is very strong in L. kunkeei, the frequency is C-ending codons is significantly higher than U-ending codons, even taking into consideration of the genomic mutation bias. For the four and six codon boxes, the selection is generally very low, so the codon usage frequency is generally follow the mutation bias pattern. But the four and six codon boxes also have some optimal codons, are they strong enough to deviate the codon frequency from the mutational pattern? In this study, we develop a function called optimal_index to estimate the relative frequency between GC/AU-ending codons in the optimal codons in the four and six codon boxes.

Here is an example of L. kunkeei genome:

```r
proportion_index(high_cds_file=system.file("sequences/L_kunkeei_highly.ffn",package="sscu"),
           genomic_cds_file=system.file("sequences/L_kunkeei_genome_cds.ffn",package="sscu"))
```

```
## [1] NA
```

It indicates that the four and six codon boxes does not have any GC-ending optimal codons. This is general pattern for most species, suggesting muation is the dominant force for the codon choice for the four and six codon boxes.

This is another example of G. vaginalis:

```r
proportion_index(high_cds_file=system.file("sequences/Gvag_highly.ffn",package="sscu"),
           genomic_cds_file=system.file("sequences/Gvag_genome_cds.ffn",package="sscu"))
```

```
## $optimal_mutation_index
## [1] 0.01422826
##
## $s_index
## [1] 0.9697102
##
## $ratio
## [1] 0.01467269
```

It has GC-ending optimal codons, but the optimal index is only 0.014. The intepretation of optimal index is very similar to S index: the value 0 means the frequency of optimal codons is very close to the mutational pattern. Thus, for the G. vaginalis, although the translational selection can be detected in the four and six codon boxes (as the optimal codons), but they are so weak do alter the mutational dominated codon frequency. In another words, these optimal codons are low frequency optimal codons.

Again, we can use the function optimal_codons to get the detailed codon usage data:

```r
head(optimal_codon_statistics(high_cds_file=system.file("sequences/Gvag_highly.ffn",package="sscu"),
               ref_cds_file=system.file("sequences/Gvag_genome_cds.ffn",package="sscu")))
```

```
##     codon aa rscu_high rscu_ref high_No_codon high_expect_No_codon
## TTT   TTT  F      0.41     1.28            46                  112
## TTC   TTC  F      1.59     0.72           177                  112
## TTA   TTA  L      0.17     1.24            16                   92
## TTG   TTG  L      2.08     1.66           192                   92
## TCT   TCT  S      2.18     2.07           135                   62
## TCC   TCC  S      1.74     0.47           108                   62
##     ref_No_codon ref_expect_No_codon p_value symbol
## TTT        10639                8316   0.000      -
## TTC         5992                8316   0.000      +
## TTA         8260                6686   0.000      -
## TTG        11095                6686   0.082     NA
## TCT        11188                5396   0.690     NA
## TCC         2524                5396   0.000      +
```

By checking the optimal codon detailed information, it is clear several optimal codons do have low proportion among the highly expressed genes. For example GUC for Valine, GCC for Alanine, CUC for Leucine are all in low proportion compare to the corresponding non-optimal U-ending codons.

4 Folders installed in the sscu package

There are three additional installed folder in the sscu package directory, named as sequences, akashi_test and correlative_test

4.1 sequences

As the name indicates, the folder has five sequences files: three whole genome coding sequence file and two highly expressed genes coding sequence file

4.2 akashi_test

This folder contains one sub-folder bifidos_alignments, one perl script make_contingency_table.pl and one data file contingency_file_Gvag. All the files are related to the function akashi_test in the package.

The akashi_test function require the input of an contingency file (example as contingency_file_Gvag). The contingency file can be generated by the perl script, you can check in detail about how to use it by reading the first few lines of the perl script. The perl script tabulates and calculate the a,b,c,d entries for the coding sequences, and output the four values for each amino acid for each gene. The input of the perl script is a folder (example as bifidos_alignments) contains the codon alignments of the genes that you are interested to calculate.

4.3 correlative_test

This folder contains two files as input to calculate optimal codons by correlative method. Gvag.codonw is the example input file for op_corre_CodonW function, and LbDelBA1.NCprime is the example input file for op_corre_NCprime function.