# kaos

**Title** Improved Python Package for DNA Sequence Encoding using Frequency Chaos Game Representation

**Version** 0.15

**Author** Abhishek Halder, Piyush, Bernadette Mathew, Debarka Sengupta

**Description** DNA sequences encoding by using the Frequency Chaos Game Representation (FCGR).

**Imports** collection, numpy, pandas, biopython

**Repository** GitHub

**Functions:**

kaos.read_fasta

**Description**

Read FASTA file

**Usage**

kaos.read_fasta(file_path: str)

**Arguments**

| | |
|---|---|
| file_path | The path to the FASTA file. |

**Details**

This function reads a FASTA file and produces a concatenated DNA sequence

**Returns**

str: Concatenated DNA sequence from the FASTA file

**Example**

```
###load data
file_path ="GCF_000005845.2_ASM584v2_genomic.fna"

### read fasta
fasta_seq = kaos.read_fasta(file_path)

###print fasta sequence
print(fasta_seq)
```

kaos.chaos_game_representation_key

**Description**

Generate FCGR key matrix

**Usage**

```
chaos_game_representation_key(kmer_length: int)
```

**Arguments**

| | |
|---|---|
| kemr_length | The length of the kmer |

**Details**

This function produces the FCGR matrix key by giving kmer length.

**Returns**

| | |
|---|---|
| np.ndarray | A 2D numpy array representing the key matrix for FCGR. |

**Examples**

```
### Generate the FCGR keys for kmers of length 2 using the 'kaos' module.
chaos_game_kmer_array = kaos.chaos_game_representation_key(kmer_length=2)

### Display the array containing the FCGR matrix keys.
chaos_game_kmer_array
```

```
[['CC', 'CG', 'GC', 'GG'],
['CA', 'CT', 'GA', 'GT'],
['AC', 'AG', 'TC', 'TG'],
 ['AA', 'AT', 'TA', 'TT']]
```

kaos.return_kmer_index

**Description**

Returns the index of a specific kmer in the FCGR matrix.

**Usage**

return_kmer_index(kmer: str, kmer_length: int)

**Arguments**

| | |
|---|---|
| kmer | The kmer for which the index is to be found. |
| kmer_length | The kmer length required to form FCGR |

**Details**

This function accepts a specific kmer and kmer length required to form FCGR and then determines the position of that kmer within the FCGR key matrix.

**Returns**

| | |
|---|---|
| tuple | The row and column indices of the kmer in the FCGR key matrix. |

**Example**

```
### Get the index of kmer "AAA" in the FCGR array using kaos module.
kaos.return_kmer_index(kmer = "AAA", kmer_length=3)


# Output shows the position of "AAA" in the array
(7,0)
```

kaos.return_kmer_at_index

**Description**

Returns the kmer at the specified index in the FCGR key matrix

**Usage**

return_kmer_at_index(kmer_length: int, tuple_index: tuple)

**Arguments**

| | |
|---|---|
| kmer_length | The length of the kmer used to generate the FCGR matrix. |
| tuple_index | The index (row, column) of the kmer in the matrix. |

**Details**

This function requires two arguments: kmer_length and tuple_index. The kmer_length specifies the length of the kmer to form the FCGR matrix. The tuple_index indicates the position of the kmer within the array. The function returns the kmer of the specified length found at the given index.

**Returns**

| | |
|---|---|
| str | The kmer at the specified index |

**Example**

```
### Retrieve the kmer at the specified index (7, 0) with a length of 3 from the chaos
        game representation.
kaos.return_kmer_at_index(kmer_length=3, tuple_index=(7, 0))


### The output indicates that the kmer at this position is 'AAA'.
'AAA'
```

chaos_frequency_matrix

**Description**

Generate Frequency Chaos Game Representation (FCGR) matrix

**Usage**

chaos_frequency_matrix(fasta_string: str, kmer_length: int, chaos_game_kmer_array: <built-in function array>(np.array, optional) = None, pseudo_count(bool, optional):True )

**Arguments**

| | |
|---|---|
| fasta_string | The DNA sequence in FASTA format. |
| kmer_length | The length of the kmer to consider. |
| chaos_game_kmer_array | The FCGR key matrix. Defaults to None. |
| pseudo_count | Whether to apply pseudo-counts (add 1 to each cell of the FCGR matrix) to the matrix. Defaults to True. |

**Details**

This function calculates the FCGR matrix for a given DNA sequence and kmer length using the FCGR key matrix.

**Returns**

tuple

- np.array The chaos frequency matrix representing kmer frequencies.
- np.array The FCGR key matrix used.

**Example**

1. **pseudo_count = False**

```
### Generate the frequency matrix and key array for kmers of length 3 from the given
FASTA sequence.


fasta_seq_dummy  = "ATTGCNATRATTT"
kaos_freq_matrix,kaos_key_array = kaos.chaos_frequency_matrix(fasta_string = fasta_seq_dummy,
kmer_length=3, chaos_game_kmer_array=None, pseudo_count=False)

### Display the chaos game frequency matrix.
kaos_freq_matrix

array([[1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 2., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 2.],
       [1., 1., 1., 3., 1., 1., 1., 2.]]
```

2. **pseudo_count = True**

```
fasta_seq_dummy  = "ATTGCNATRATTT"

kaos_freq_matrix,kaos_key_array = kaos.chaos_frequency_matrix(fasta_string = fasta_seq_dummy,
kmer_length=3, chaos_game_kmer_array=None, pseudo_count=True)

array([[0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 1.],
       [0., 0., 0., 2., 0., 0., 0., 1.]])
```

kaos.chaos_frequency_dictionary

**Description**

Frequency dictionary of kmers

**Usage**

chaos_frequency_dictionary(fasta_string: str, kmer_length: int, chaos_game_kmer_array: <built-in function array>(np.array, optional) = None, pseudo_count(bool, optional):True)

**Arguments**

| | |
|---|---|
| fasta_string | The DNA sequence in FASTA format. |
| kmer_length | The length of the kmer to consider. |
| chaos_game_kmer_array | The FCGR key matrix. Defaults to None. |
| pseudo_count | Whether to apply pseudo-counts (add 1 to each cell of the FCGR matrix) to the matrix. Defaults to True. |

**Details**

Calculate the frequency dictionary of kmers in the FCGR matrix.

**Returns**

| | |
|---|---|
| dictionary | A dictionary containing kmers as keys and their frequencies as values. |

**Example**

1. **pseudo_count = False**

```
### Create a frequency dictionary for kmer length 3 from a FASTA sequence using
pseudo counts as False.
### The `chaos_game_kmer_array` stores an array of kmer keys of a specific length that is
        determined by the input parameter. This array is generated using the
        `chaos_game_representation_key` function included in our package.


fasta_seq_dummy  = "ATTGCNATRATTT"
kaos.chaos_frequency_dictionary(fasta_string=fasta_seq_dummy, kmer_length=3,
chaos_game_kmer_array=chaos_game_kmer_array, pseudo_count=False)

### Output

    { 'GAG': 0.0,
      'GTC': 0.0,
      'GTG': 0.0,
      'CAA': 0.0,
      …
      'ATT': 2.0,
      'TAA': 0.0,
      'TAT': 0.0,
      'TTA': 0.0,
      'TTT': 1.0}
```

2. **pseudo_count = True**

```
### Create a frequency dictionary for kmer length 3 from a FASTA sequence using pseudo counts as
True
### The `chaos_game_kmer_array` stores an array of kmer keys of a specific length that is
        determined by the input parameter. This array is generated using the
        `chaos_game_representation_key` function included in our package.


 fasta_seq_dummy  = "ATTGCNATRATTT"
kaos.chaos_frequency_dictionary(fasta_string=fasta_seq_dummy, kmer_length=3,
chaos_game_kmer_array= chaos_game_kmer_array, pseudo_count=True)

### Output

    { 'GAG': 1.0,
      'GTC': 1.0,
      'GTG': 1.0,
      'CAA': 1.0,
      …
      'ATT': 3.0,
      'TAA': 1.0,
      'TAT': 1.0,
      'TTA': 1.0,
      'TTT': 2.0}
```

kaos.return_kmer_count_individual

**Description**

Count specific kmer

**Usage**

return_kmer_count_individual(key_name: str, fasta_content: str)

**Arguments**

| | |
|---|---|
| key_name | The kmer sequence for which the count is to be calculated. |
| fasta_content | The input DNA sequence in which the kmer count is to be calculated. |

**Details**

Calculate the count of a specific kmer in a given DNA sequence.

**Returns**

| | |
|---|---|
| count | The count of the specified kmer in the DNA sequence. |

**Example**

```
### fasta_seq_dummy =  "ATTGCNATRATTT"
### Retrieve the count of the specific kmer "ATT" from the provided FASTA content.
kaos.return_kmer_count_individual(key_name="ATT", fasta_content=fasta_seq_dummy)
### Output
2
```