Code Documentation

Annabel Page
Birkbeck, University of London
MSc Bioinformatics and Systems Biology
BioComputing II

To use the below functions, you will be required to import the business module at the top of your code.

def search_db():

This function will search the DB for query value, either accession, gene_id, location or protein_id and return a dictionary with the search outcome.

parametereter: NIL - an input prompt will appear to insert a value

return: a dictionary of all raw DB information

Example:

```
>>>business.search_db()

Please enter a valid accession, gene_id, location or protein_id: AB002059

{'accession': 'AB002059', 'gene_bp': 28984, 'gene_id': 'HP2XM', 'location': '22q11', 'dna_seq':
'ggtgaaacctcatctctactaaaaataca, 'cds': '(9106..9239,9843..9993,11889..11960,16575..16650',
'complement': 'NO', 'protein_id': 'BAA22047.1', 'product': 'Human P2XM', 'translation':
'MGSPGATTGWGLLDYKTEKYVMTRNW'}
```

def column_list(name):

This function takes a column name and will return all non-duplicated values and remove all 'None' values of the column.

parametereter name: name of a column in SQL table, column names to choose from are either accession, gene_id, location or protein_id. The search format must be in a string

returns: a list of entries from column

```
>>> business.column_list('accession')

['AB002059', 'AB003151', 'AB007166', 'AB016435', 'AB029901', 'AB049212', 'AB049213', 'AB049214',
'AB049215', 'AB050770', 'AB050771', 'AB050772', 'AB050773', 'AB050774', 'AB057594', 'AB065933',
'AB065934', 'AB067442', 'AB067443', 'AB067444', 'AB083085', 'AB086231', 'AF001361', 'AF006988']
```

def search_RE(Search_Query, RE_name):

This function takes a restriction enzyme name and will search the gene DNA sequence and return the location of the restriction enzyme.

parametereter RE_name: the name of the restriction enzyme you want to search either EcoRI, BamHI or BsuRI. The search format must be in a string.

returns: a list of the location of the searched enzyme in the sequence

```
>>> business.search_RE('AB002059', 'EcoRI')

('The location sites for EcoRI are', [2971, 11116, 12483, 20616, 21539, 21571])
```

def CDS_markup(Search_Query):

This function will take a DNA sequence, identify CDS sites and produce a marked up sequence that identifies one or more CDS sites in the DNA sequence.

parametereter: user search query. The search format must be in a string

return: marked up sequence in string format

```
>>> business.CDS_markup('AB002059')

'ggtgaaac<cacccgtccgtcccactggctaactgctgggtcgacgaggactgccccgaaggggagggaggcacacacagccac>ggtaact
gtgggctctgtcttccagtgcccccagcagggtgggggccgggctgggatcctgggtggctcctgagtgcaggccctgctcgcctctgtccctgca
tctctctttctgccaacaaccccctggctgaaggcctccccaggcctgcagagatttgaaggtctggagttcatcttttgttttctag<gtgtaaaaa
caggccagtgtgtggtgttcaatgggacccacaggacctgtgagatctggagttggtgccccgtggagagtggcgttgtgccct>cgtaagtgtc
cccacaatcccctaccccaactggcgcagggccccaggcctggcagaggctgtcacctcccttccacctgcag<gaggcccctgctggcccagg
cccagaacttcacactgttcatcaaaaacacagtcaccttcagcaagttcaacttctcta>agtaagcagagtgggtctcatctgccccaagacc
ctccttgtcccctacctcatctgacctttcccactcctcccag<gtccaatgccttggagacctgggaccccacctattttaagcactgccgctatga
accacaattcagcccctactgtcccgtgttccgcattggggacctcgtggccaaggctggagggaccttcgaggacctggcgttgct>ggtgggt
cccaagttgggggcagggttcctagagggctctgggag’
```

def CDS_DNA(Search_Query):

This function will take the marked up CDS DNA sequence and extract only the CDS information.

parametereter: user search query. The search format must be in a string.

return: list of CDS DNA sequence

```
>>> business.CDS_DNA('AB002059')

['atgggctccccaggggctacgacaggctgggggcttctggattataagacggagaagtatgtgatgaccaggaactggcgggtgggcgccct
gcagaggctgctgcagtttgggatcgtggtctatgtggtag',
'gtgggcgctcctcgccaaaaaaggctaccaggagcgggacctggaaccccagttttccatcatcaccaaactcaaaggggtttccgtcactcag
atcaaggagcttggaaaccggctgtgggatgtggccgacttcgtgaagccacctca',
'ggagagaacgtgttcttcttggtgaccaacttccttgtgacgccagcccaagttcagggcagatgcccaga']
```

def CDS_Protein(Search_Query):

Takes the list of CDS sequences and returns a list of the translated protein sequence

parameter: user search query. The search format must be in a string

return: list of translated protein sequences

```
>>> business.CDS_Protein('AB002059')
```

```
 ['MGSPGATTGWGLLDYKTEKYVMTRNWRVGALQRLLQFGIVVYVV',
'VGAPRQKRLPGAGPGTPVFHHHQTQRGFRHSDQGAWKPAVGCGRLREATS',
'GENVFFLVTNFLVTPAQVQGRCP', 'HPSVPLANCWVDEDCPEGEGGTHSH',
'V*KQASVWCSMGPTGPVRSGVGAPWRVALCP']
```

def matched_seq(Search_Query):

This function takes a search query and will match the translated protein CDS sequence with the original CDS DNA sequence.

parameter: user search query. The search format must be in a string

return: list of tuples of matched DNA and protein sequences

```
>>>business.matched_seq('AB002059')


[('atgggctccccaggggctacgacaggctgggggcttctggattataagacggagaagtatgtgatgaccaggaactggcgggtgggcgccc
tgcagaggctgctgcagtttgggatcgtggtctatgtggtag',
'MGSPGATTGWGLLDYKTEKYVMTRNWRVGALQRLLQFGIVVYVV'),
('gtgggcgctcctcgccaaaaaaggctaccaggagcgggacctggaaccccagttttccatcatcaccaaactcaaaggggtttccgtcactca
gatcaaggagcttggaaaccggctgtgggatgtggccgacttcgtgaagccacctca',
'VGAPRQKRLPGAGPGTPVFHHHQTQRGFRHSDQGAWKPAVGCGRLREATS')]
```

def codon_freq(Search_Query):

This function takes a user search query and will return a dictionary of codon usage frequencies from the DNA seq.

parameter: user search query The search format must be in a string

return: dictionary of codons and their frequencies

```
>>> business.codon_freq('AB002059')

{'ggt': 157, 'gaa': 103, 'acc': 173, 'tca': 153, 'tct': 167, 'cta': 86, 'aaa': 144, 'ata': 58, 'caa': 126, 'att': 89,
'cag': 325, 'gcg': 51, 'tgg': 279, 'ctc': 242, 'atg': 143, 'cct': 277, 'gta': 62, 'atc': 103, 'cca': 289, 'gca': 215,
'ctt': 140, 'gag': 218, 'gcc': 246, 'ggc': 224, 'gga': 183, 'aag': 131, 'ctg': 330, 'gct': 181, 'aac': 95, 'acg':
52, 'gtg': 205, 'ccc': 273, 'cgg': 70, 'tac': 66, 'taa': 80, 'tta': 67, 'agg': 277, 'cgc': 81, 'gat': 91, 'ggg': 290,
'cac': 222, 'tag': 96, 'tcc': 225, 'aat': 95, 'tgc': 193, 'agt': 108, 'aga': 181, 'tcg': 35, 'act': 119, 'aca': 166,
'gac': 120, 'gtc': 127, 'cat': 117, 'ttg': 146, 'ccg': 74, 'tga': 185, 'cgt': 34, 'tat': 61, 'tgt': 170, 'ttt': 189,
'agc': 187, 'gtt': 87, 'ttc': 145, 'cga': 37}
```

def CDS_RE(Search_Query, RE_Site):

This function will take a search query and specified RE and return the location of RE sites in the CDS sequence of the gene.

parameter: user search query, resriction enzyme name. The search format must be in a string

return: list of RE locations in seq

```
>>> business.CDS_RE('AB002059', 'EcoRI')
```

```
There are no EcoRI locations in CDS region
atgggctccccaggggctacgacaggctgggggcttctggattataagacggagaagtatgtgatgaccaggaactggcgggtgggcgccctg
cagaggctgctgcagtttgggatcgtggtctatgtggtag
```

```
There are no EcoRI locations in CDS region
gtgggcgctcctcgccaaaaaaggctaccaggagcgggacctggaaccccagttttccatcatcaccaaactcaaaggggtttccgtcactcag
atcaaggagcttggaaaccggctgtgggatgtggccgacttcgtgaagccacctca
```

def DNA_Codon():

Will take the DNA sequence across the whole chromosome and calculate codon frequencies and return a populated dictionary. This will also create a table and store the output in a new table in the database.

parameter: NIL

return: a dictionary populated with codon frequencies

```
>>> business.DNA_Codon()

Table 'dna_codon' created and data inserted successfully

{'ggt': 101125, 'gaa': 117033, 'acc': 101161, 'tca': 130375, 'tct': 146107, 'cta': 73578, 'aaa': 202743,
'ata': 86662, 'caa': 116135, 'att': 117205, 'cag': 187314, 'gcg': 32620, 'tgg': 165426, 'ctc': 153736,
'atg': 111184, 'cct': 164065, 'gta': 64281, 'atc': 85021, 'cca': 164031, 'gca': 120444, 'ctt': 123884,
'gag': 151419, 'gcc': 133098, 'ggc': 134202, 'gga': 133375, 'aag': 124090, 'ctg': 188421, 'gct': 124021,
'aac': 88349, 'acg': 26633, 'gtg': 131990, 'ccc': 147460, 'cgg': 38752, 'tac': 64397, 'taa': 95772, 'tta':
95676, 'agg': 164103, 'cgc': 32825, 'gat': 85272, 'ggg': 147205, 'cac': 130591, 'tag': 73040, 'tcc':
133526, 'aat': 116548, 'tgc': 119763, 'agt': 104995, 'aga': 144907, 'tcg': 24331, 'act': 104723, 'aca':
129991, 'gac': 78430, 'gtc': 78464, 'cat': 110988, 'ttg': 115261, 'ccg': 38843, 'tga': 130418, 'cgt':
26129, 'tat': 86360, 'tgt': 129667, 'ttt': 201777, 'agc': 123610, 'gtt': 87173, 'ttc': 117788, 'cga': 23926,
'gnt': 2, 'ntc': 2, 'aan': 3, 'nat': 3, 'anc': 4, 'ana': 2, 'gan': 2, 'ant': 1, 'can': 1, 'tcn': 1, 'ann': 2, 'nnn': 98,
'nng': 1, 'nag': 3, 'ngg': 6, 'gay': 1, 'tsg': 1, 'ccs': 1, 'cma': 1, 'gcy': 1, 'rag': 1, 'ygg': 1, 'gkt': 1, 'ktt': 1,
'kca': 1, 'cna': 3, 'cnt': 3, 'ggn': 4, 'nnt': 1, 'ccr': 1, 'ggy': 1, 'gsc': 1, 'gyt': 1, 'ytg': 1, 'nca': 2, 'nct': 4, 'trt':
1, 'gcd': 1, 'trr': 1, 'ddn': 1, 'ats': 1, 'yac': 1, 'cng': 1, 'cnn': 1, 'ngc': 1, 'ncc': 4, 'ncn': 1, 'ncg': 1, 'ntg': 2,
'ccn': 3, 'nnc': 1, 'ang': 2, 'tgn': 1, 'agn': 1, 'nac': 4, 'cgn': 1, 'gnc': 2, 'gcn': 1, 'ttn': 1, 'nna': 1, 'nta': 1,
'tan': 1, 'gnn': 1, 'gtM': 1, 'app': 1, 'osi': 1, 'tio': 1, 'qte': 1, 'gng': 1, 'tty': 1, 'mca': 1, 'sac': 1}
```

def get_DNA_codon():

This function will return the codon frequency for the whole chromosome DNA sequence that has been calculated and stored in the database.

parameter: NIL

return: list of tuples containing the codon and frequency

```
>>> business.get_DNA_codon()

[(202743, 'aaa'), (88349, 'aac'), (124090, 'aag'), (3, 'aan'), (116548, 'aat'), (129991, 'aca'), (101161,
'acc'), (26633, 'acg'), (104723, 'act'), (144907, 'aga'), (123610, 'agc'), (164103, 'agg'), (1, 'agn'),
(104995, 'agt'), (2, 'ana'), (4, 'anc'), (2, 'ang'), (2, 'ann'), (1, 'ant'), (1, 'app'), (86662, 'ata'), (85021,
'atc'), (111184, 'atg'), (1, 'ats'), (117205, 'att'), (116135, 'caa'), (130591, 'cac'), (187314, 'cag'), (1,
'can'), (110988, 'cat'), (164031, 'cca'), (147460, 'ccc'), (38843, 'ccg'), (3, 'ccn'), (1, 'ccr'), (1, 'ccs'),
(164065, 'cct'), (23926, 'cga'), (32825, 'cgc'), (38752, 'cgg'), (1, 'cgn'), (26129, 'cgt'), (1, 'cma'), (3,
'cna'), (1, 'cng'), (1, 'cnn'), (3, 'cnt'), (73578, 'cta'), (153736, 'ctc'), (188421, 'ctg'), (123884, 'ctt'), (1,
'ddn'), (117033, 'gaa'), (78430, 'gac'), (151419, 'gag'), (2, 'gan'), (85272, 'gat'), (1, 'gay'), (120444,
'gca'), (133098, 'gcc'), (1, 'gcd'), (32620, 'gcg'), (1, 'gcn'), (124021, 'gct'), (1, 'gcy'), (133375, 'gga'),
(134202, 'ggc'), (147205, 'ggg'), (4, 'ggn'), (101125, 'ggt'), (1, 'ggy'), (1, 'gkt'), (2, 'gnc'), (1, 'gng'), (1,
'gnn'), (2, 'gnt'), (1, 'gsc'), (1, 'gtM'), (64281, 'gta'), (78464, 'gtc'), (131990, 'gtg'), (87173, 'gtt'), (1,
'gyt'), (1, 'kca'), (1, 'ktt'), (1, 'mca'), (4, 'nac'), (3, 'nag'), (3, 'nat'), (2, 'nca'), (4, 'ncc'), (1, 'ncg'), (1,
'ncn'), (4, 'nct'), (1, 'ngc'), (6, 'ngg'), (1, 'nna'), (1, 'nnc'), (1, 'nng'), (98, 'nnn'), (1, 'nnt'), (1, 'nta'), (2,
'ntc'), (2, 'ntg'), (1, 'osi'), (1, 'qte'), (1, 'rag'), (1, 'sac'), (95772, 'taa'), (64397, 'tac'), (73040, 'tag'), (1,
'tan'), (86360, 'tat'), (130375, 'tca'), (133526, 'tcc'), (24331, 'tcg'), (1, 'tcn'), (146107, 'tct'), (130418,
'tga'), (119763, 'tgc'), (165426, 'tgg'), (1, 'tgn'), (129667, 'tgt'), (1, 'tio'), (1, 'trr'), (1, 'trt'), (1, 'tsg'),
(95676, 'tta'), (117788, 'ttc'), (115261, 'ttg'), (1, 'ttn'), (201777, 'ttt'), (1, 'tty'), (1, 'yac'), (1, 'ygg'), (1,
'ytg')]
```