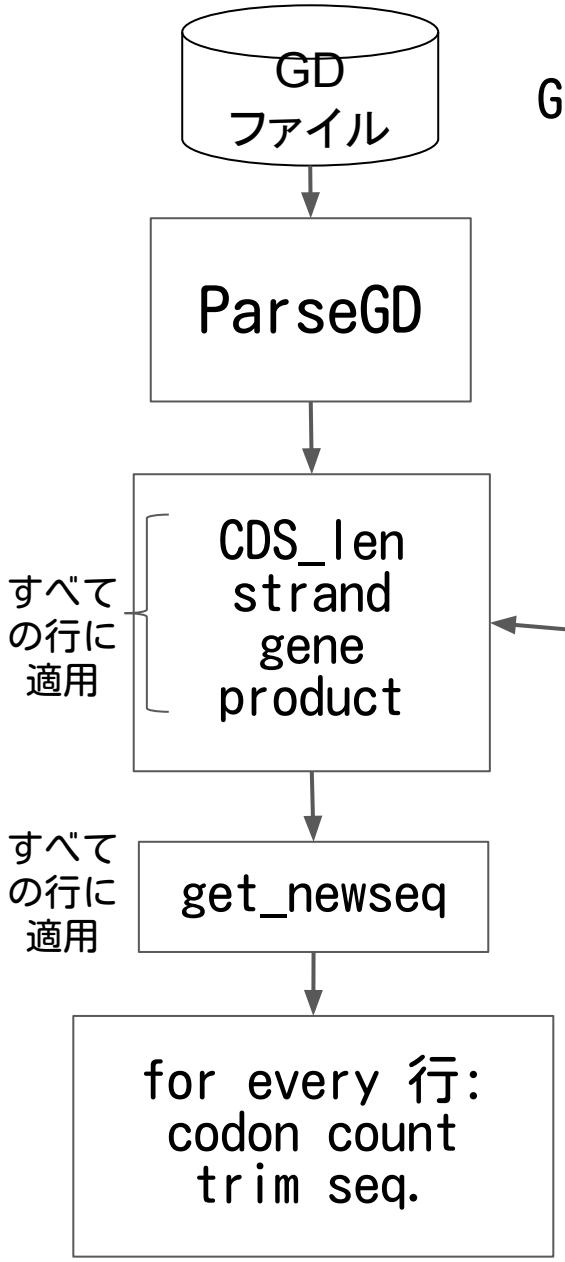
Breseq出力の ＧＤ(Gene Differenc) ファイルの処理

GDファイルにCDS情報を追加する

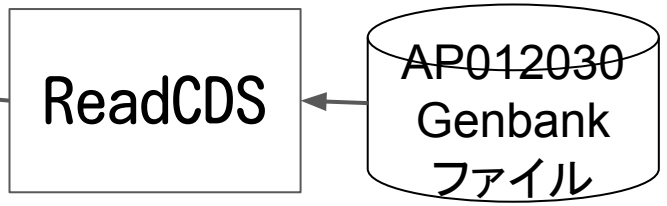
2020-01-06 山内長承

処理の流れ



GDファイルを読んでDataFrameに変換

B	C	D	E	F	G	H	I	J	K
type	id	parentID	pos	size	ewbas	inspos	refbase	epeat_r	strand
DEL	1	14	311	1			T		
SNP	2	15	268336		G		A		
INS	3	24	859013	1	T				
INS	4	25	1164301	1	A				
INS	5	38	1924115	1	C				
SNP	6	39	2594895		T		G		
INS	7	397	2640492	91	CGGATGCGGCGTGAACGCCTTATCCGAC				



ReadCDSの出力

pos	len	strand	gene	product	function	codon_start	protein_id	original_seq	reversed_seq
189	66	1	['thrL']	['thr operoi	0	['1']	['BAJ41857.	ATGAAAC(	ATGAAAC(
337	2463	1	['thrA']	['bifunctior	['Aspartokir	['1']	['BAJ41858.	ATGCGAG	ATGCGAG
2801	933	1	['thrB']	['homoserir	['Homoseri	['1']	['BAJ41859.	ATGGTTA/	ATGGTTA/
3734	1287	1	['thrC']	['threonine	['Threonine	['1']	['BAJ41860.	ATGAAAC1	ATGAAAC1
5234	297	1	['yaaX']	['predicted	0	['1']	['BAJ41861.	GTGAAAA/	GTGAAAA/

ReadCDS

GenBank(GB)ファイルを読んでDataFrameに変換

- GBフォーマット　⇒　BioPythonのSeq-IOで読める
  必要なフィールドを取出して、DFに書込む

```
import pandas as pd
from Bio import SeqIO
from Bio import SeqFeature
def ReadCDS(infile):
    CDStable = pd.DataFrame(columns=['pos', 'len', 'strand', 'gene', 'product',
     'function', 'codon_start', 'protein_id', 'original_seq', 'reversed_seq'])
    for rec in SeqIO.parse(infile, "genbank"):
        for feat in rec.features:
            if feat.type not in ['CDS']:
                continue
            CDS_row = pd.Series(['0']*len(CDStable.columns), index=CDStable.columns)
            CDS_row['pos'] = feat.location.start
            CDS_row['len'] = feat.location.end - feat.location.start
            CDS_row['strand'] = feat.location.strand
            feat_seq = feat.location.extract(rec.seq)
            CDS_row['original_seq'] = feat_seq  # すでにstrandに合わせてrev_compされてい
る
            for u in feat.qualifiers.keys():
                CDS_row[u] = feat.qualifiers[u]
            CDStable = CDStable.append(CDS_row, ignore_index=True)
    return(CDStable)
```

CDS_posを計算する

```
def getCDSpos(u):    # uはSNPなどの位置
    u = u-1          # u is 1-base, CDStable is 0-base
    cds = CDStable[ (CDStable['pos']<=u) & (u<(CDStable['pos']+CDStable['len'])) ]
    if len(cds)==1:
        return(int(cds['pos']))
    elif len(cds)>=2:
        return(int(cds['pos'].to_list()[0]))
    else:
        return(-1)
```

戻り値は
pos欄の値

CDStableの行の中で、uの値が
pos欄の値と(pos欄+len欄)の値の
間にあるような行を選ぶ

pos	len	strand	gene	product	function	codon_start	protein_id	original_seq	reversed_seq
189	66	1	['thrL']	['thr opero	0	['1']	['BAJ41857.	ATGAAAC(	ATGAAAC(
337	2463	1	['thrA']	['bifunctior	['Aspartoki	['1']	['BAJ41858.	ATGCGAG	ATGCGAG
2801	933	1	['thrB']	['homoserir	['Homoseri	['1']	['BAJ41859.	ATGGTTA/	ATGGTTA
3734	1287	1	['thrC']	['threonine	['Threonine	['1']	['BAJ41860.	ATGAAAC1	ATGAAAC
5234	297	1	['yaaX']	['predicted	0	['1']	['BAJ41861.	GTGAAAA/	GTGAAAA

```
outdf['CDS_pos'] = outdf['pos'].apply(getCDSpos)
```

outdfのすべての行について、outdf['pos']から、それを含むCDSの位置を求めて
outdf['CDS_pos']に書く

CDS_lenなどを書き写す

```
outdf['CDS_len'] = outdf['CDS_pos'].apply(\
    (lambda x: (CDStable[CDStable['pos']==x]['len'].iloc[0]) if x>=0 else ''))
```

lambda x: ~~~~~~ は

outdf[‘CDS_pos’]の１つの要素をxとして受け取って、

もしx>=0ならば

CDStable上で’pos’欄がxと等しい行を見つけ、その行の[‘len’]欄の値（iloc[0]）を返し

そうでなければ’’を返す関数

この関数をoutdf[‘CDS_pos’]（これはSeries）のすべての要素に適用して、同じ要素数のSeriesを作る（applyの機能）

できたSeriesを、outdf[‘CDS_len’]に代入（欄が存在しないので新しく追加）する

```
outdf['strand'] = outdf['CDS_pos'].apply(\
    (lambda x: (CDStable[CDStable['pos']==x]['strand'].iloc[0]) if x>=0 else ''))
outdf['gene'] = outdf['CDS_pos'].apply(\
    (lambda x: (CDStable[CDStable['pos']==x]['gene'].iloc[0]) if x>=0 else ''))
outdf['product'] = outdf['CDS_pos'].apply(\
    (lambda x: (CDStable[CDStable['pos']==x]['product'].iloc[0]) if x>=0 else \
         'intergenic' if x==-1 else ''))
outdf['original_seq'] = outdf['CDS_pos'].apply(\
    (lambda x: (CDStable[CDStable['pos']==x]['original_seq'].iloc[0]) if x>=0 \
          else ''))
```

SNPなどによる変更を加えた新しい配列を作る

```
def get_newseq(r):
    #target_pos = int(r['pos'])-int(r['CDS_pos'])-1   # strandに関係なく絶対位置
    # In case CDS seq contains skips, calculate num_seqskips and adjust target_pos
    if r['type'] in ['SNP', 'INS', 'DEL'] and r['strand']==-1:
        target_pos = len(r['original_seq']) - ( int(r['pos'])-int(r['CDS_pos']) )
    else:
        target_pos = int(r['pos'])-int(r['CDS_pos'])-1
    if r['CDS_pos']<0:
        return(Seq(''))
    if r['type'] == 'SNP':
        if (r['refbase'] not in ['A','G','C','T']) or ¥
        ((r['original_seq'])[target_pos]!=RevCompSeq(r['strand'],Seq(r['refbase']))):
            print('Not modifiableS', 'pos=', r['pos'], 'target_pos', target_pos, ¥
                  'refbase=', r['refbase'], ¥
                  'newbase=', r['newbase'], 'strand', r['strand'], ¥
                  "(r['original_seq'])[target_pos]=", (r['original_seq'])[target_pos], ¥
                 (r['original_seq'])[target_pos-3:target_pos+3])
        else:
            return( (r['original_seq'])[:target_pos] + ¥
                    str(RevCompSeq(r['strand'], Seq(r['newbase']))) + ¥
                   (r['original_seq'])[target_pos+1:] )
```

SNPなどによる変更を加えた新しい配列を作る

```
    elif r.loc['type'] == 'INS':
        return( (r['original_seq'])[:target_pos] + ¥
            r['newbase'] + (r['original_seq'])[target_pos:] )
    elif r.loc['type'] == 'DEL':
        if (r['original_seq'])[target_pos] != RevCompSeq(r['strand'],
(r['refbase'])[0:1]):
            print('Not modifiableD', 'pos', r['pos'], 'refbase', (r['refbase'])[0:0],¥
                  (r['original_seq'])[target_pos])
        else:
            return( (r['original_seq'])[:target_pos] + ¥
                (r['original_seq'])[target_pos+r['size']:] )
    elif r.loc['type'] == 'SUB':
        return( (r['original_seq'])[:target_pos] + r['newbase'] +¥
            (r['original_seq'])[target_pos+1+r['size']:] )
    elif r.loc['type'] in ['MOB', 'AMP', 'CON', 'INV']:
        print('Not processed type', r['type'])
        return(Seq(''))
```

```
dfx = outdf.copy()  #
dfx['new_seq'] = dfx.apply(get_newseq, axis=1)
```

それぞれの配列のcodon数の計算

```
    dfx['trimmed_seq'] = pd.Series([Seq('')]*len(dfx))    空の欄を作っておく

    for i, r in dfx.iterrows():    Dataframe dfx の行ごとに繰返しループ
        print('======', i, '========================================')
        print('======', i, '========================================', file=logfp)
        if r.loc['CDS_pos']==-1:    posに対応するCDSが無ければ飛ばす
            continue

        # 元配列に対してcodon数をカウント
        original_seq_rc = r.loc['original_seq']
        orig_codon = [str(original_seq_rc)[i : i+3] ¥
                      for i in range(0, len(original_seq_rc), 3)]  codonのリスト
        codonlist = CodonsList     codon名のリスト
        orig_codon_count = codon_count(r.loc['index'], orig_codon)  codon_countで計数
        for cdn in codonlist:
            dfx.loc[i, 'o'+cdn] = orig_codon_count[cdn]  dfxの oXXX の欄にコドン数書込
```

```
def codon_count(id, codons):
    codon_count= collections.Counter(codons)
    return codon_count
```

それぞれの配列のcodon数の計算

```
# 変異後の配列new_seqに対してcodon数をカウント
new_seq_rc = r.loc['new_seq']
if (len(new_seq_rc)%3) != 0:
    print('%s length %d not multiple of 3' % (i, len(new_seq_rc)))
new_codon = [str(new_seq_rc)[i:i+3] for i in range(0,len(new_seq_rc)//3*3,3)]
if len(new_codon)>0 and new_codon[0] not in start_codons:
    print('%s starts with %s which is not in start_codons' % (i, new_codon[0]))
if len(new_codon)>0 and new_codon[-1] not in stop_codons:
    print('%s ends with %s which is not in end_codons' % (i, new_codon[-1]))
 new_codon_count = codon_count(r.loc['index'], new_codon)
for cdn in codonlist:
    dfx.loc[i, 'n'+cdn] = new_codon_count[cdn]

if dfx.loc[i, 'type'] == 'SNP':
    new_codon_diff = new_codon_count - orig_codon_count プラスの差分
    orig_codon_diff = orig_codon_count - new_codon_count  マイナスの差分

    dfx.loc[i, 'old_codon'] = orig_codon_diff    プラスになっているcodon名
    dfx.loc[i, 'new_codon'] = new_codon_diff    マイナスになっているcodon名
```

配列のstop codonによるトリミング

```
num_stop_codons = 0
for sc in stop_codons:
    num_stop_codons += dfx.loc[i, 'n'+sc]
dfx.loc[i, 'num_stop_codons'] = num_stop_codons    stop codonの数
if num_stop_codons > 1:
    trimmed_codon_list = new_codon  前に求めた変異後配列のcodonのリスト
    for u in stop_codons:   stop codonは複数あるのでそれぞれについて
        for j in range(len(trimmed_codon_list)):  trimmed_codon_listを先頭から
            if trimmed_codon_list[j]==u:
                trimmed_codon_list = trimmed_codon_list[0:j+1]
                   # stop_codonを含めるのでjまでで切る
                break     最初に見つかったところで切ってしまって脱出

    trimmed_seq = Seq(''.join(trimmed_codon_list))   codonリストを結合して配列
    trimmed_codon=¥                              trimmed_seqのcodonリストを作り直す
       [str(trimmed_seq)[i:i+3] for i in range(0,len(trimmed_seq), 3)]
    trimmed_codon_count = codon_count(r.loc['index'], trimmed_codon)  数える
    for cdn in codonlist:
        dfx.loc[i, 't'+cdn] = trimmed_codon_count[cdn]
```

最後に列順を適宜入れ替えて、Excelファイルに出力する（略）