summarize_probe_results

May 13, 2020

0.0.1 Summarize probe results

- Analyze individual qPCR plate data with the output and the template, return ddCt averages from technical reps
- Combine into entire experiment df, get fold change and std fold change of the three replicates
- Combine the qPCR data with probe thermodynamic properties

[1]:
```
#Imports
import sys
import pandas as pd
import matplotlib.pyplot as plt
import os
import gffutils
import seaborn as sns
import numpy as np
import subprocess
from Bio import SeqIO
import HTSeq
import primer3
from Bio.SeqUtils import MeltingTemp as mt
import primer3
from Bio.Seq import Seq

sys.path.append('../scripts/')
from plot_helpers import *
import analyze_qpcr_plate
sys.path.append(os.path.join(probe_designer_dir, 'scripts'))
import screen_kmers
import choose_probes

%matplotlib inline
%load_ext autoreload
%autoreload 2
```

0.0.2 Part I: Summarize the qPCR depletion data

[2]:
```
outdir = '../figures/F1/'
os.makedirs(outdir, exist_ok = True)
```

[3]:
```
qpcr_dir = os.path.join(results_dir, 'qPCR_data')

data = [
'190417_rep1_A/
 ↪20190417_171207_CT003077__QPCRBIOSMALQuantificationPlateViewResults.xlsx',
'190418_rep2_A/
 ↪20190418_154147_CT003077__QPCRBIOSMALQuantificationPlateViewResults.xlsx',
'190425_rep3_A/
 ↪20190425_152906_CT003077__QPCRBIOSMALQuantificationPlateViewResults.xlsx',
'190418_rep1_B/
 ↪20190418_165533_CT003077__QPCRBIOSMALQuantificationPlateViewResults.xlsx',
'190419_rep2_B/
 ↪20190419_130040_CT003077__QPCRBIOSMALQuantificationPlateViewResults.xlsx',
'190426_rep3_B/
 ↪20190426_125032_CT003077__QPCRBIOSMALQuantificationPlateViewResults.xlsx',
'190422_rep1_C/
 ↪20190422_143153_CT003077__QPCRBIOSMALQuantificationPlateViewResults.xlsx',
'190419_rep2_C/
 ↪20190419_153322_CT003077__QPCRBIOSMALQuantificationPlateViewResults.xlsx',
'190422_rep3_C/
 ↪20190422_161800_CT003077__QPCRBIOSMALQuantificationPlateViewResults.xlsx',
'190625_rep1_D/
 ↪20190625_133705_CT003077__QPCRBIOSMALQuantificationPlateViewResults.xlsx',
'190626_rep2_D/
 ↪20190626_134856_CT003077__QPCRBIOSMALQuantificationPlateViewResults.xlsx',
'190628_rep3_D/
 ↪20190628_125833_CT003077__QPCRBIOSMALQuantificationPlateViewResults.xlsx'
]

templates = [
'190417_rep1_A/qPCR_analysis_template_rep1_A.xlsx',
'190418_rep2_A/qPCR_analysis_template_rep2_A.xlsx',
'190425_rep3_A/qPCR_analysis_template_rep3_A.xlsx',
'190418_rep1_B/qPCR_analysis_template_rep1_B.xlsx',
'190419_rep2_B/qPCR_analysis_template_rep2_B.xlsx',
'190426_rep3_B/qPCR_analysis_template_rep3_B.xlsx',
'190422_rep1_C/qPCR_analysis_template_rep1_C.xlsx',
'190419_rep2_C/qPCR_analysis_template_rep2_C.xlsx',
'190422_rep3_C/qPCR_analysis_template_rep3_C.xlsx',
'190625_rep1_D/qPCR_analysis_template_rep1_D.xlsx',
'190626_rep2_D/qPCR_analysis_template_rep2_D.xlsx',
'190628_rep3_D/qPCR_analysis_template_rep3_D.xlsx'
]

exps = {'data': [os.path.join(qpcr_dir, i) for i in data],
       'templates': [os.path.join(qpcr_dir, i) for i in templates]}
```

[4]:
```
#Get the probe sequences, and add to the df
probe_seqs = os.path.join(qpcr_dir, 'probe_seqs.csv')
seq_df = pd.read_csv(probe_seqs, index_col = 'probe_name')
```

[5]:
```
#Get all the data from each replicate
#https://stackoverflow.com/questions/19078325/
 ↪naming-returned-columns-in-pandas-aggregate-function
long_probe_ids = ['D2', 'D6', 'D13', 'D17', 'D19']
df_list = []
for i in range(0, len(exps['data'])):
    #For set D, remove B2 and B15 because these were included as plate controls
    if i > 8:
        df_list.append(analyze_qpcr_plate.main(exps['data'][i],␣
 ↪exps['templates'][i], 'act5c', drop_samples = ['B2', 'B15']))
    else:
        df_list.append(analyze_qpcr_plate.main(exps['data'][i],␣
 ↪exps['templates'][i], 'act5c'))
df = pd.concat(df_list)
qpcr_df = df.groupby(['primer', 'sample']).agg(mean_frac_remaining =␣
 ↪('fold_change', np.mean), std_frac_remaining = ('fold_change', np.std))
qpcr_df.index.rename('probe_name', level = 'sample', inplace = True)
qpcr_df = qpcr_df.loc[qpcr_df.index.get_level_values('probe_name').map(lambda x:
 ↪ (x.startswith('B') or x in long_probe_ids))].copy()
#Retain all the values, including the ddCt values, for stats later
full_df = df.loc[df.index.get_level_values('sample').map(lambda x: (x.
 ↪startswith('B') or x in long_probe_ids))].copy()
full_df_num = pd.merge(full_df.reset_index('sample', drop = False),␣
 ↪seq_df[['probe_num']], left_on = 'sample', right_index = True)
```

[6]:
```
#Add the Tm, sequence, and structure values to the set of tested 18S probes
#merge on name, but then switch to using probe number
probe_18S_df = pd.merge(qpcr_df, seq_df[['probe_num', 'sequence']], left_index␣
 ↪= True, right_index = True, how = 'right')
probe_18S_df['length'] = probe_18S_df['sequence'].apply(lambda x: len(x))
probe_18S_df['target_seq'] = probe_18S_df['sequence'].apply(lambda x: Seq(x).
 ↪reverse_complement())
probe_18S_df['Tm'] = probe_18S_df['target_seq'].apply(lambda x: mt.Tm_NN(x.
 ↪transcribe(), nn_table = mt.R_DNA_NN1, Na = 300, saltcorr = 4, dnac1 = 250,␣
 ↪dnac2 = 0))
```

```
#to get the params for any probes, even those that did not pass filters, for␣
↪analysis
default_rules = ['GC_content_rule', '4xA_stack_rule', '4xC_stack_rule',␣
↪'any5_rule']
probe_18S_df = screen_kmers.sequence_composition_filter(probe_18S_df, 0.4, 0.6,␣
↪default_rules, filter = False)
probe_18S_df = screen_kmers.structure_filter(probe_18S_df, -3, -10, 300, filter␣
↪= False)
```

[7]:
```
#Annotate id labels with categories
pool1_ids = range(1, 21)
pool2_ids = range(21, 31)
long_ids = range(31, 36)
lowtm_pool1_ids = range(1, 12)

probe_18S_df['length_category'] = probe_18S_df['probe_num'].map(lambda x:␣
↪'~50mer' if x in long_ids else '~30mer')
probe_18S_df['tm_category'] = probe_18S_df['probe_num'].map(lambda x: 'high Tm'␣
↪if x in pool2_ids else ('low Tm' if x in lowtm_pool1_ids else np.nan))

#write the table with the ddCt values
full_df_num['length_category'] = full_df_num['probe_num'].map(lambda x:␣
↪'~50mer' if x in long_ids else '~30mer')
full_df_num['tm_category'] = full_df_num['probe_num'].map(lambda x: 'high Tm'␣
↪if x in pool2_ids else ('low Tm' if x in lowtm_pool1_ids else np.nan))
full_df_num.to_csv(os.path.join(outdir, 'full_probe_18S_summary.csv'))
```

0.0.3 Part II: Use the sequences to annotate the target positions on the 18S and calculate thermodynamic metrics

[8]:
```
#Build the rRNA metrics for mapping the probes to the sequence
#Mapping to check the position is necessary because the PDB-derived sequences␣
↪have some indels relative to the consensus
struct_18S = os.path.join(results_dir,'other_data/structure_18S_frompdb.fasta')
struct_28S = os.path.join(results_dir,'other_data/structure_28S_frompdb.fasta')

cons_18S = os.path.join(results_dir, 'probe_design_results/dmel_200504/
↪target_sequences/consensus/18S.fa')
cons_28S = os.path.join(results_dir, 'probe_design_results/dmel_200504/
↪target_sequences/consensus/28S.fa')

fasta_dict = {'struct_small': struct_18S, 'struct_large': struct_28S,␣
↪'cons_small': cons_18S, 'cons_large': cons_28S}

struct_outdir = os.path.join(outdir, 'data_structure_mapping')
```

```
os.makedirs(struct_outdir, exist_ok = True)
to_build = ['struct_small', 'cons_small']

for i in to_build:
    fasta_file = os.path.join(struct_outdir, '%s.fasta' % i)
    this_seq = next(SeqIO.parse(fasta_dict[i], "fasta"))
    this_seq.seq = this_seq.seq.back_transcribe()
    SeqIO.write(this_seq, fasta_file, "fasta")
    subprocess.run(['bowtie2-build', fasta_file, os.path.join(struct_outdir, i)])
```

[9]:
```
#Align with bowtie to the consensus sequence or the structural sequence
probe_string = ','.join(probe_18S_df.reset_index()[['probe_num', 'sequence']].
 ↪sort_values('probe_num')['sequence'].values)
to_align = ['struct_small', 'cons_small']

seed_len = 8
for i in to_align:
    index_base = os.path.join(struct_outdir, i)
    sam_outfile = os.path.join(struct_outdir, '%s_aligned_seqs.sam' % i)
    subprocess.run(['bowtie2', '-x', index_base, '-c', probe_string, '-L',␣
 ↪str(seed_len), '-S', sam_outfile])
```

[10]:
```
#Parse the alignment files and add the consensus sequence or structural sequence␣
 ↪indices (convert to 1-based)
def add_aln_pos(sam_file, start_name, end_name):
    pos_dict = {}
    sam_reader = HTSeq.SAM_Reader(sam_file)
    for aln in sam_reader:
        if aln.aligned:
            #convert positions back to 1-based, end inclusive for pymol
            probe_num = int(aln.read.name) + 1
            pos_dict[probe_num] = {}
            pos_dict[probe_num][start_name] = aln.iv.start + 1
            pos_dict[probe_num][end_name] = aln.iv.end

    pos_df = pd.DataFrame.from_dict(pos_dict, orient = 'index')
    return pos_df

cons_aln = os.path.join(struct_outdir, '%s_aligned_seqs.sam' % 'cons_small')
struc_aln = os.path.join(struct_outdir, '%s_aligned_seqs.sam' % 'struct_small')
cons_df = add_aln_pos(cons_aln, 'consensus_start', 'consensus_end')
probe_18S_df = pd.merge(probe_18S_df, cons_df, left_on = 'probe_num',␣
 ↪right_index = True)
struct_df = add_aln_pos(cons_aln, 'structure_start', 'structure_end')
probe_18S_df = pd.merge(probe_18S_df, struct_df, left_on = 'probe_num',␣
 ↪right_index = True)
```

```
[11]: #write table S1
cols = ['probe_num', 'probe_name', 'mean_frac_remaining', 'std_frac_remaining',␣
 ↪'consensus_start', 'consensus_end', 'structure_start', 'structure_end',␣
 ↪'sequence', 'length', 'Tm', 'GC_content', 'hairpin_dG', 'homodimer_dG']
probe_18S_df.reset_index().sort_values(by = 'probe_num')[cols].to_csv(os.path.
 ↪join(outdir, 'TableS1_18S_candidate_properties.csv'), index = False)
```