ToS_presentation_v2

January 17, 2022

1 Converting reactome complex with sets tree to complex of proteins

1.0.1 Define classes to construct a Reactome tree

[ ]:
```python
import anytree
from anytree import NodeMixin
import copy

class ReactomeDatabaseObject(object):
    def __init__(self, dbId):
        self.dbId = dbId

class ReactomePhysicalEntity(ReactomeDatabaseObject):
    def __init__(self, dbId):
        super().__init__(self, dbId)

class Entity(ReactomePhysicalEntity, NodeMixin):
    def __init__(self, dbId, parent=None):
        super(ReactomePhysicalEntity, self).__init__(dbId)
        self.parent = parent

class ReactomeComplex(ReactomePhysicalEntity, NodeMixin):
    def __init__(self, dbId, parent=None, children=None):
        super(ReactomePhysicalEntity, self).__init__(dbId)
        self.parent = parent
        if children:
            self.children = children

class ReactomeEntitySet(ReactomePhysicalEntity, NodeMixin):
    def __init__(self, dbId, parent=None, children=None):
        super(ReactomePhysicalEntity, self).__init__(dbId)
        self.parent = parent
        if children:
            self.children = children
```

1.0.2 Define Functions to expand the Reactome tree into a complex of proteins

[ ]:
```
def get_lol_names(list_of_lists):
    return [[value.dbId for value in sublist] for sublist in list_of_lists]

def get_l_names(sublist):
    return [value.dbId for value in sublist]

def make_concrete(physical_entity):
    list_of_lists = [[physical_entity]]
    print('The starting list_of_lists is {}'.
 ↪format(get_lol_names(list_of_lists)))
    while not is_concrete(list_of_lists):
        print('entered while loop')
        expand_complex(list_of_lists)
        expand_set(list_of_lists)
    print('Concrete list_of_lists is {}'.format(get_lol_names(list_of_lists)))
    print([list_of_lists.count(item) for item in list_of_lists])

def is_concrete(list_of_lists):
    flag = True
    for sublist in list_of_lists:
        for element in sublist:
            if type(element) != Entity:
                flag = False
                return flag
    return flag

def expand_complex(list_of_lists):
    assert list_of_lists[0][0]
    if type(list_of_lists[0][0]) == list:
        raise ValueError('list_of_lists must be two dimensional')
    print('    expanding complexes in {}'.format(get_lol_names(list_of_lists)))
    for sublist in list_of_lists:
        print('        Working on sublist {}'.format(get_l_names(sublist)))
        elements_to_remove = []
        for element in sublist:
            print('            checking element {}'.format(element.dbId))
            if isinstance(element, ReactomeComplex):
                elements_to_remove.append(element)
                print('                Expanding {}'.format(element.dbId))
                element_idx = sublist.index(element)
                # sublist.remove(element)
                i = 1
                for child in element.children:
                    print('                    inserting child {}'.format(child.
 ↪dbId))
```

```
                    sublist.insert(element_idx+i, child)
                    i=+1
                print('                Expanded sublist to {}'.
↪format(get_l_names(sublist)))
        for element in elements_to_remove:
            sublist.remove(element)
    print('    expanded complexes to yield {}'.
↪format(get_lol_names(list_of_lists)))
    print('    ---------------')


def expand_set(list_of_lists):
    # Checking if list_of_lists is a list of lists.
    assert list_of_lists[0][0]
    if type(list_of_lists[0][0]) == list:
        raise ValueError('list_of_lists must be two dimensional')
    print('    expanding sets in {}'.format(get_lol_names(list_of_lists)))

    sublists_to_remove = []
    for sublist in list_of_lists:
        if any(isinstance(element, ReactomeEntitySet) for element in sublist):
            sublists_to_remove.append(sublist)
            for idx in range(len(sublist)):
                if isinstance(sublist[idx], ReactomeEntitySet):
                    for child in sublist[idx].children:
                        new_sublist = copy.copy(sublist)
                        new_sublist[idx] = child
                        list_of_lists.append(new_sublist)
                    break
    for sublist_to_remove in sublists_to_remove:
        list_of_lists.remove(sublist_to_remove)

    print('    expanded set to yield {}'.format(get_lol_names(list_of_lists)))
    print('    ---------------')
```

1.0.3 Create Reactome sample tree and expand

[ ]:
```
root = ReactomeComplex('root')
'''subunit1 = Entity('subunit1', parent = root)
subunit2 = ReactomeEntitySet('subunit2', parent = root)
alternative1 = Entity('alternative1', parent=subunit2)
alternative2 = Entity('alternative2', parent=subunit2)'''

P1c = Entity('P1c', parent=root)
C1c = ReactomeComplex('C1c', parent=root)
S1c = ReactomeEntitySet('S1c', parent=root)
```

```
P2c1 = Entity('P2c1', parent=C1c)
C2c1 = ReactomeEntitySet('C2c1', parent=C1c)
S2s1 = ReactomeComplex('S2s1', parent=C1c)
P2s1 = Entity('P2s2', parent=S1c)
C2s1 = ReactomeEntitySet('C2s1', parent=S1c)

P_I = Entity('P_I', parent=C2c1)
P_II = Entity('P_II', parent=C2c1)
P_III = Entity('P_III', parent=S2s1)
P_IV = Entity('P_IV', parent=S2s1)
P_V = Entity('P_V', parent=C2s1)
P_VI = Entity('P_VI', parent=C2s1)

make_concrete(root)
```

```
The starting list_of_lists is [['root']]
entered while loop
    expanding complexes in [['root']]
        Working on sublist ['root']
            checking element root
                Expanding root
                    inserting child P1c
                    inserting child C1c
                    inserting child S1c
                Expanded sublist to ['root', 'S1c', 'C1c', 'P1c']
            checking element S1c
            checking element C1c
                Expanding C1c
                    inserting child P2c1
                    inserting child C2c1
                    inserting child S2s1
                Expanded sublist to ['root', 'S1c', 'C1c', 'S2s1', 'C2c1',
'P2c1', 'P1c']
            checking element S2s1
                Expanding S2s1
                    inserting child P_III
                    inserting child P_IV
                Expanded sublist to ['root', 'S1c', 'C1c', 'S2s1', 'P_IV',
'P_III', 'C2c1', 'P2c1', 'P1c']
            checking element P_IV
            checking element P_III
            checking element C2c1
            checking element P2c1
            checking element P1c
    expanded complexes to yield [['S1c', 'P_IV', 'P_III', 'C2c1', 'P2c1',
'P1c']]
```

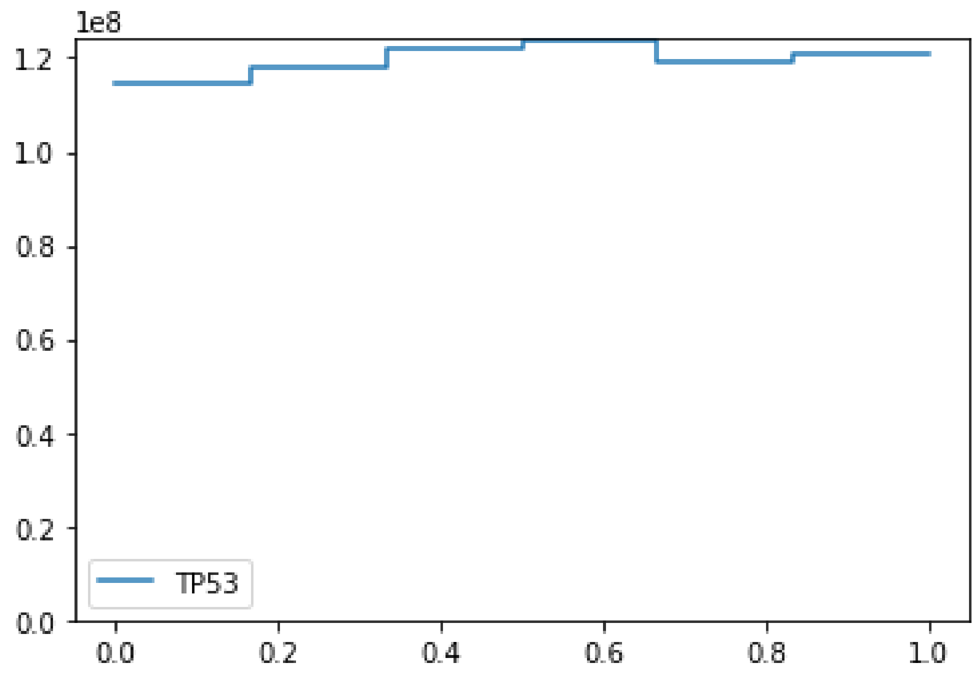
```
    ---------------
    expanding sets in [['S1c', 'P_IV', 'P_III', 'C2c1', 'P2c1', 'P1c']]
    expanded set to yield [['P2s2', 'P_IV', 'P_III', 'P_I', 'P2c1', 'P1c'],
['P2s2', 'P_IV', 'P_III', 'P_II', 'P2c1', 'P1c'], ['P_V', 'P_IV', 'P_III',
'P_I', 'P2c1', 'P1c'], ['P_V', 'P_IV', 'P_III', 'P_II', 'P2c1', 'P1c'], ['P_VI',
'P_IV', 'P_III', 'P_I', 'P2c1', 'P1c'], ['P_VI', 'P_IV', 'P_III', 'P_II',
'P2c1', 'P1c']]
    ---------------
Concrete list_of_lists is [['P2s2', 'P_IV', 'P_III', 'P_I', 'P2c1', 'P1c'],
['P2s2', 'P_IV', 'P_III', 'P_II', 'P2c1', 'P1c'], ['P_V', 'P_IV', 'P_III',
'P_I', 'P2c1', 'P1c'], ['P_V', 'P_IV', 'P_III', 'P_II', 'P2c1', 'P1c'], ['P_VI',
'P_IV', 'P_III', 'P_I', 'P2c1', 'P1c'], ['P_VI', 'P_IV', 'P_III', 'P_II',
'P2c1', 'P1c']]
[1, 1, 1, 1, 1, 1]
```

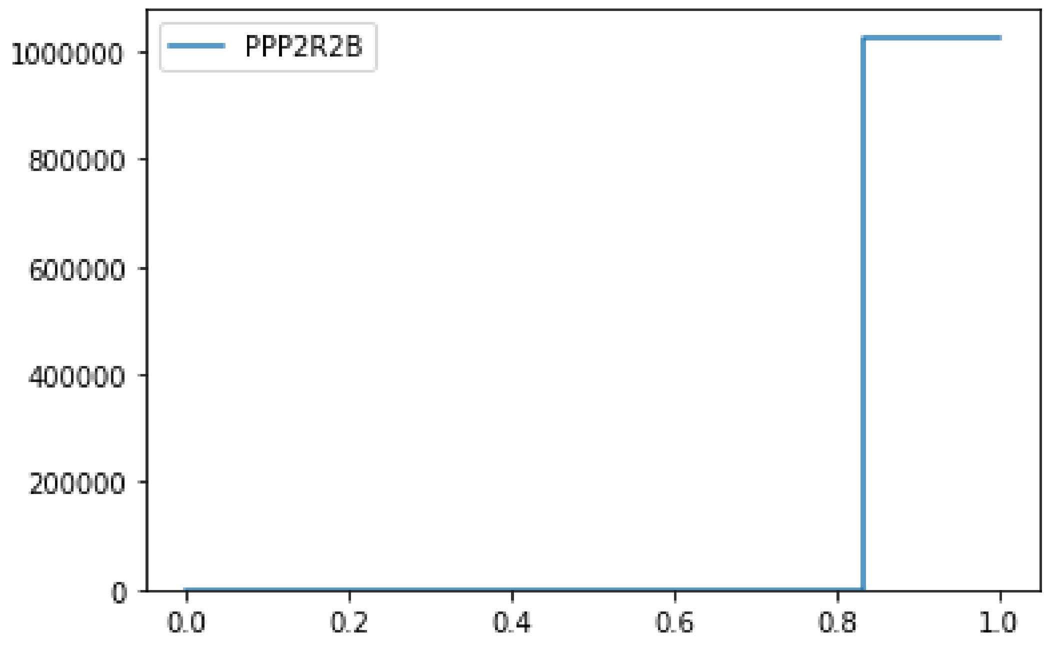
2 Ly et al. 2014 - Elutriation

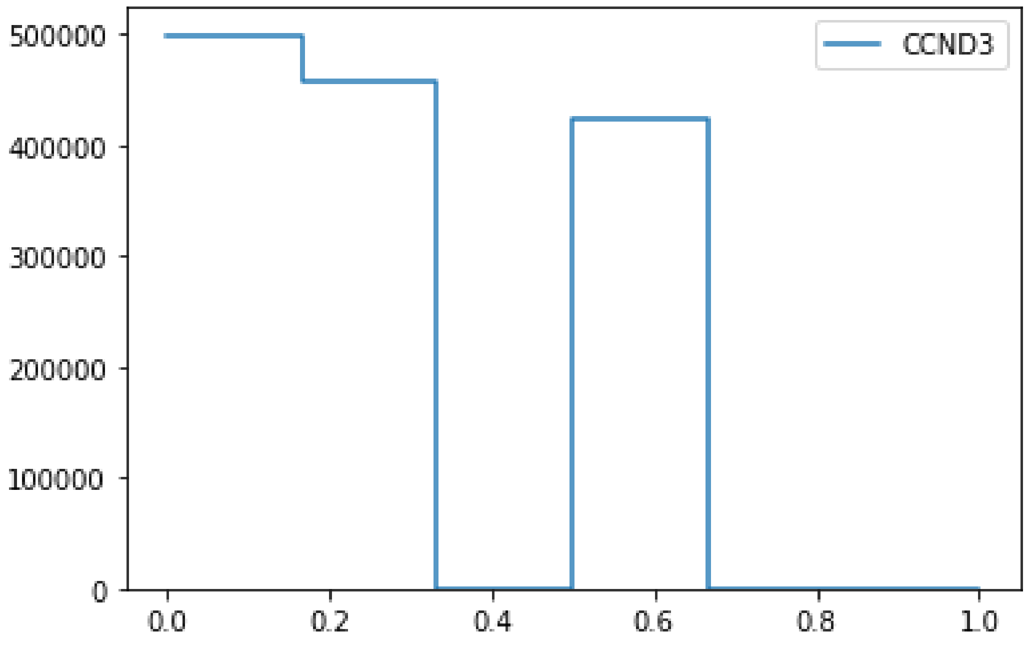
[ ]:
```python
##################
# Elutriation
##################

import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
from ast import literal_eval

# Enter (part of) gene names to plot
display = []

directory = os.path.abspath('')
if not display:
    data_file = os.path.join(directory, 'merged.xlsx')
    df = pd.read_excel(data_file)
    display = list(set(df.loc[:, 'Gene name']))
    print(df.loc[:, ['Gene name', 'ccModel Paul ID']].drop_duplicates().
 ↪dropna())
    display = [val for val in display if isinstance(val, str)]

# Fetch data of interest and prepare for plotting
data_file = os.path.join(directory, '2014_Ly_Mass_Spec_6 phases_full.csv')
df = pd.read_csv(data_file)

genes_of_interest = []
for string in display:
    for item in list(df.loc[:, 'gene_names']):
        if string == item:
            genes_of_interest.append(item)

rows_of_interest = [True if item in genes_of_interest else False for item in␣
 ↪list(df['gene_names'])]
df_of_interest = df.loc[rows_of_interest]

time = np.linspace(1/6, 1-1/6, 5)
time = [val for val in time for _ in (0, 1)]
time.insert(0, 0)
time.insert(100, 1)

data_to_plot = {}
for gene in genes_of_interest:
    row = df_of_interest.loc[df_of_interest['gene_names'] == gene]
    abundances = []
    for i in range(6):
        column_name = 'LFQ_intensity_F{}'.format(i+1)
        abundances.append(row.iloc[0][column_name])
    doubled = []
    for val in abundances:
        doubled.extend([val, val])
    data_to_plot[gene] = doubled

# Plot data
for gene in data_to_plot:
    plt.figure()
    plt.plot(time, data_to_plot[gene])
    plt.gca().set_ylim(bottom=0)
    plt.gca().legend((gene,))

# Exclude samples where zero intesity occurs in non-consecutive fractions
```

```
    Gene name ccModel Paul ID
0       CDC27             Apc
20       E2F2             E2f
22     ANAPC4             Apc
33    PPP2R2C             B55
42     ANAPC5             Apc
66      CDC23             Apc
75     CDC25B           Cdc25
84       E2F1             E2f
85      CCNE1              Ce
92       FZR1             Cdh
100     CCND1              Cd
112     FOXM1             Fox
122    CDKN1B             p27
```

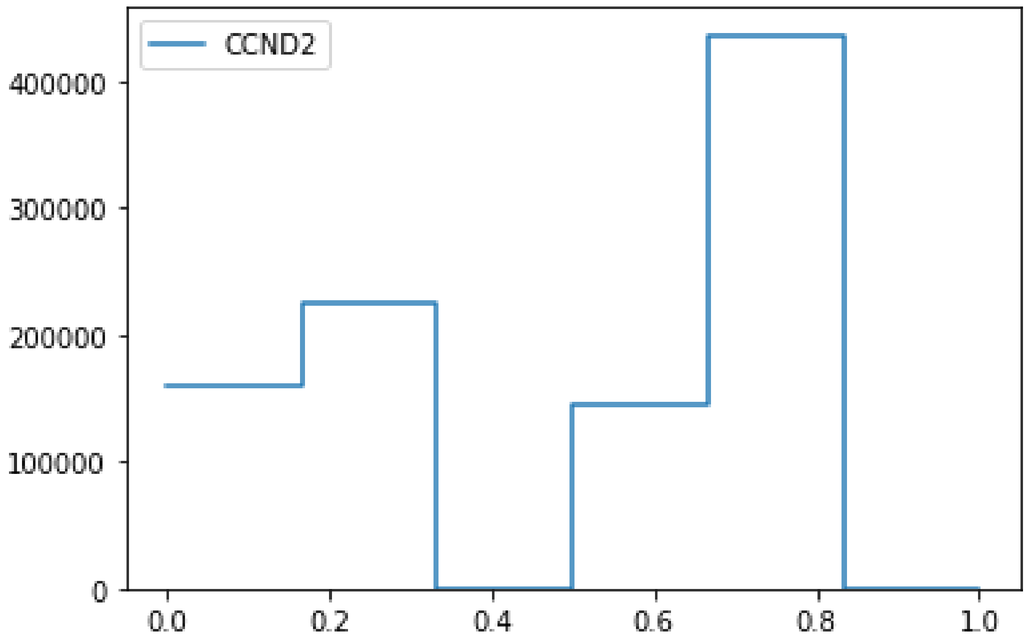
```
126     FBXO5              Emi
129      E2F3              E2f
132     CCND3               Cd
155     CDC20            Cdc20
159     CCND2               Cd
163     MASTL               Gw
167    CDKN1A              p21
175    ARPP19             Ensa
188     CDC16              Apc
200     CCNA1               Ca
207     CCNB1               Cb
214       RB1               Rb
220      TP53              p53
248   ANAPC11              Apc
271      ENSA             Ensa
286     CCNA2               Ca
288     CCNB3               Cb
296    ANAPC1              Apc
306   PPP2R2B              B55
329     CCNB2               Cb
334    CDC25C            Cdc25
345    CDC25A            Cdc25
350   ANAPC10              Apc
367      WEE1              Wee
375      CDK1              pCb
384     CCNE2               Ce
392   PPP2R2D              B55
400    ANAPC2              Apc
408     CDC26              Apc
412    ANAPC7              Apc
424   PPP2R2A              B55

C:\Users\wolf5212\AppData\Local\Continuum\anaconda3\lib\site-
packages\ipykernel_launcher.py:54: RuntimeWarning: More than 20 figures have
been opened. Figures created through the pyplot interface
(`matplotlib.pyplot.figure`) are retained until explicitly closed and may
consume too much memory. (To control this warning, see the rcParam
`figure.max_open_warning`).
```


1e8
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
TP53

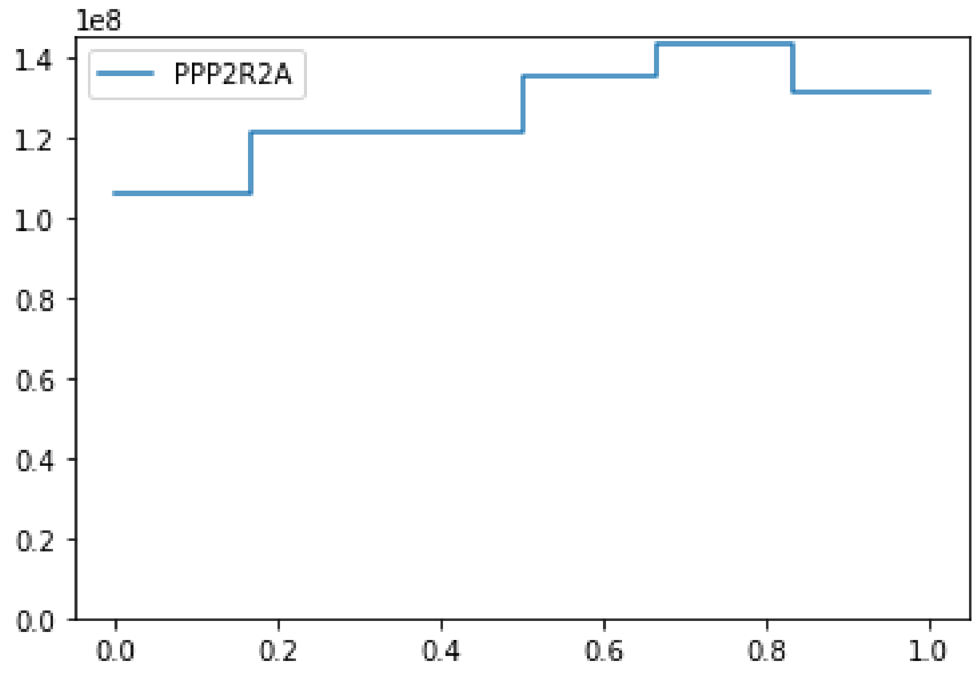

1000000
800000
600000
400000
200000
0
0.0
0.2
0.4
0.6
0.8
1.0
PPP2R2B

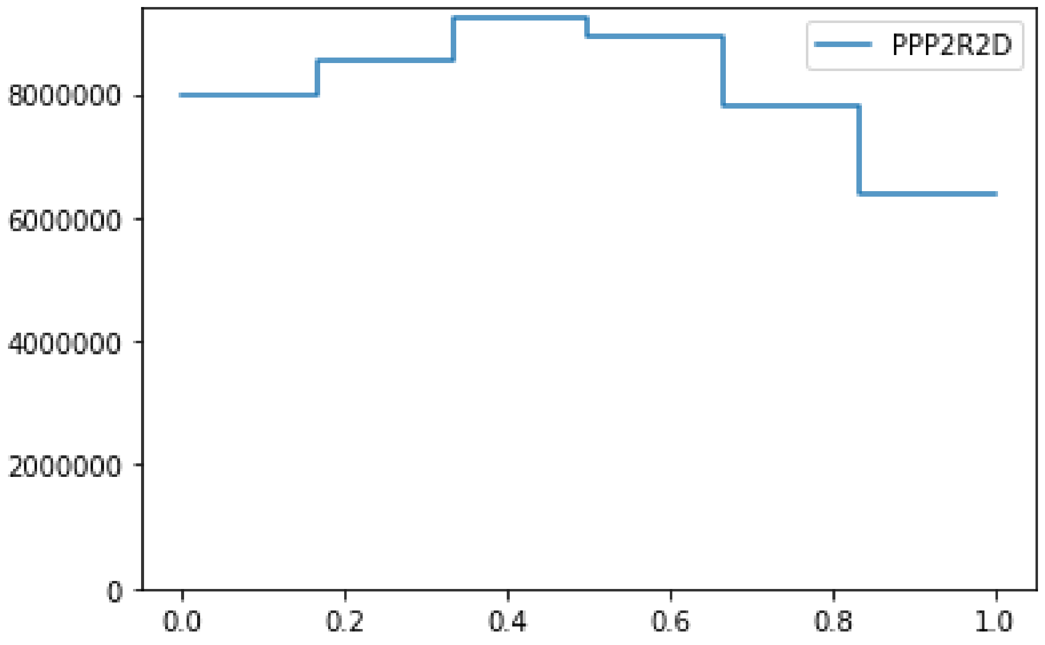


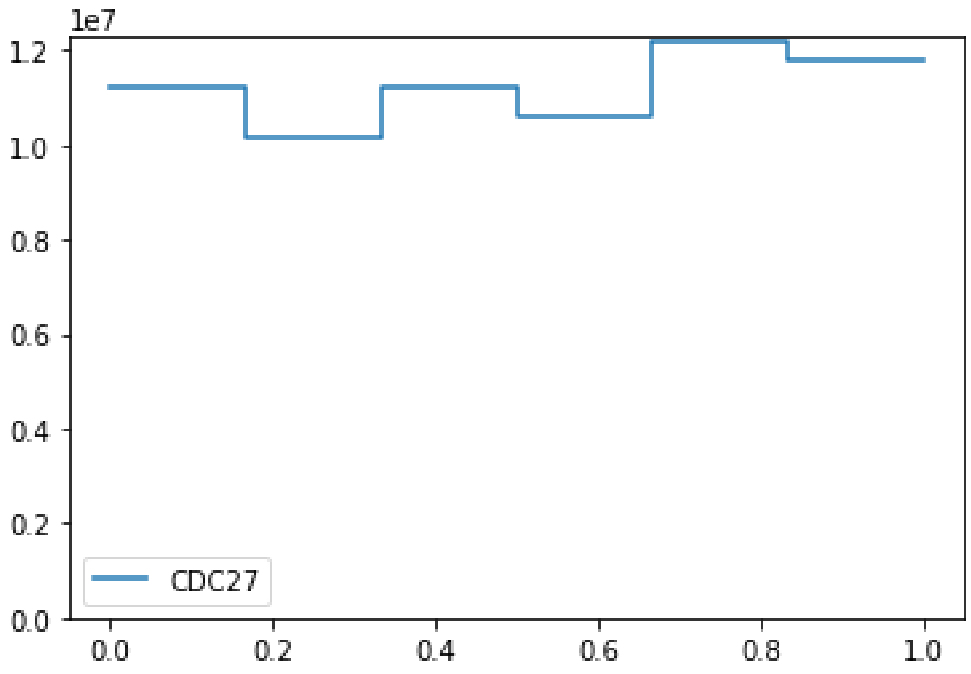
CCND3
500000
400000
300000
200000
100000
0
0.0
0.2
0.4
0.6
0.8
1.0

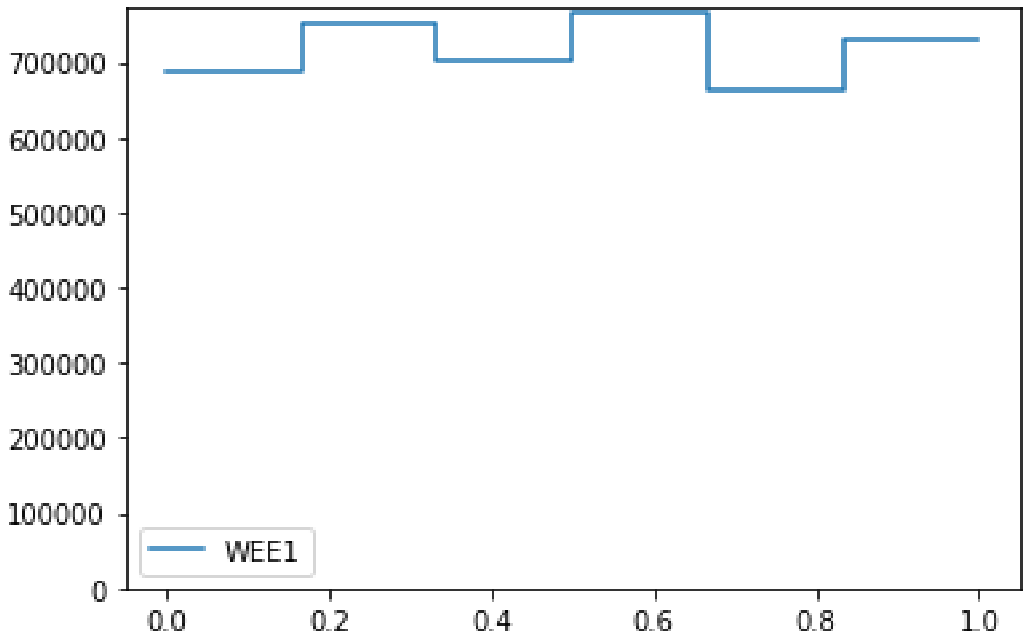


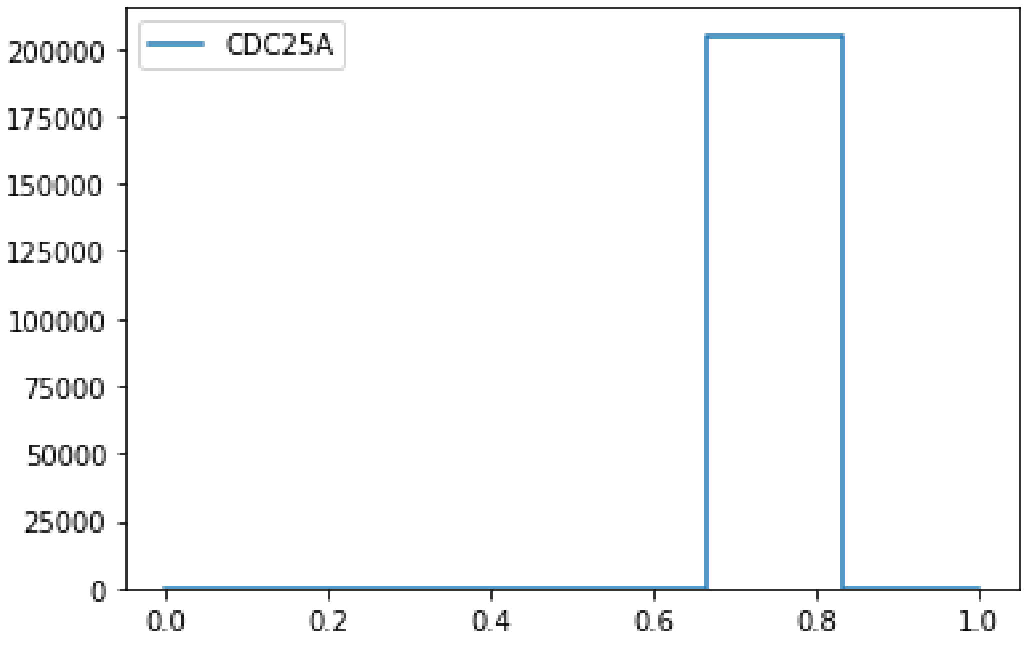
CCND2
400000
300000
200000
100000
0
0.0
0.2
0.4
0.6
0.8
1.0

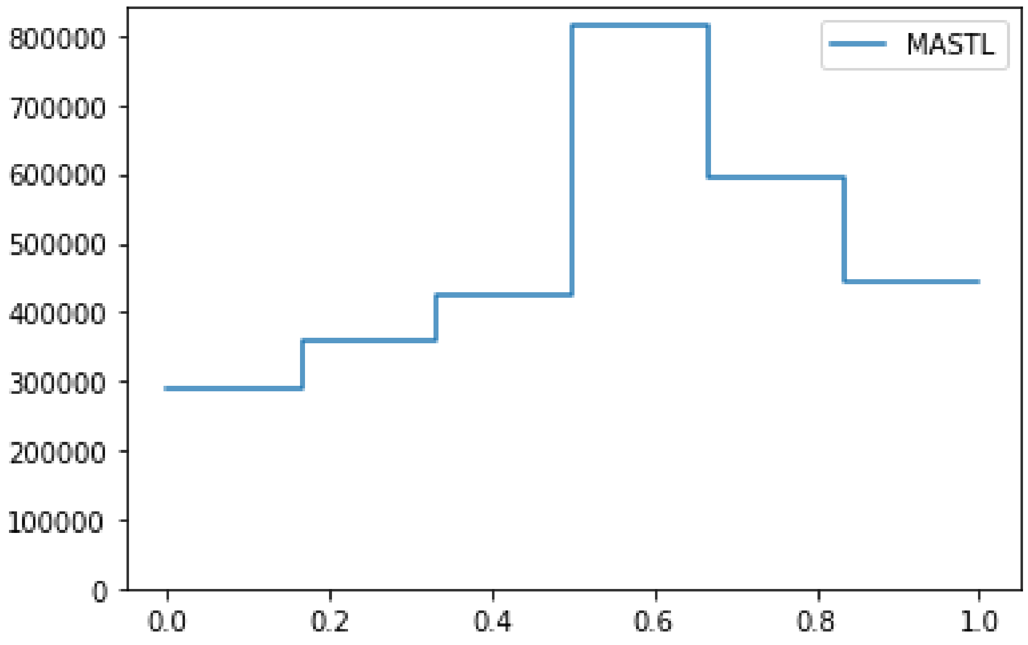

1e8
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
PPP2R2A

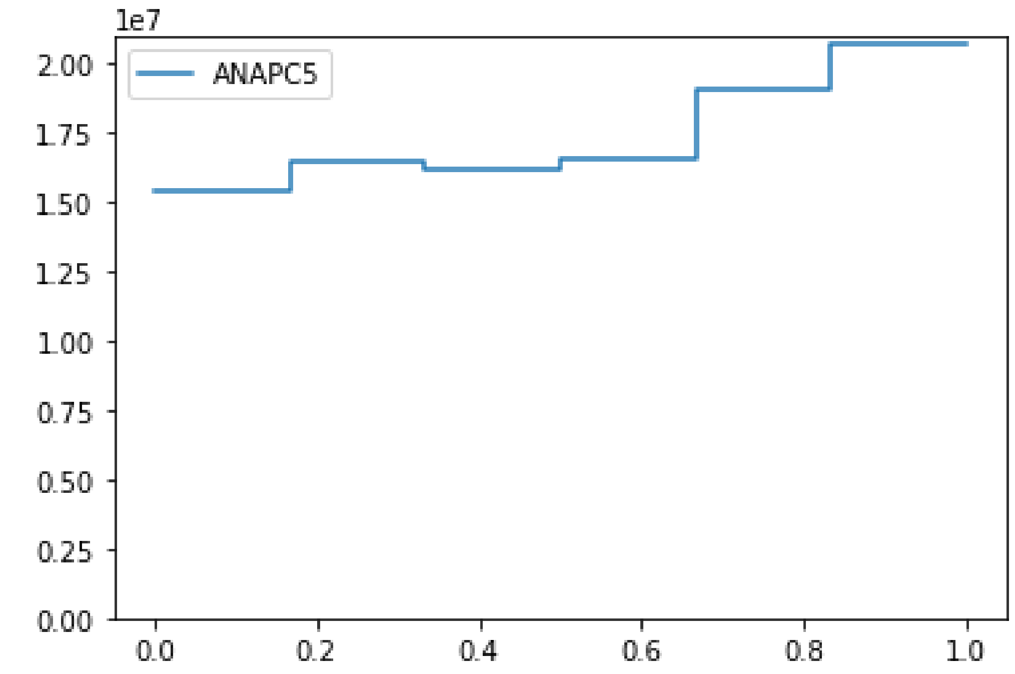

8000000
6000000
4000000
2000000
0
0.0
0.2
0.4
0.6
0.8
1.0
PPP2R2D

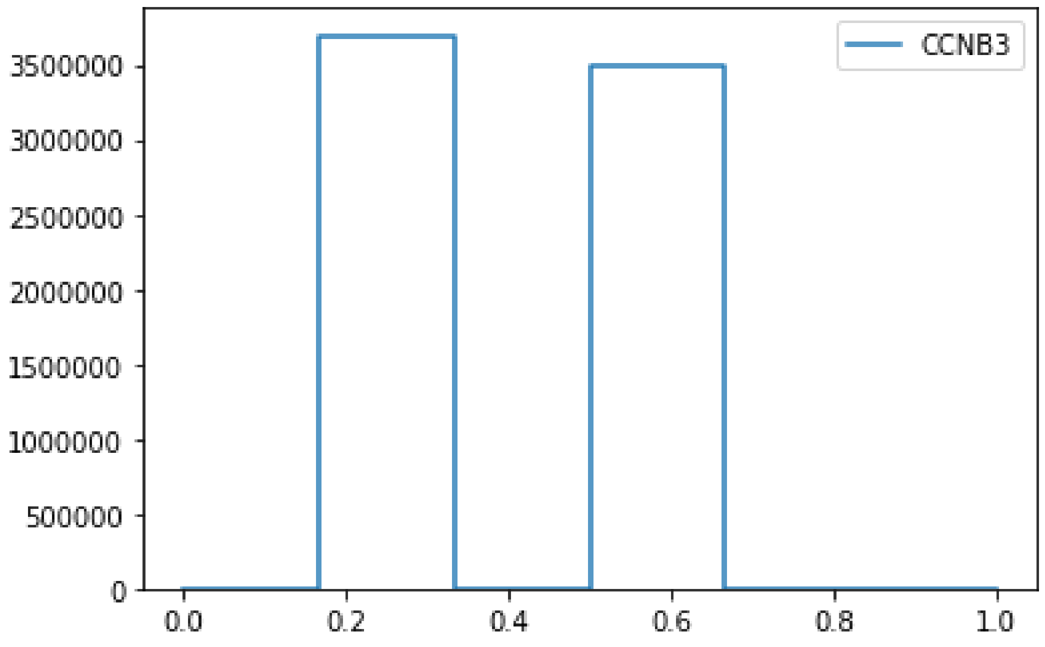

1e7
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
CDC27

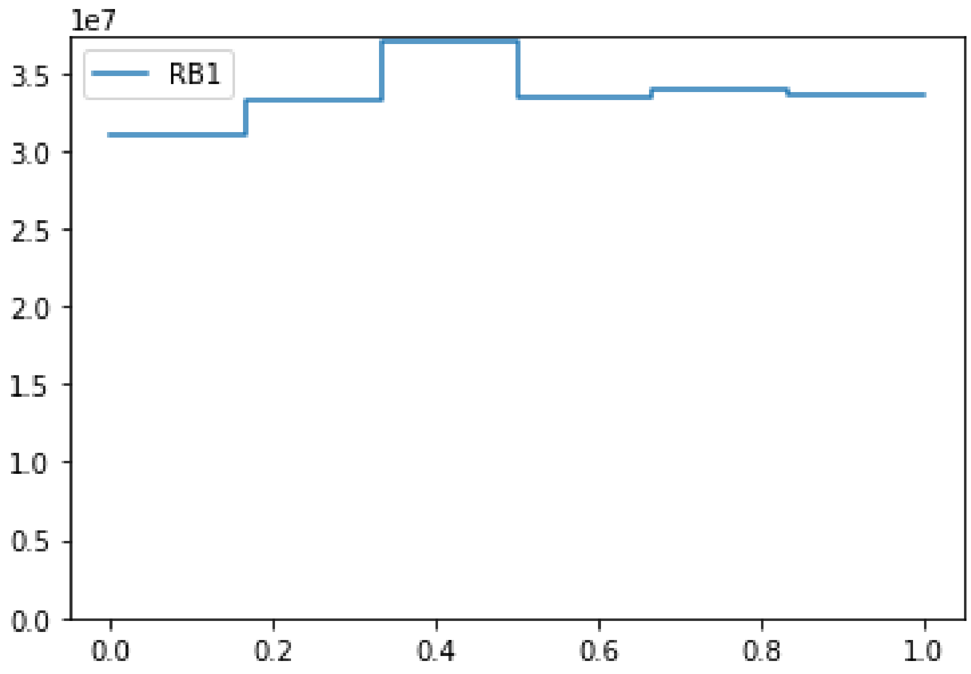

700000
600000
500000
400000
300000
200000
100000
0
0.0
0.2
0.4
0.6
0.8
1.0
WEE1

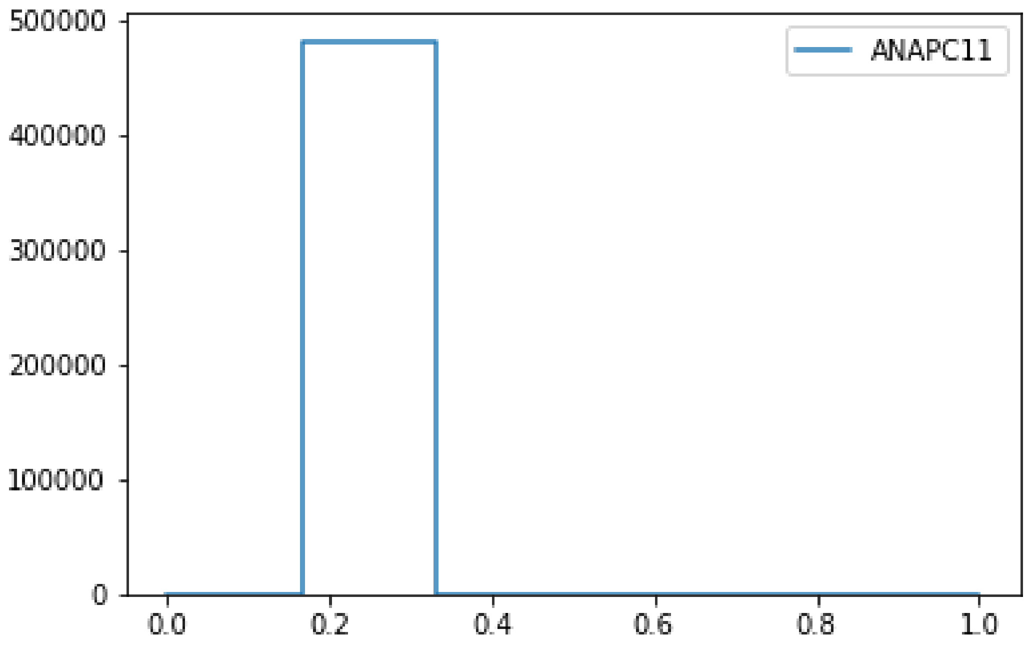

CDC25A
200000
175000
150000
125000
100000
75000
50000
25000
0
0.0
0.2
0.4
0.6
0.8
1.0

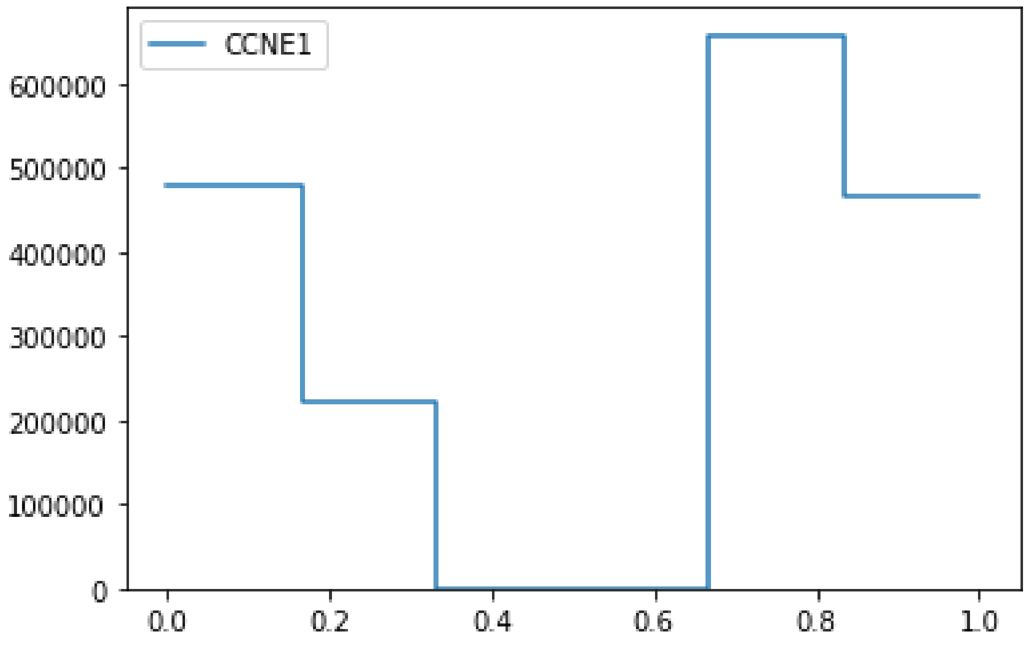

MASTL
800000
700000
600000
500000
400000
300000
200000
100000
0
0.0
0.2
0.4
0.6
0.8
1.0

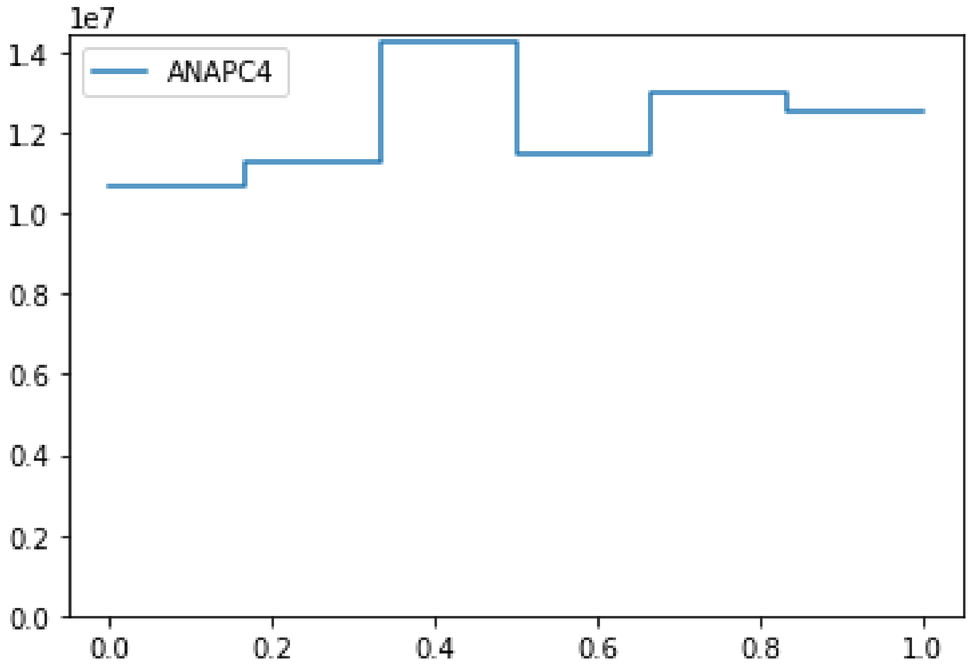

1e7
ANAPC5
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0

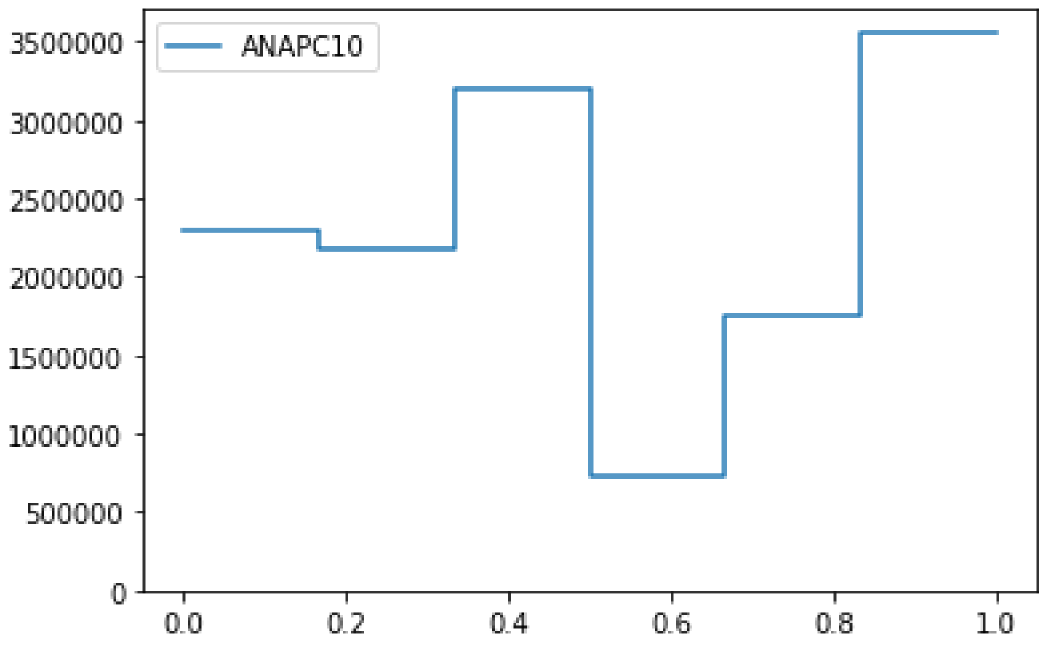

CCNB3
3500000
3000000
2500000
2000000
1500000
1000000
500000
0
0.0
0.2
0.4
0.6
0.8
1.0

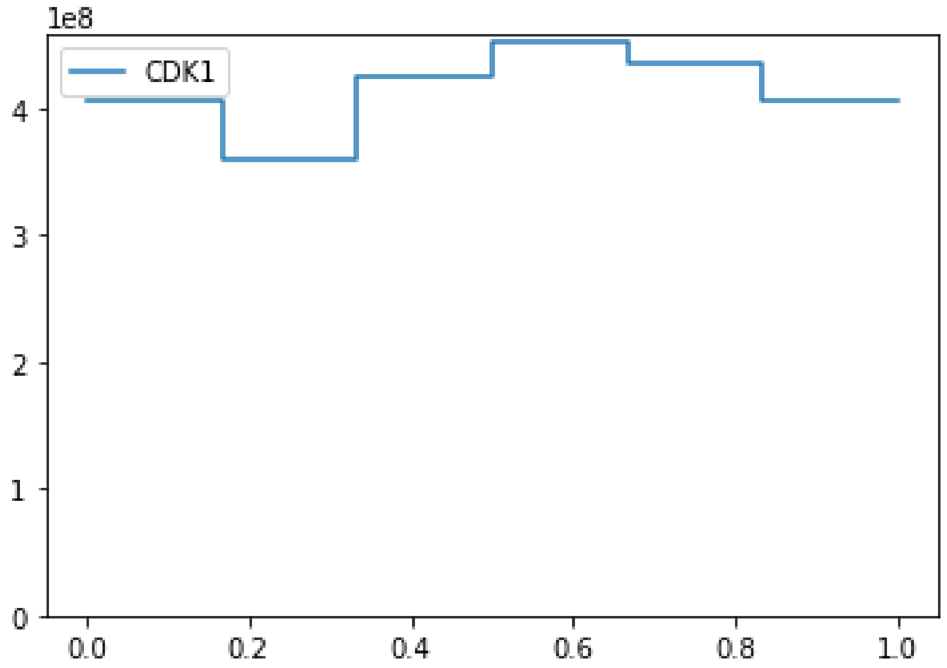

1e7
RB1
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

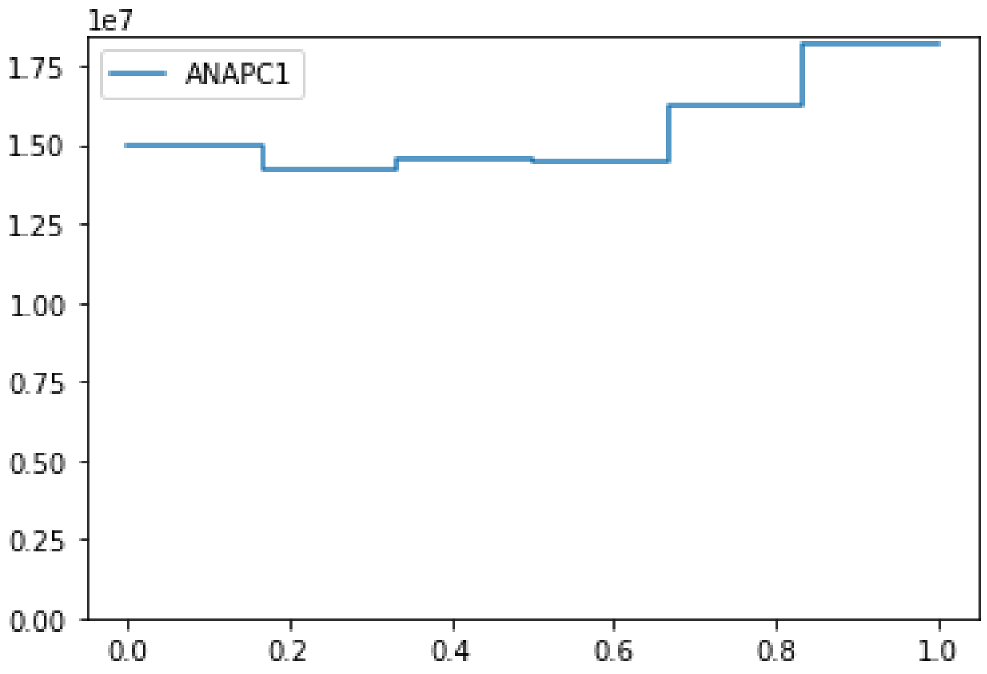

500000
400000
300000
200000
100000
0
ANAPC11
0.0
0.2
0.4
0.6
0.8
1.0

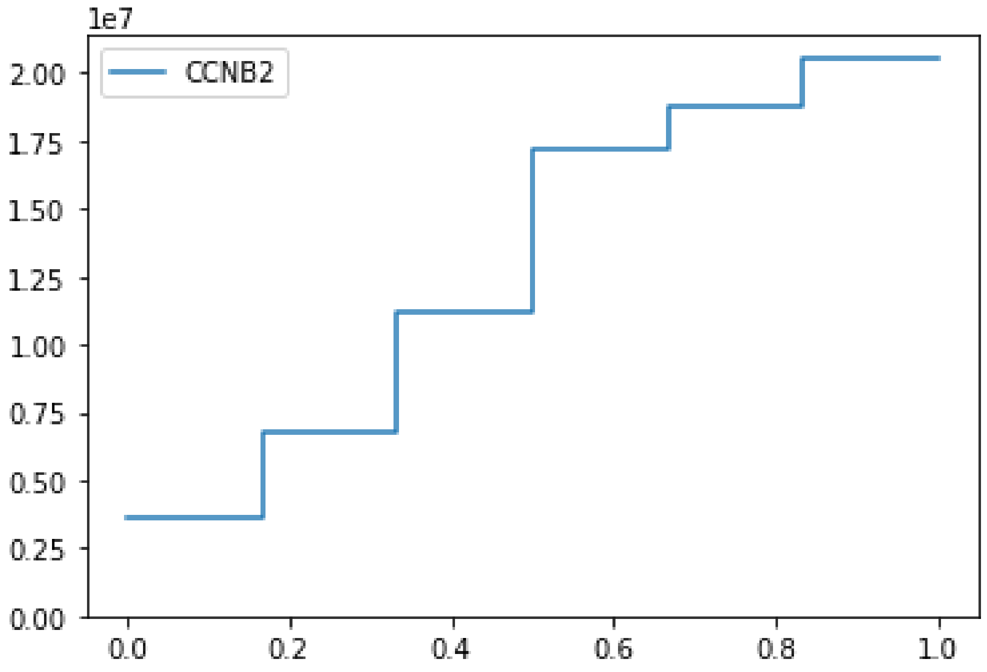

CCNE1
600000
500000
400000
300000
200000
100000
0
0.0
0.2
0.4
0.6
0.8
1.0

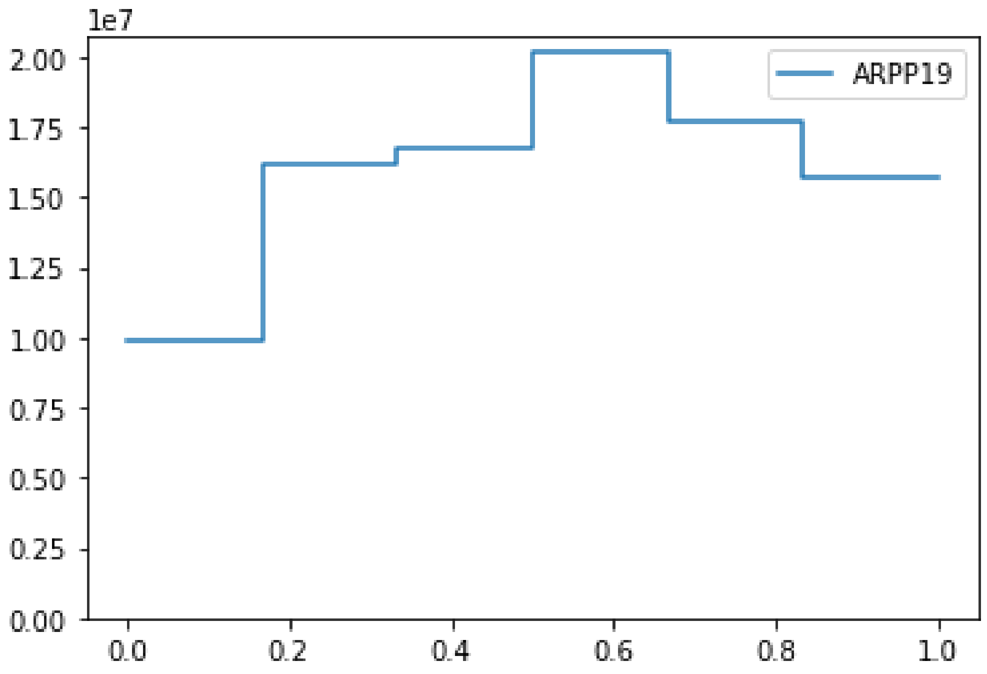

1e7
ANAPC4
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

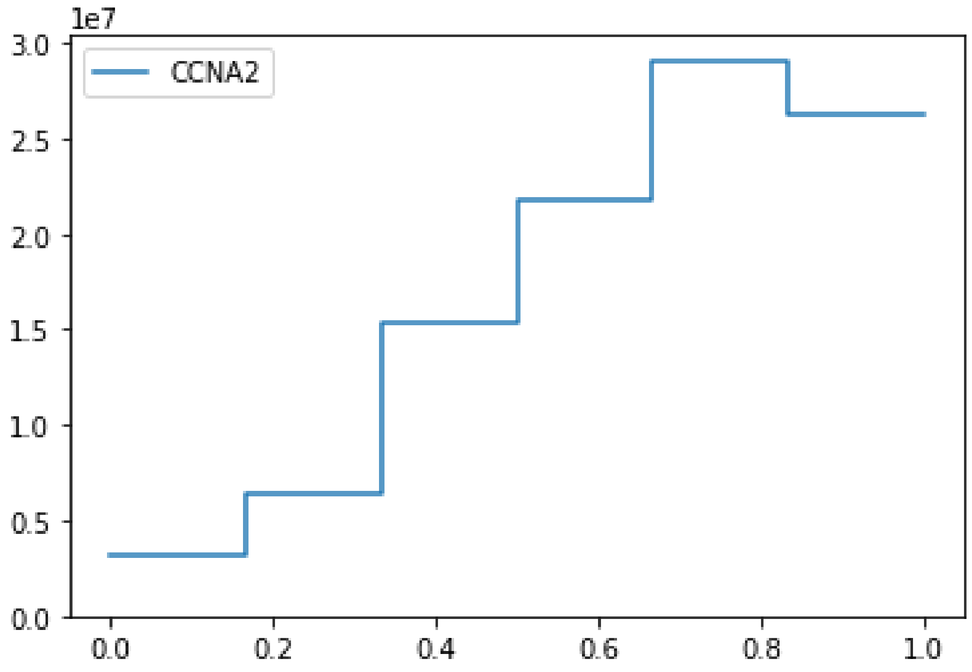

ANAPC10
3500000
3000000
2500000
2000000
1500000
1000000
500000
0
0.0
0.2
0.4
0.6
0.8
1.0

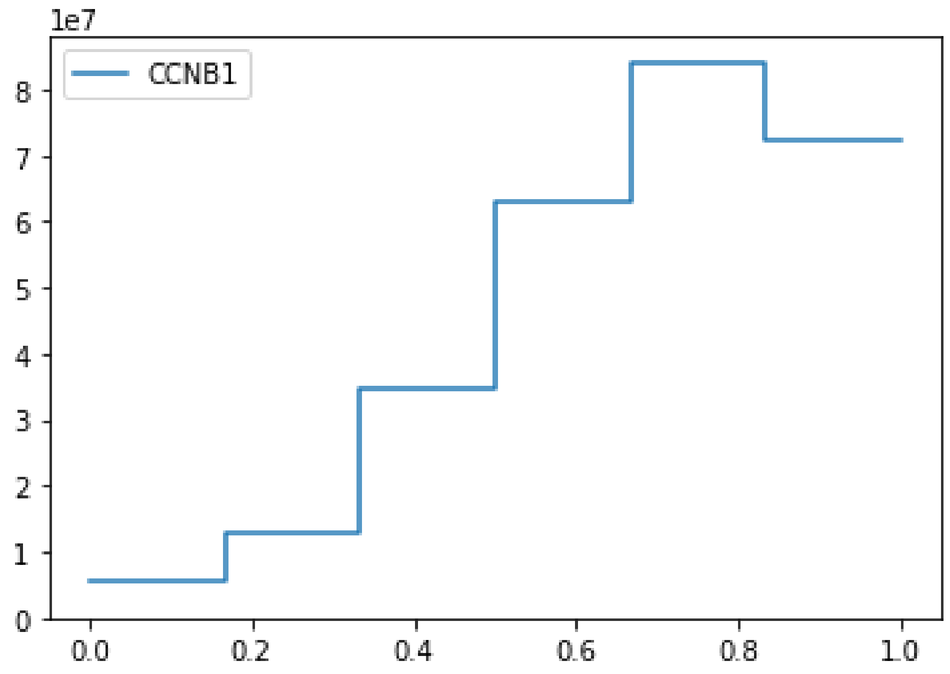

1e8
CDK1
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0

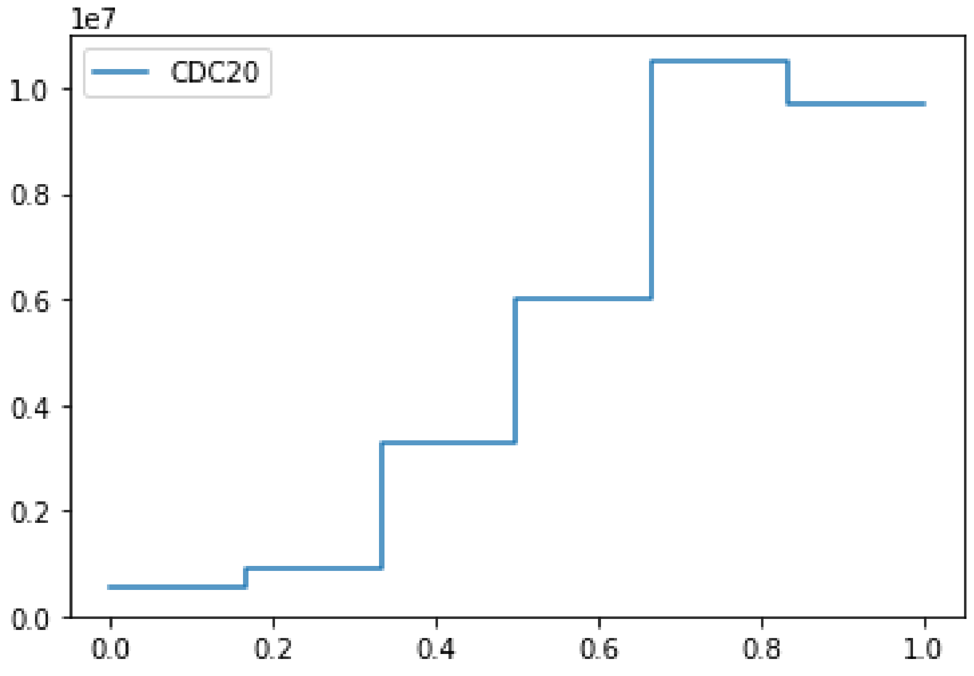

1e7
ANAPC1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
0.0
0.2
0.4
0.6
0.8
1.0

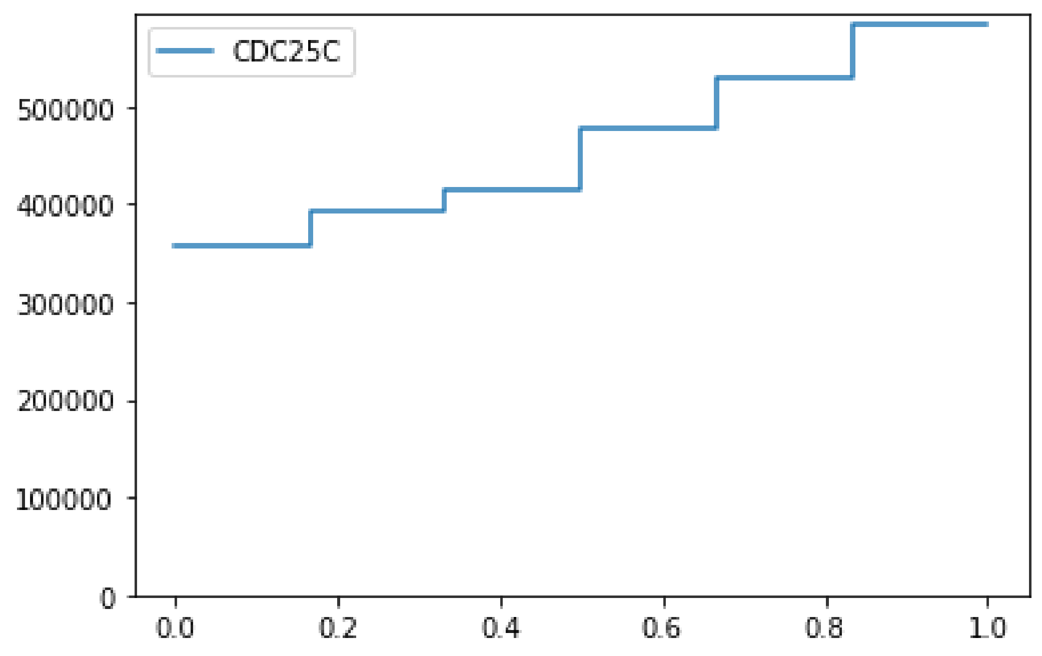

1e7
CCNB2
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
0.0
0.2
0.4
0.6
0.8
1.0

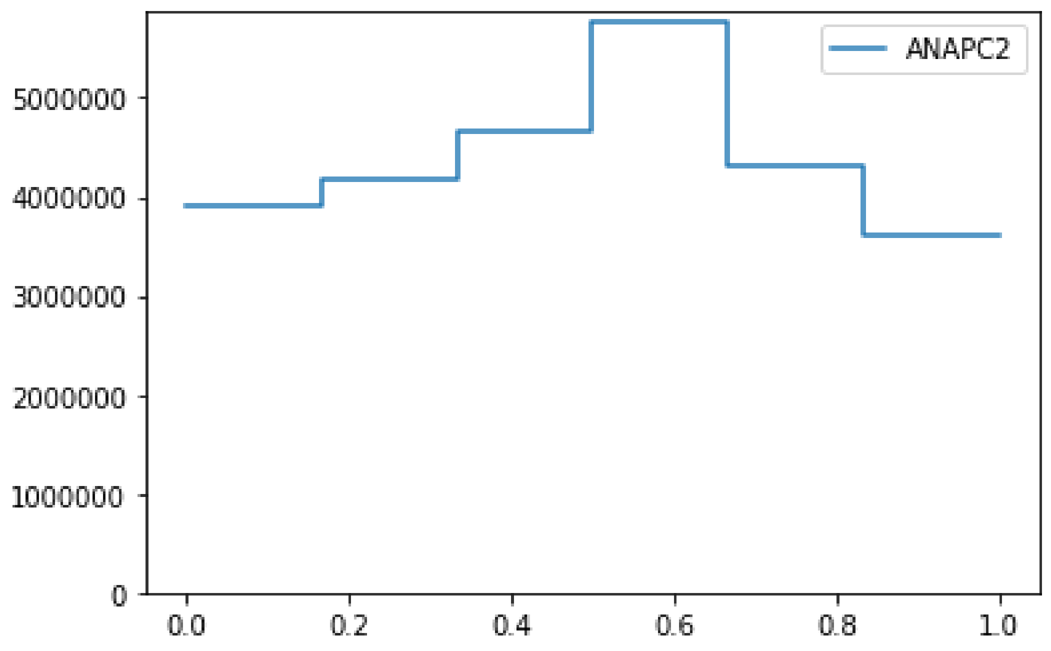

1e7
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0
ARPP19

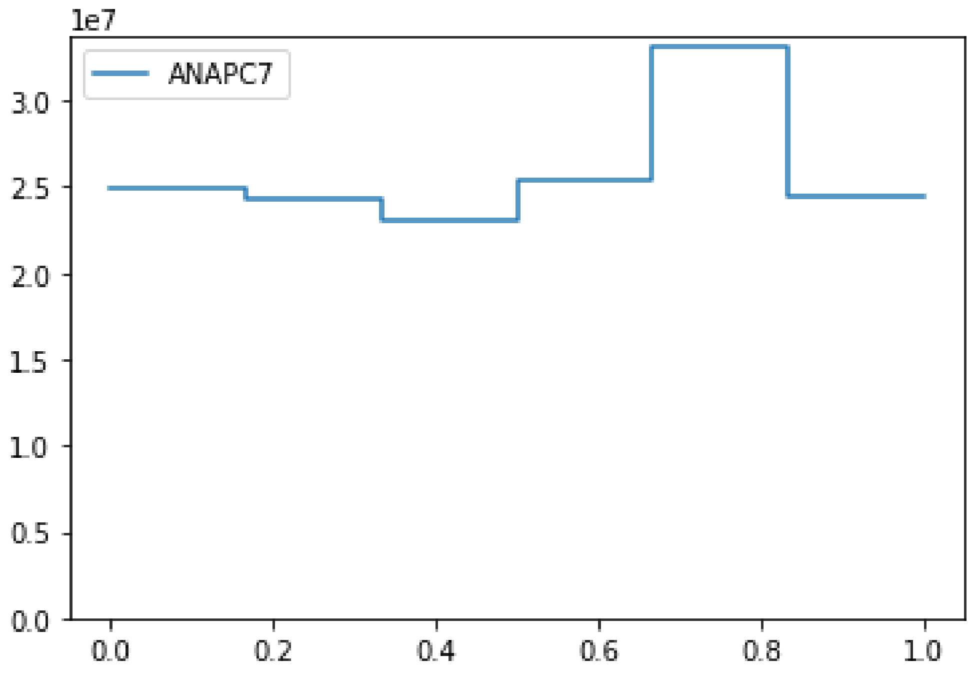

1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
CCNA2

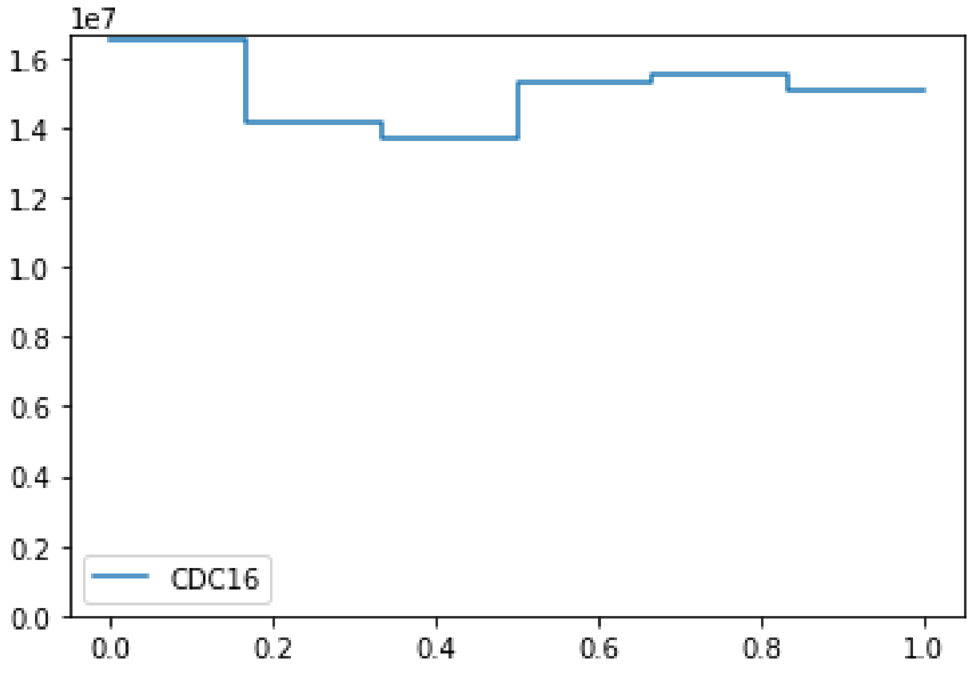

1e7
CCNB1
0
1
2
3
4
5
6
7
8
0.0
0.2
0.4
0.6
0.8
1.0

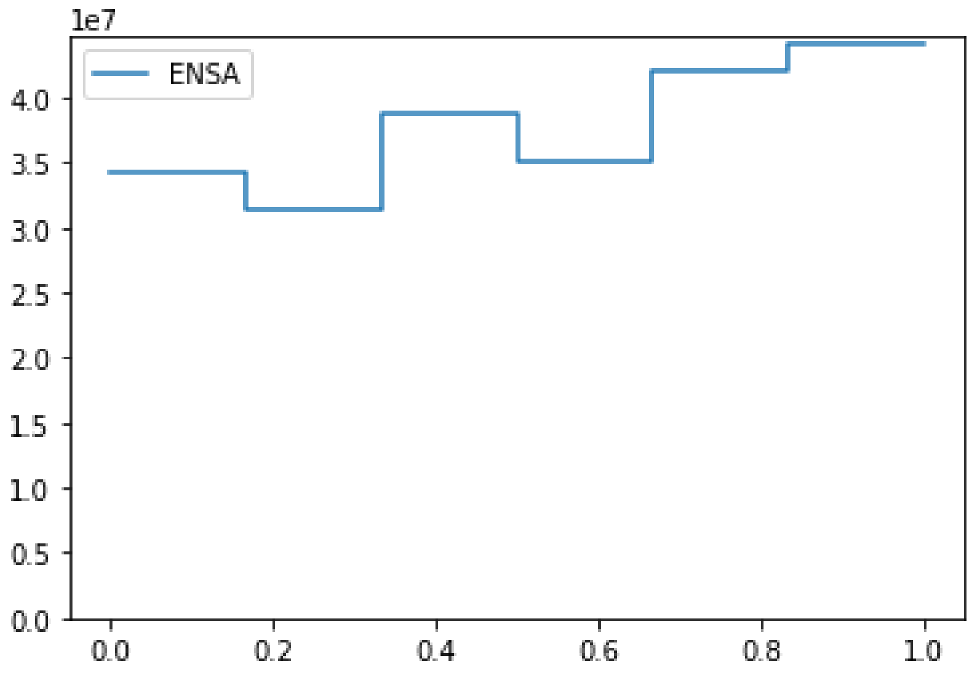

1e7
CDC20
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

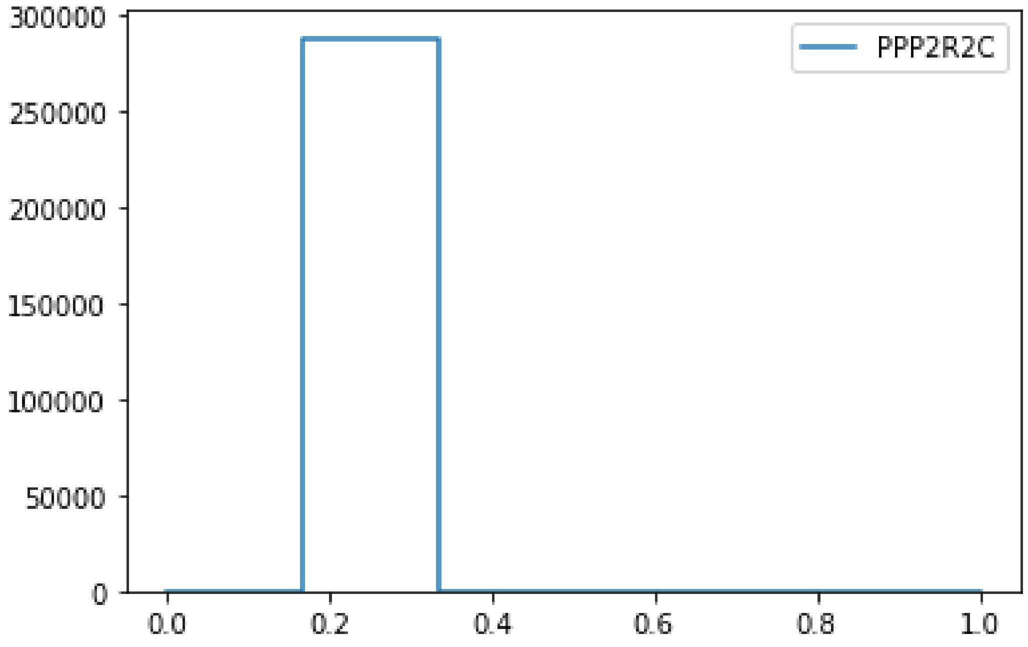

CDC25C
0
100000
200000
300000
400000
500000
0.0
0.2
0.4
0.6
0.8
1.0

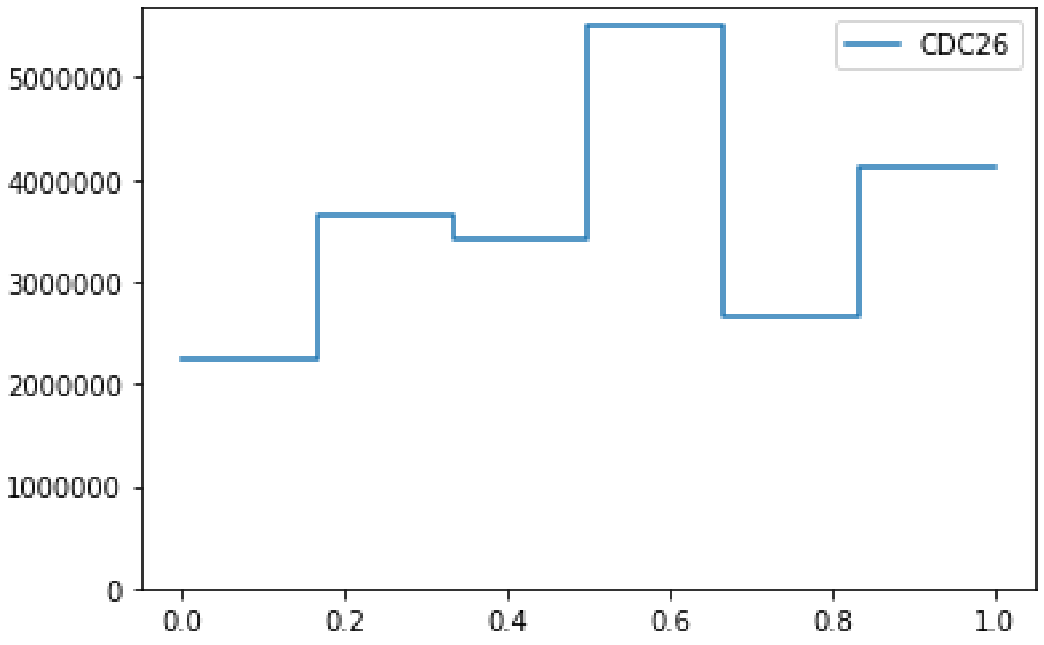

ANAPC2
0
1000000
2000000
3000000
4000000
5000000
0.0
0.2
0.4
0.6
0.8
1.0


1e7
ANAPC7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

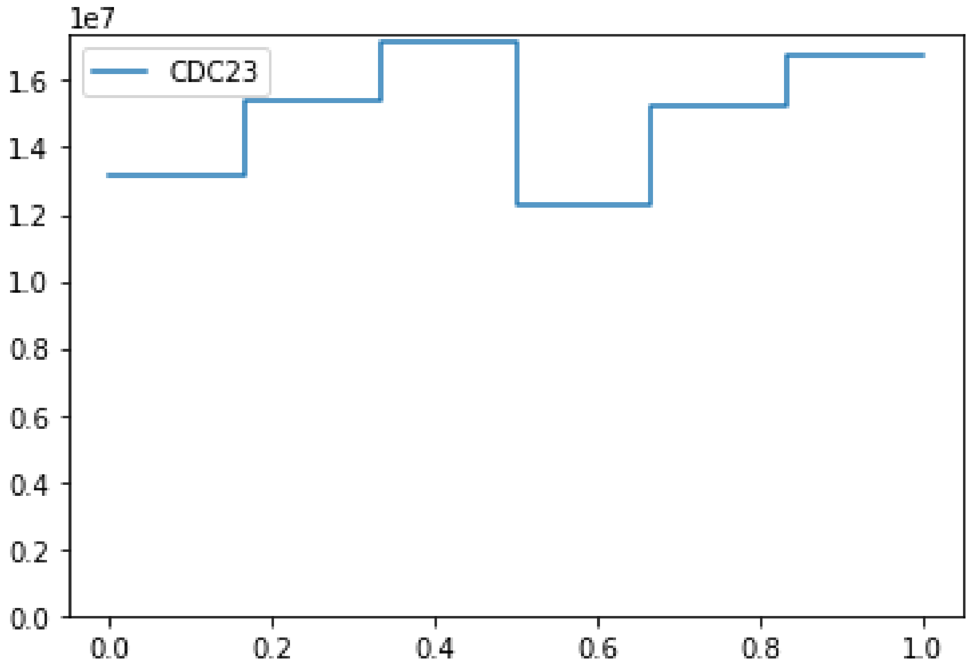

1e7
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
CDC16
0.0
0.2
0.4
0.6
0.8
1.0


1e7
ENSA
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

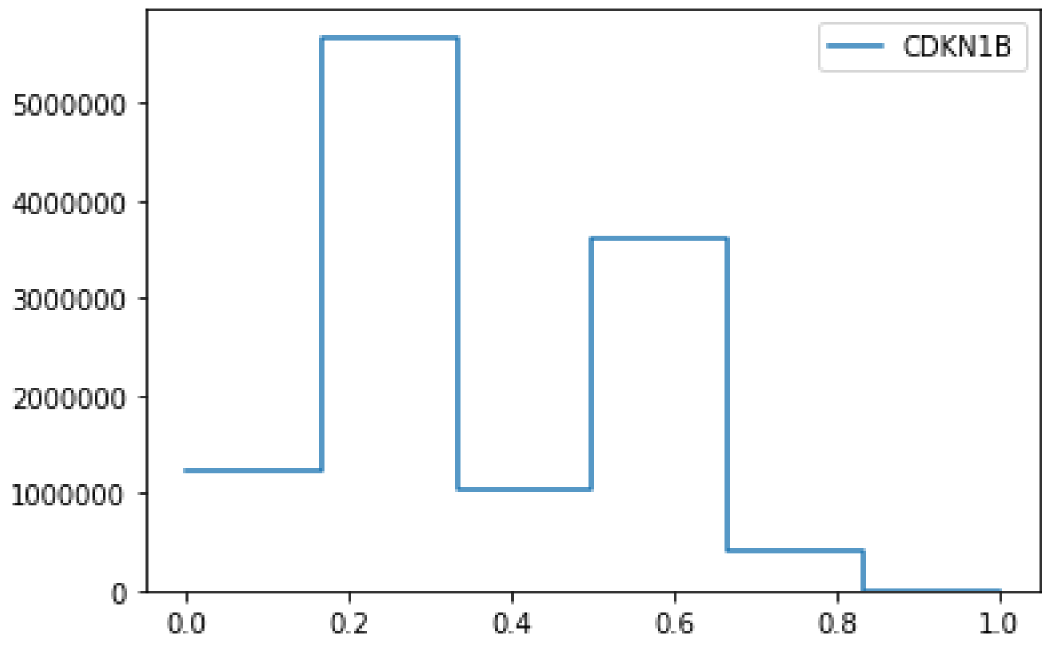

PPP2R2C
300000
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0

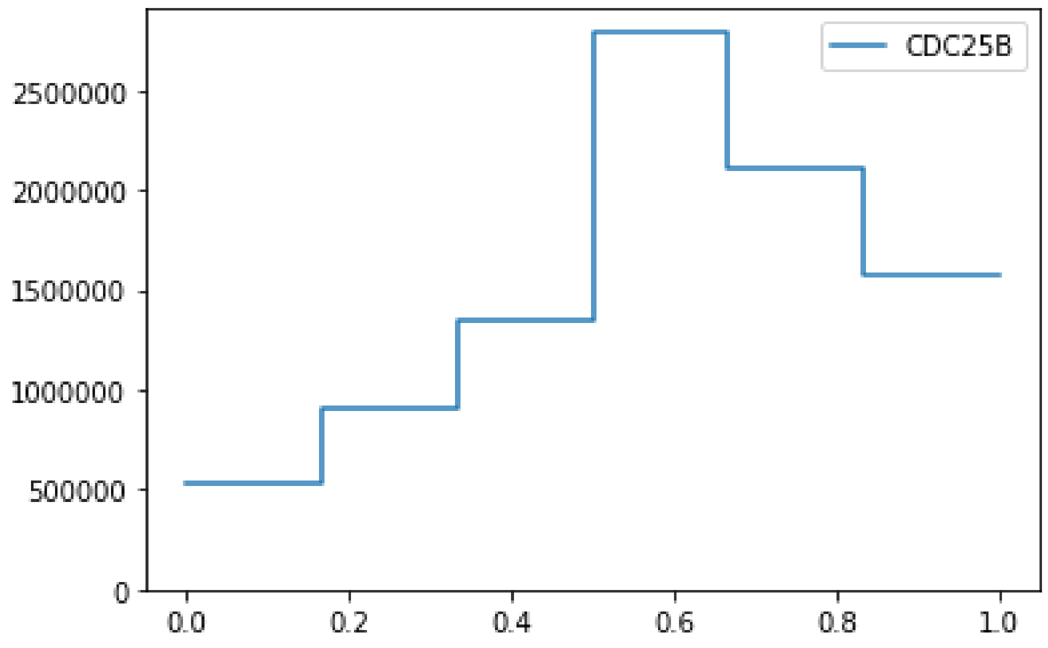


CDC26
5000000
4000000
3000000
2000000
1000000
0
0.0
0.2
0.4
0.6
0.8
1.0



1e7
CDC23
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0





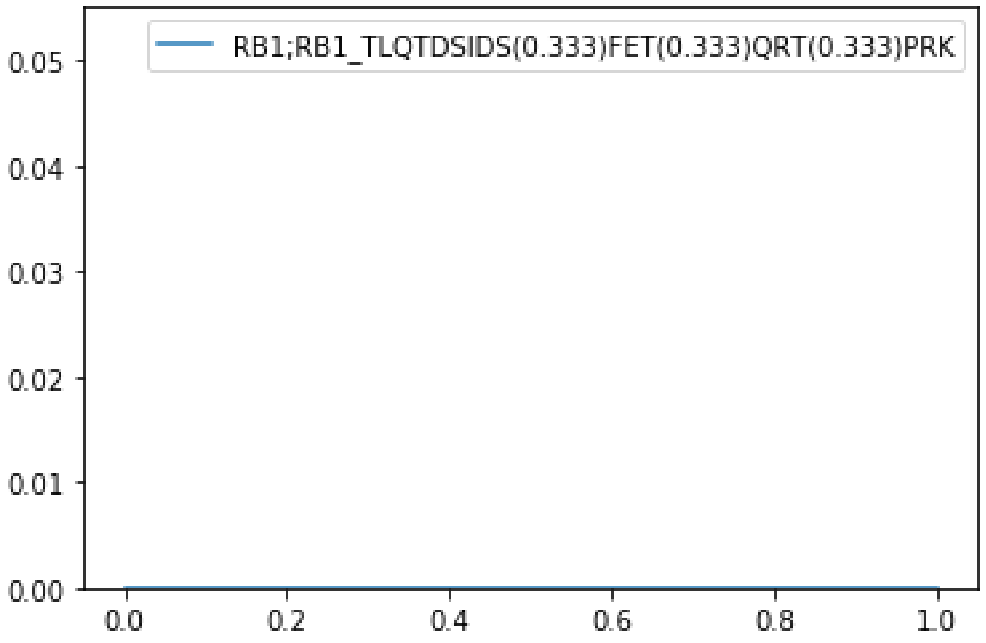
Conclusion: * Several cell cycle regulators do not change concentration * No replicate data provided * Contains CDC20 and shows that it increases over the cell cycle * Low fold changes, presumably due to limited cell cycle separation * Cyclin E does not behave as expected

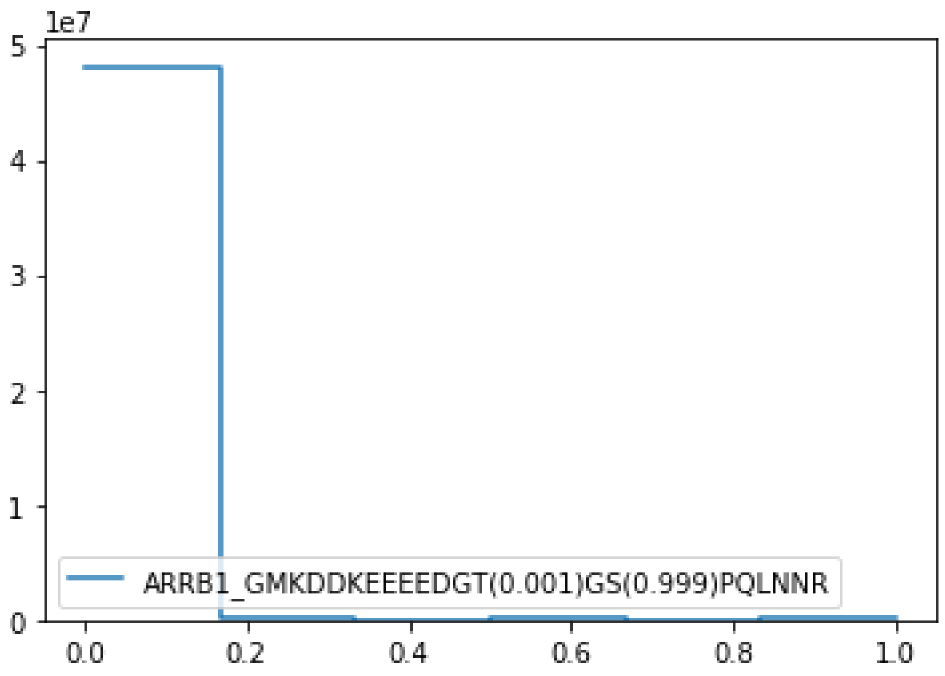
[ ]:
```
##################
# Elutriation phospho
##################

import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
from ast import literal_eval

# Enter (part of) gene names to plot
display = []

directory = os.path.abspath('')
if not display:
    data_file = os.path.join(directory, 'merged.xlsx')
    df = pd.read_excel(data_file)
    display = list(set(df.loc[:, 'Gene name']))
    print(df.loc[:, ['Gene name', 'ccModel Paul ID']].drop_duplicates().
 ↪dropna())
    display = [val for val in display if isinstance(val, str)]

# Fetch data of interest and prepare for plotting
data_file = os.path.join(directory, 'elife-01630-supp4-v1.xlsx')
df = pd.read_excel(data_file)

genes_of_interest = []
sites_of_interest = []
for string in display:
    for gene, phospho_site in zip(list(df.loc[:, 'Gene.names']), list(df.loc[:,␣
 ↪'Phospho..STY..Probabilities'])):
        if string in str(gene):
            genes_of_interest.append(gene)
            sites_of_interest.append(phospho_site)
genes_and_sites = zip(genes_of_interest, sites_of_interest)

rows_of_interest = [True if item in genes_of_interest else False for item in␣
 ↪list(df['Gene.names'])]
df_of_interest = df.loc[rows_of_interest]

time = np.linspace(1/6, 1-1/6, 5)
time = [val for val in time for _ in (0, 1)]
time.insert(0, 0)
time.insert(100, 1)

data_to_plot = {}
for gene, site in genes_and_sites:
    row = df_of_interest.loc[df_of_interest['Phospho..STY..Probabilities'] ==␣
 ↪site]
    abundances = []
    for i in range(6):
        column_name = 'Intensity.F{}'.format(i+1)
        abundances.append(row.iloc[0][column_name])
    doubled = []
    for val in abundances:
        doubled.extend([val, val])
    data_to_plot[gene+'_'+site] = doubled

# Plot
for peptide in data_to_plot:
    plt.figure()
    plt.plot(time, data_to_plot[peptide])
    plt.gca().set_ylim(bottom=0)
    plt.gca().legend((peptide,))

# Exclude samples where zero intesity occurs in non-consecutive fractions
```

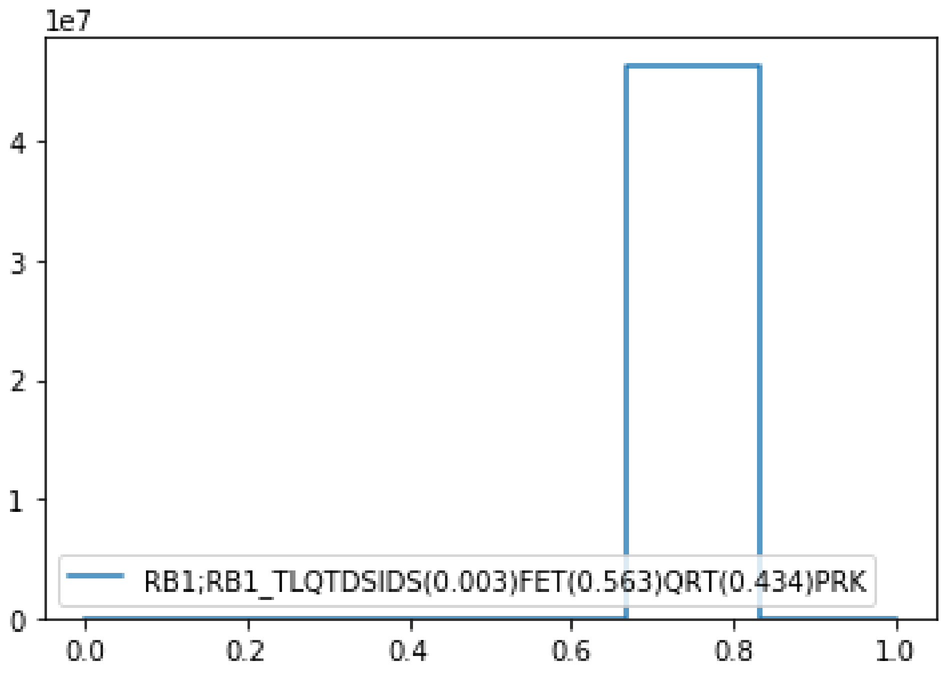
```
    Gene name ccModel Paul ID
0       CDC27             Apc
20       E2F2             E2f
22     ANAPC4             Apc
33    PPP2R2C             B55
42     ANAPC5             Apc
66      CDC23             Apc
75     CDC25B           Cdc25
84       E2F1             E2f
85      CCNE1              Ce
92       FZR1             Cdh
100     CCND1              Cd
112     FOXM1             Fox
122    CDKN1B             p27
126     FBXO5             Emi
129      E2F3             E2f
132     CCND3              Cd
155     CDC20           Cdc20
159     CCND2              Cd
163     MASTL              Gw
167    CDKN1A             p21
175    ARPP19            Ensa
188     CDC16             Apc
200     CCNA1              Ca
207     CCNB1              Cb
214       RB1              Rb
220      TP53             p53
```

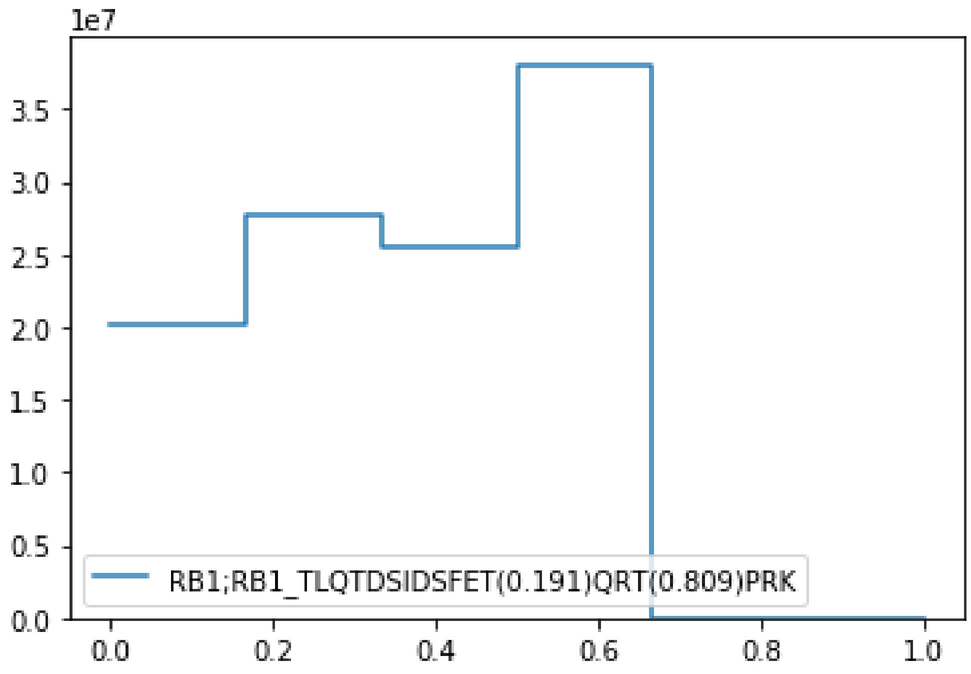
```
248   ANAPC11             Apc
271      ENSA            Ensa
286     CCNA2              Ca
288     CCNB3              Cb
296    ANAPC1             Apc
306   PPP2R2B             B55
329     CCNB2              Cb
334    CDC25C           Cdc25
345    CDC25A           Cdc25
350   ANAPC10             Apc
367      WEE1             Wee
375      CDK1             pCb
384     CCNE2              Ce
392   PPP2R2D             B55
400    ANAPC2             Apc
408     CDC26             Apc
412    ANAPC7             Apc
424   PPP2R2A             B55
```

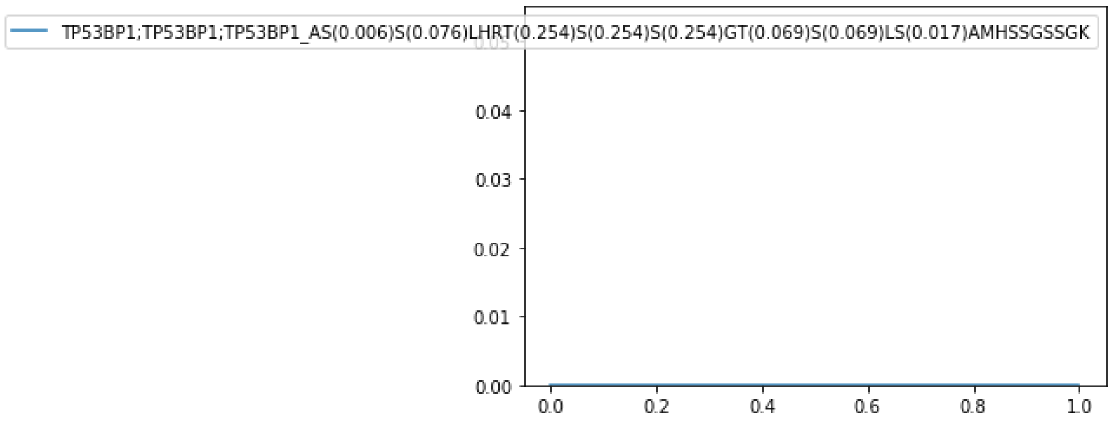
```
C:\Users\wolf5212\AppData\Local\Continuum\anaconda3\lib\site-
packages\ipykernel_launcher.py:57: RuntimeWarning: More than 20 figures have
been opened. Figures created through the pyplot interface
(`matplotlib.pyplot.figure`) are retained until explicitly closed and may
consume too much memory. (To control this warning, see the rcParam
`figure.max_open_warning`).
```

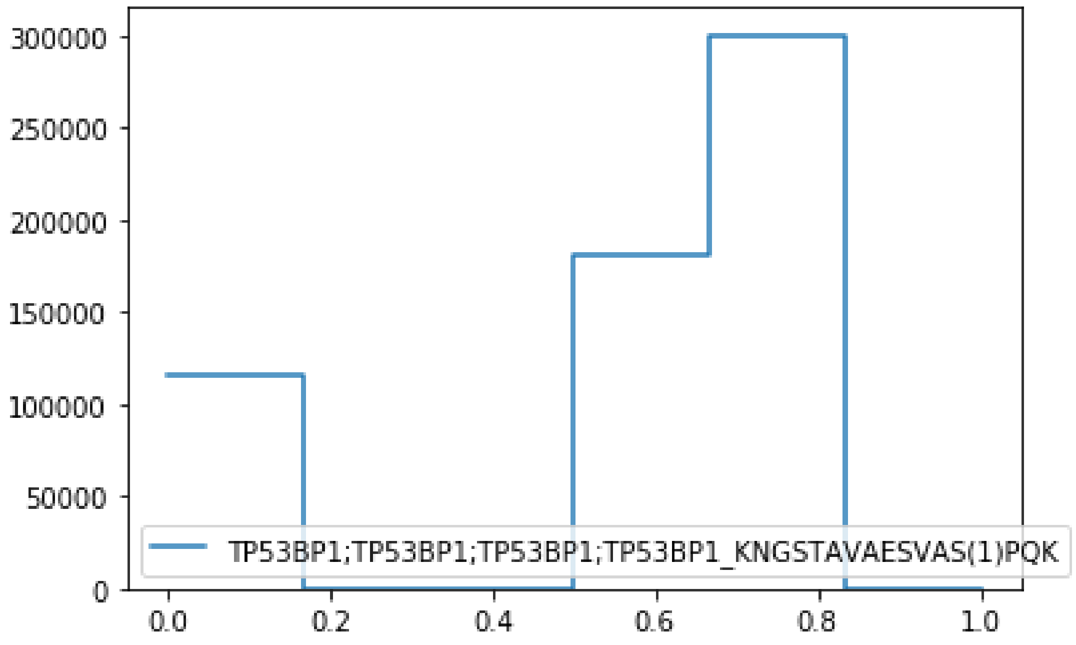


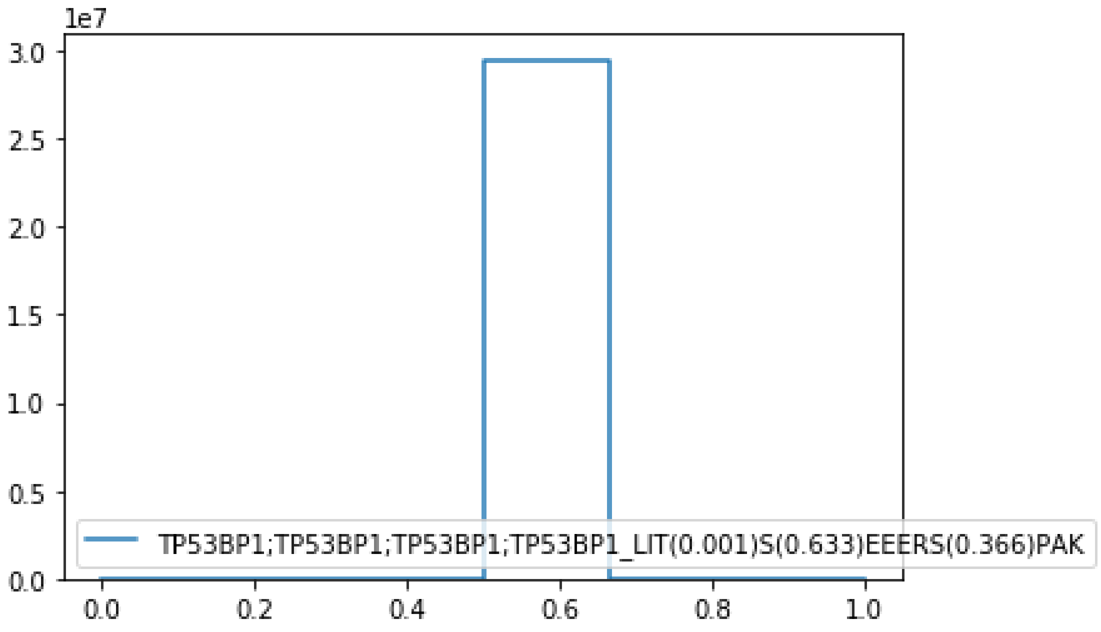

1e7
5
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0
ARRB1_GMKDDKEEEEDGT(0.001)GS(0.999)PQLNNR

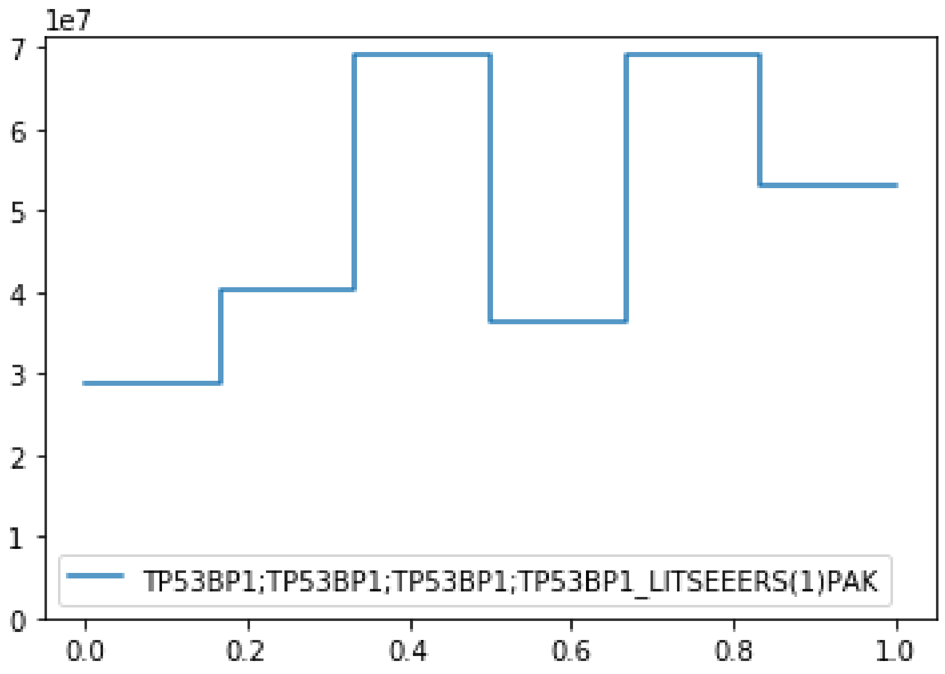

1e7
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0
RB1;RB1_TLQTDSIDS(0.003)FET(0.563)QRT(0.434)PRK

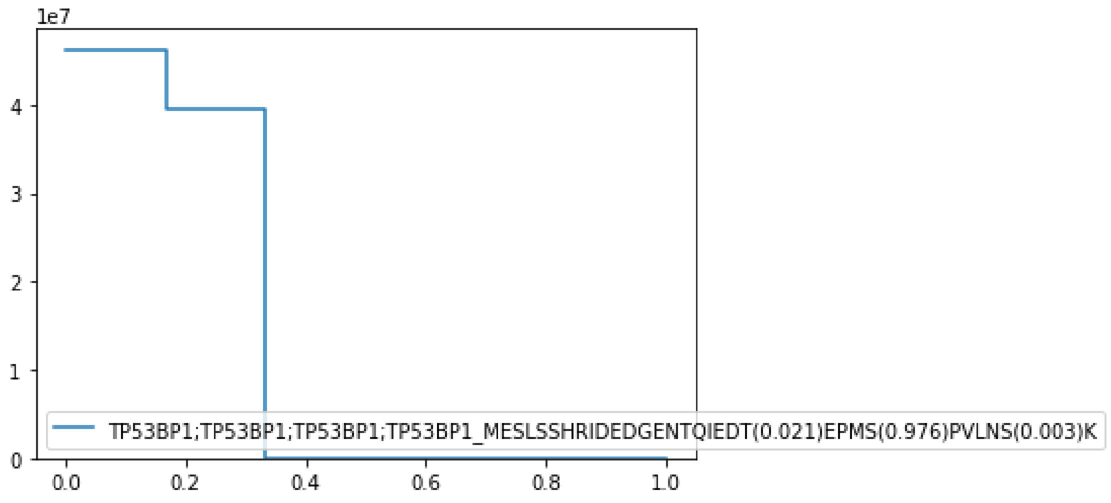

1e7
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
RB1;RB1_TLQTDSIDSFET(0.191)QRT(0.809)PRK

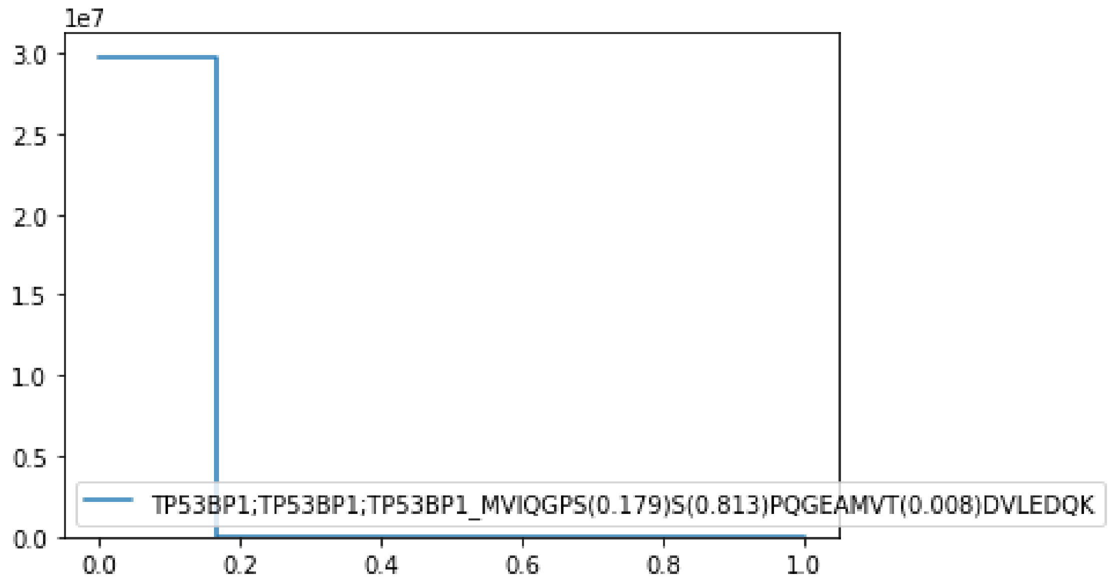

TP53BP1;TP53BP1;TP53BP1_AS(0.006)S(0.076)LHRT(0.254)S(0.254)S(0.254)GT(0.069)S(0.069)LS(0.017)AMHSSGSSGK
0.04
0.03
0.02
0.01
0.00
0.0
0.2
0.4
0.6
0.8
1.0

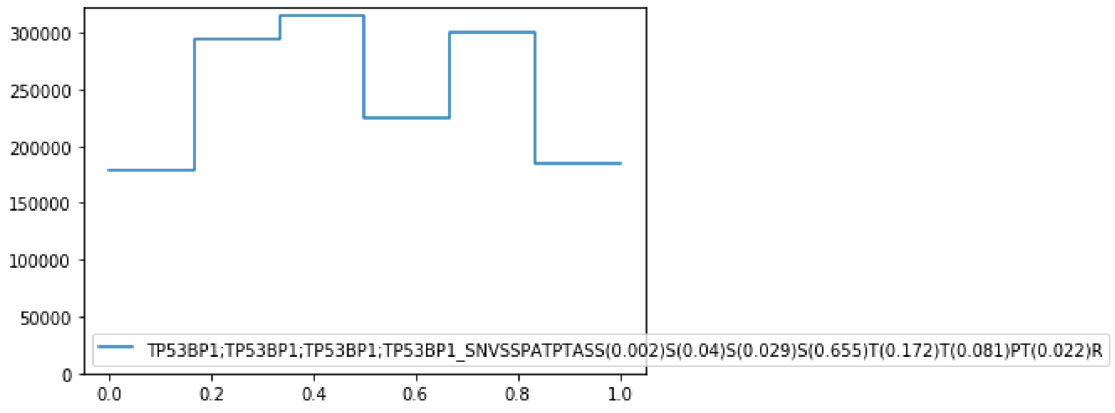

300000
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0
TP53BP1;TP53BP1;TP53BP1;TP53BP1_KNGSTAVAESVAS(1)PQK

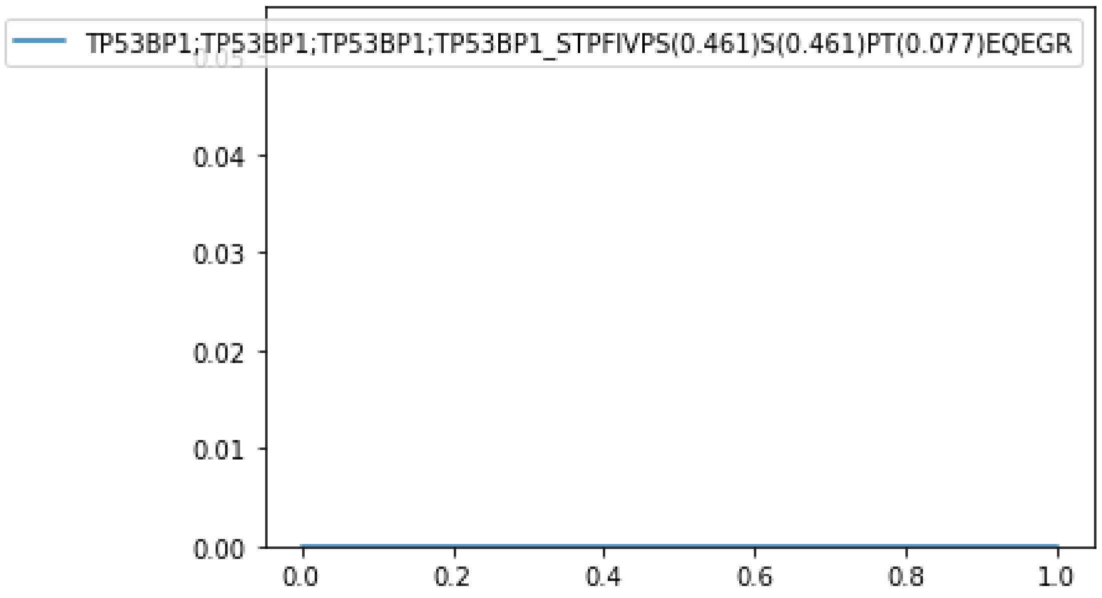

1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
TP53BP1;TP53BP1;TP53BP1;TP53BP1_LIT(0.001)S(0.633)EEERS(0.366)PAK


1e7
7
6
5
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0
TP53BP1;TP53BP1;TP53BP1;TP53BP1_LITSEEERS(1)PAK

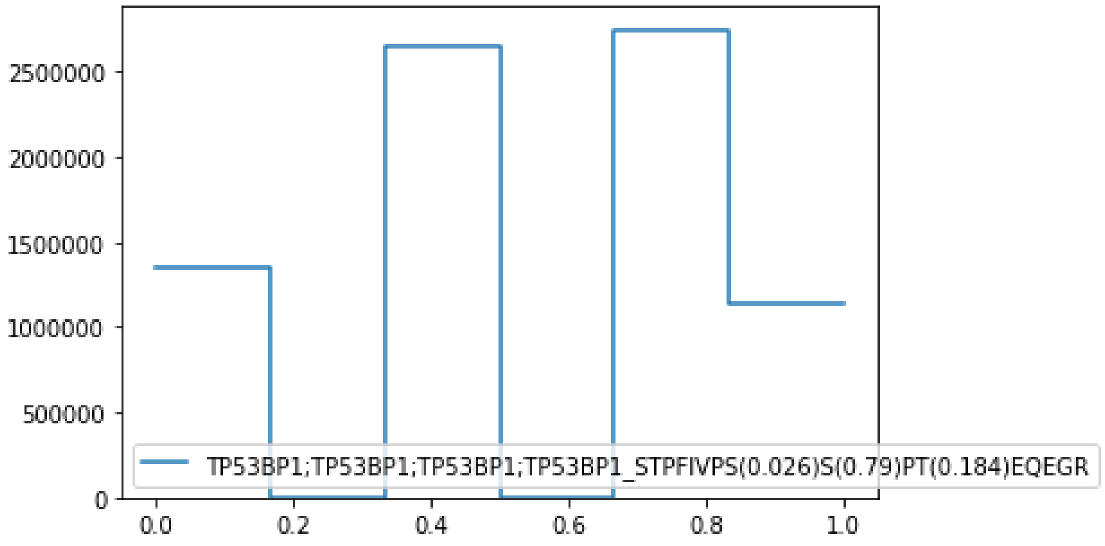

1e7
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0
TP53BP1;TP53BP1;TP53BP1;TP53BP1_MESLSSHRIDEDGENTQIEDT(0.021)EPMS(0.976)PVLNS(0.003)K


1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
TP53BP1;TP53BP1;TP53BP1_MVIQGPS(0.179)S(0.813)PQGEAMVT(0.008)DVLEDQK

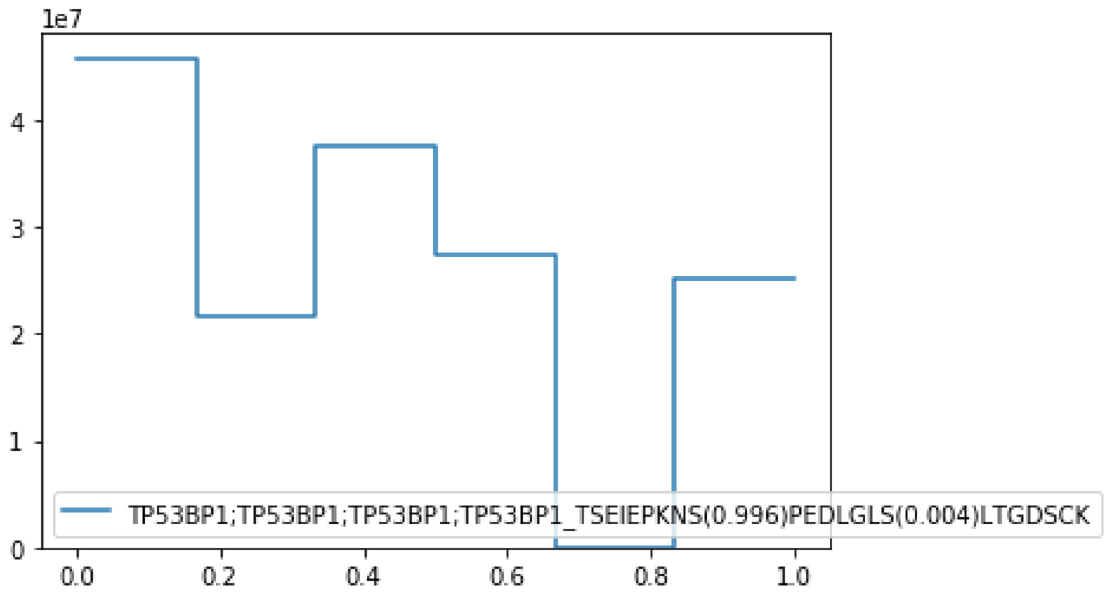

300000
250000
200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0
TP53BP1;TP53BP1;TP53BP1;TP53BP1_SNVSSPATPTASS(0.002)S(0.04)S(0.029)S(0.655)T(0.172)T(0.081)PT(0.022)R

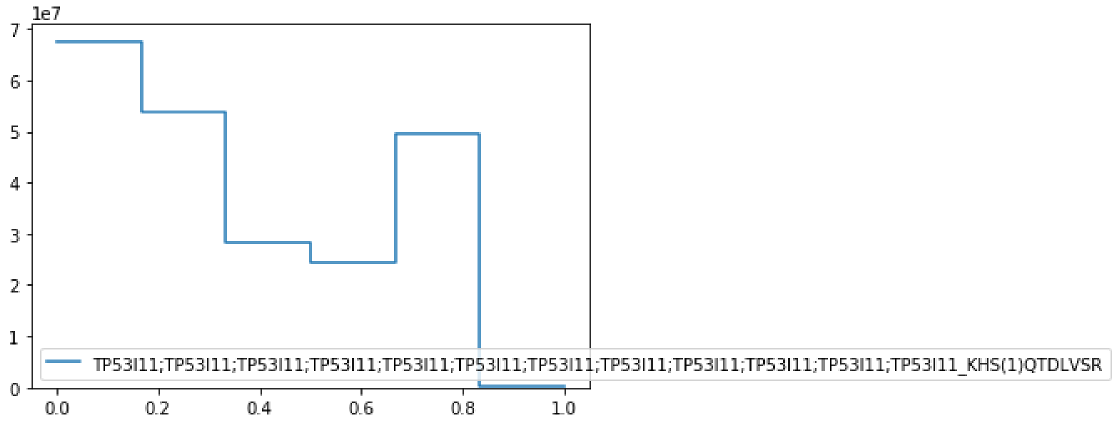


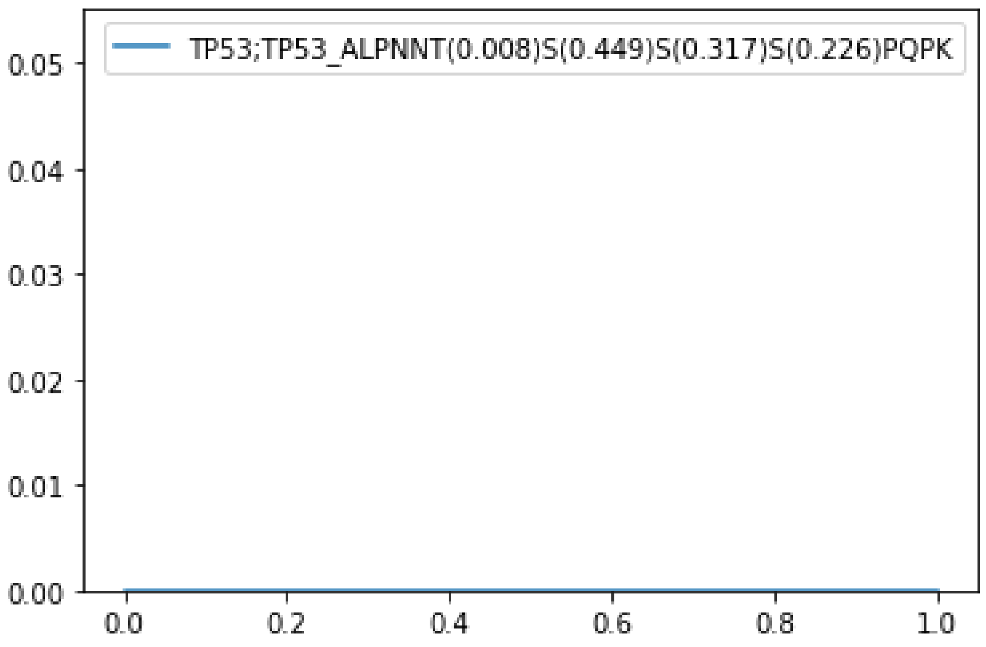
TP53BP1;TP53BP1;TP53BP1;TP53BP1_STPFIVPS(0.461)S(0.461)PT(0.077)EQEGR
0.05
0.04
0.03
0.02
0.01
0.00
0.0
0.2
0.4
0.6
0.8
1.0

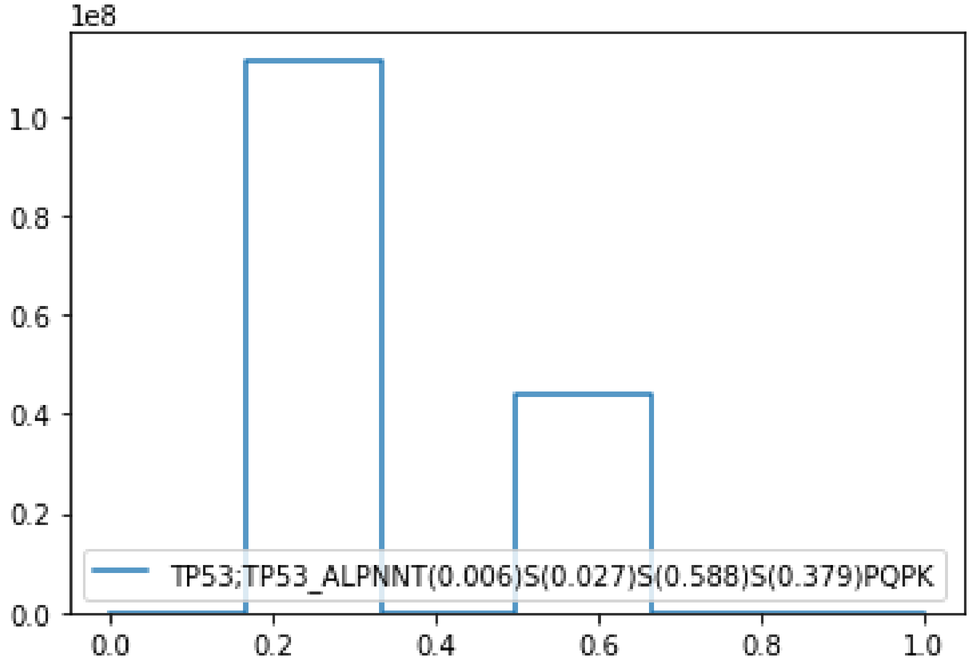


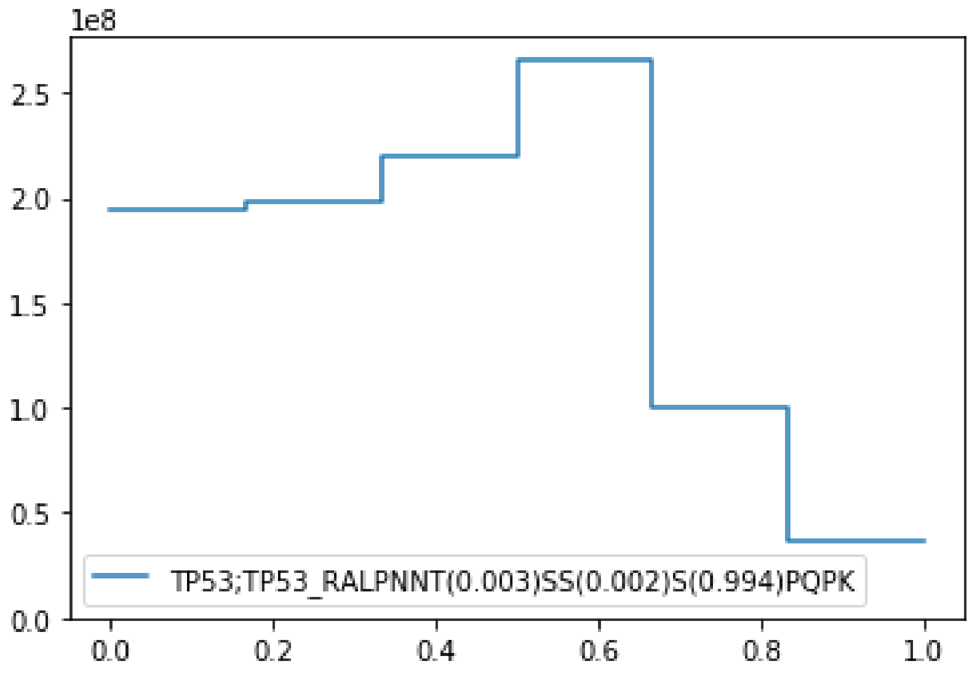
2500000
2000000
1500000
1000000
500000
0
0.0
0.2
0.4
0.6
0.8
1.0
TP53BP1;TP53BP1;TP53BP1;TP53BP1_STPFIVPS(0.026)S(0.79)PT(0.184)EQEGR

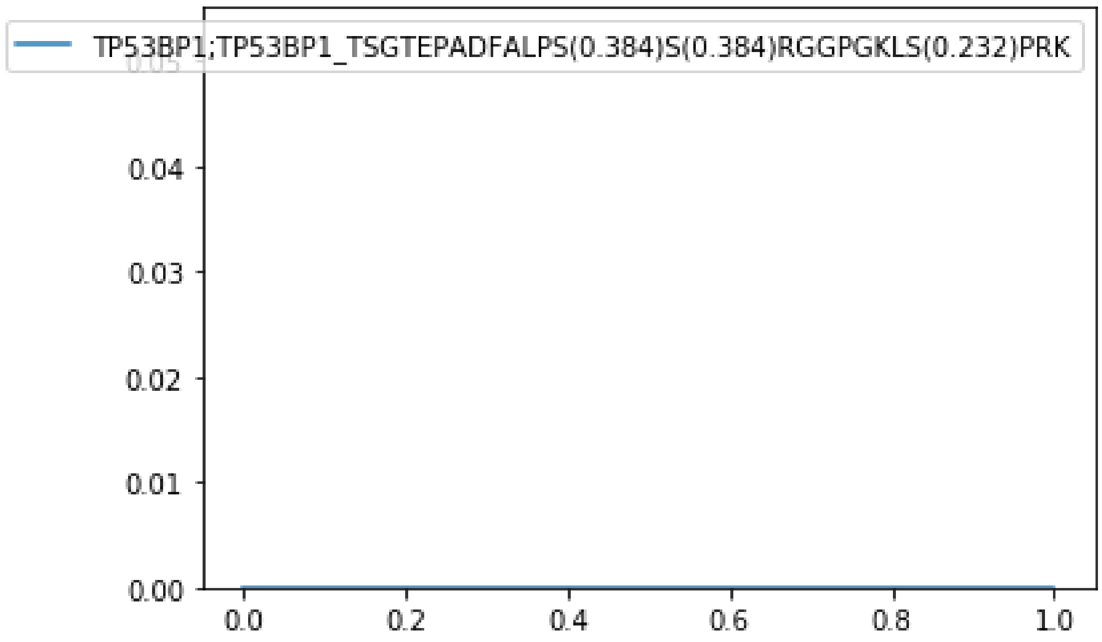

1e7
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0
TP53BP1;TP53BP1;TP53BP1;TP53BP1_TSEIEPKNS(0.996)PEDLGLS(0.004)LTGDSCK

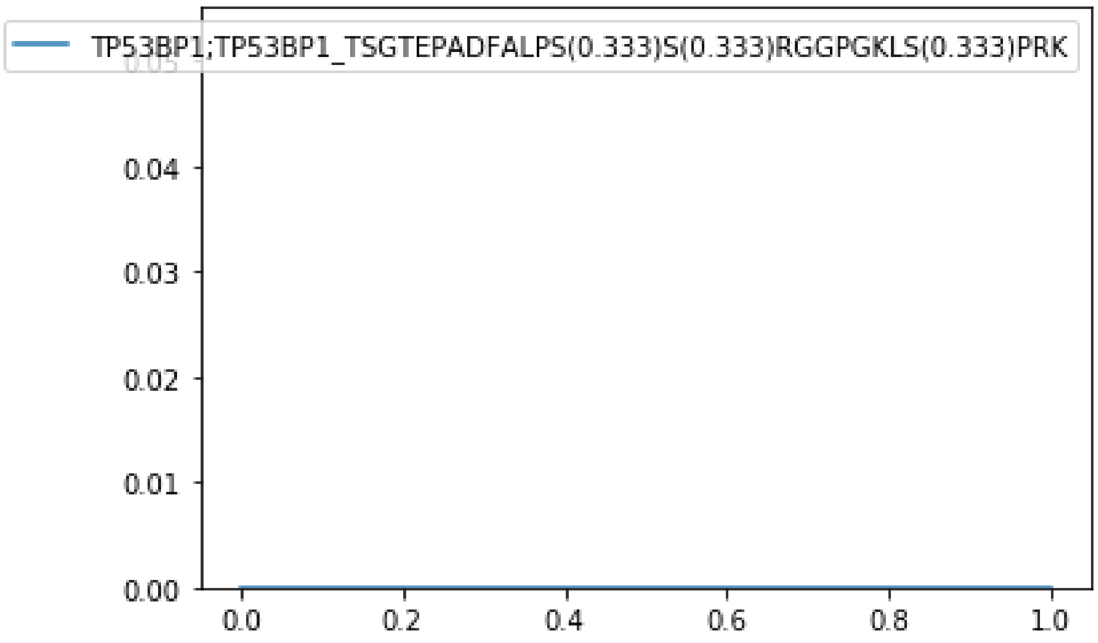

1e7
7
6
5
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0
TP53I11;TP53I11;TP53I11;TP53I11;TP53I11;TP53I11;TP53I11;TP53I11;TP53I11;TP53I11;TP53I11;TP53I11_KHS(1)QTDLVSR

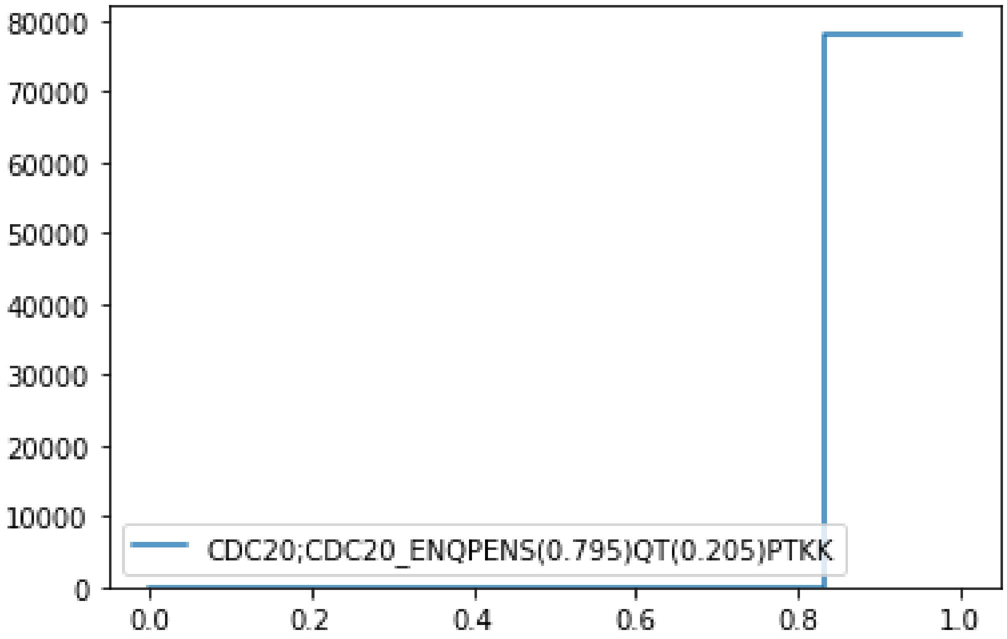

TP53;TP53_ALPNNT(0.008)S(0.449)S(0.317)S(0.226)PQPK
0.05
0.04
0.03
0.02
0.01
0.00
0.0
0.2
0.4
0.6
0.8
1.0

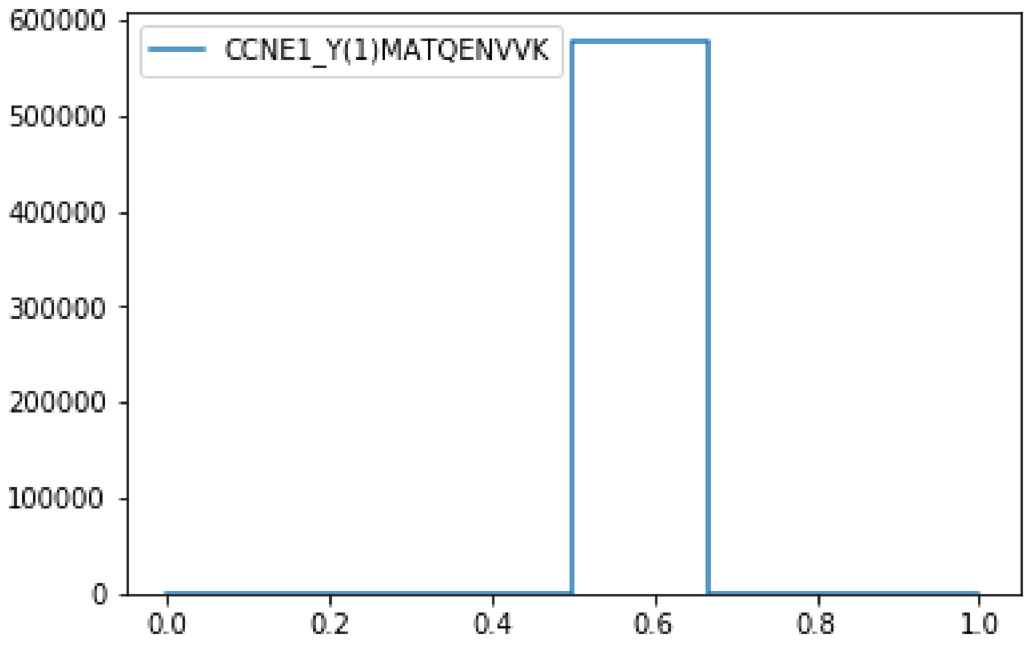

1e8
TP53;TP53_ALPNNT(0.006)S(0.027)S(0.588)S(0.379)PQPK
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

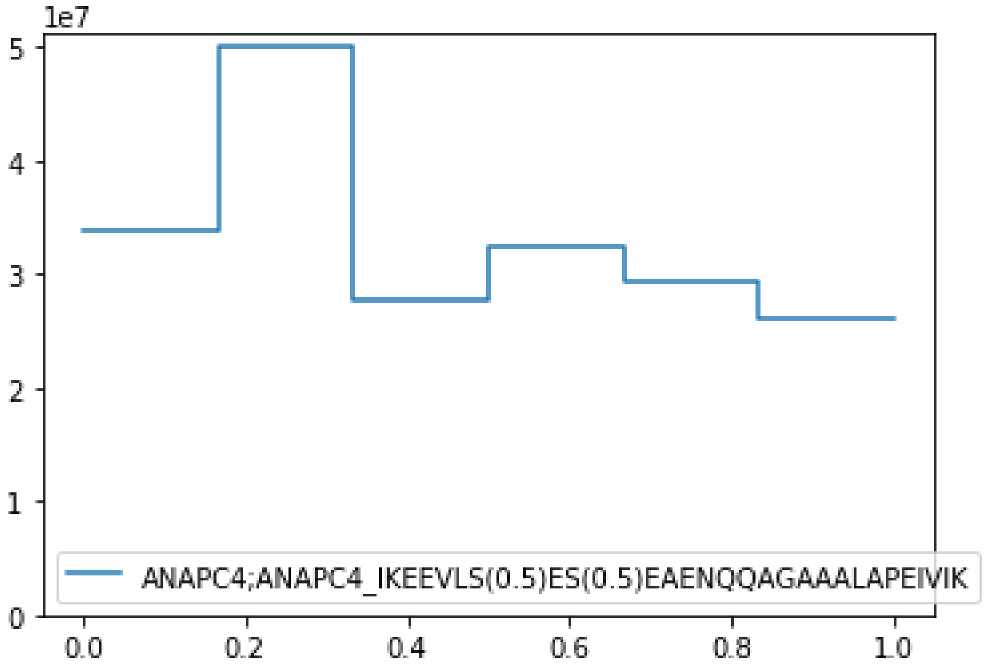

1e8
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
TP53;TP53_RALPNNT(0.003)SS(0.002)S(0.994)PQPK

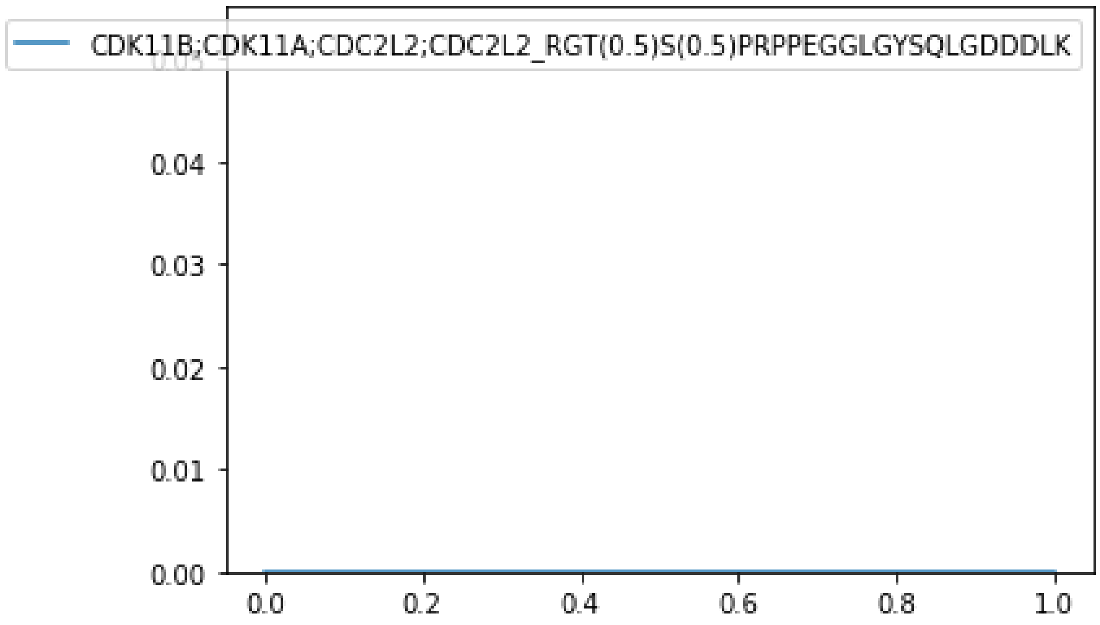

TP53BP1;TP53BP1_TSGTEPADFALPS(0.384)S(0.384)RGGPGKLS(0.232)PRK
0.04
0.03
0.02
0.01
0.00
0.0
0.2
0.4
0.6
0.8
1.0

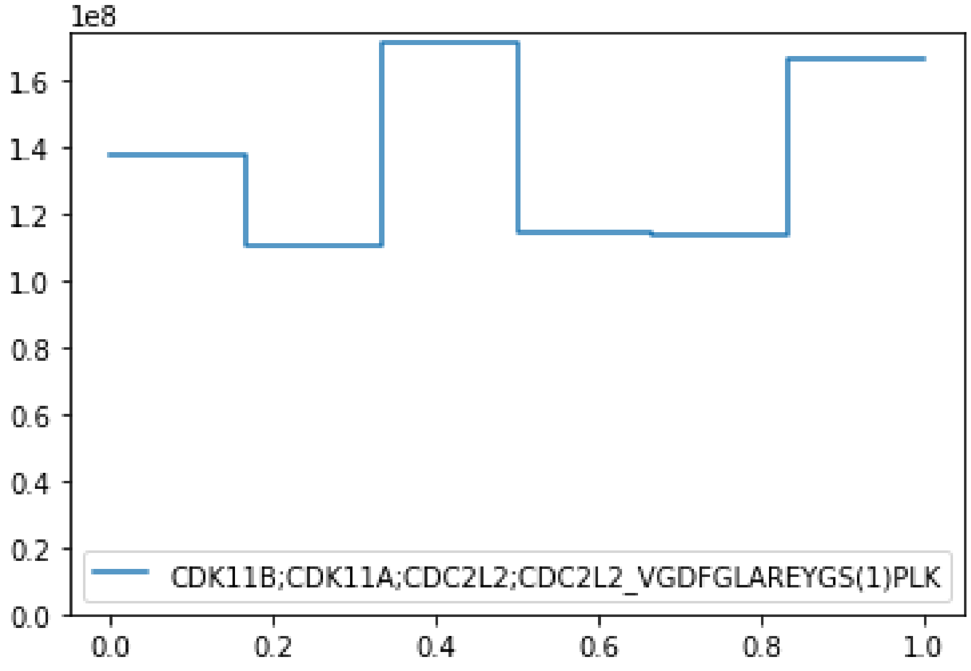

TP53BP1;TP53BP1_TSGTEPADFALPS(0.333)S(0.333)RGGPGKLS(0.333)PRK
0.05
0.04
0.03
0.02
0.01
0.00
0.0
0.2
0.4
0.6
0.8
1.0

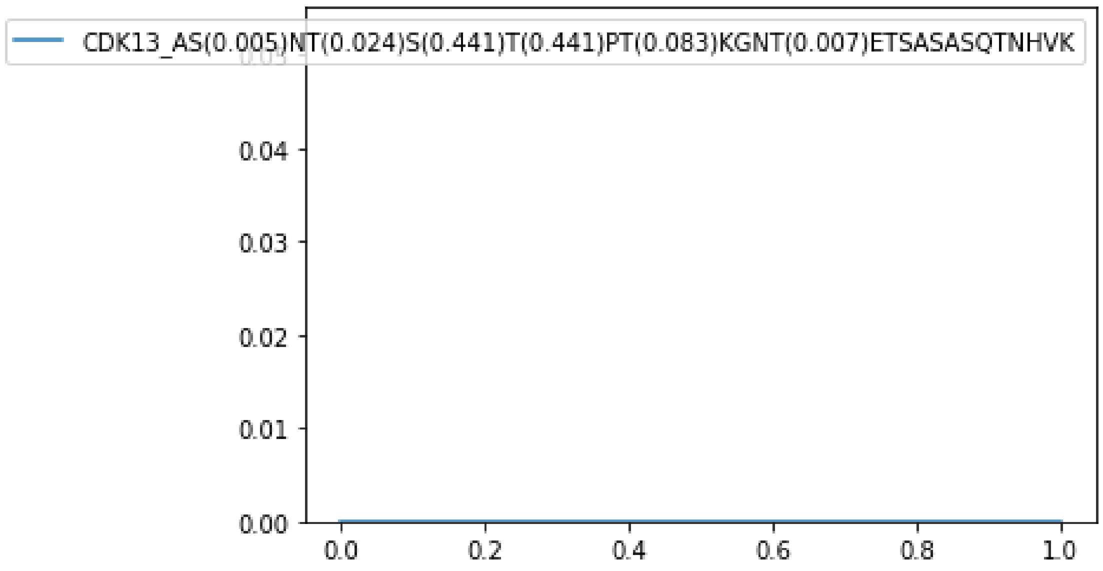

80000
70000
60000
50000
40000
30000
20000
10000
0
CDC20;CDC20_ENQPENS(0.795)QT(0.205)PTKK
0.0
0.2
0.4
0.6
0.8
1.0

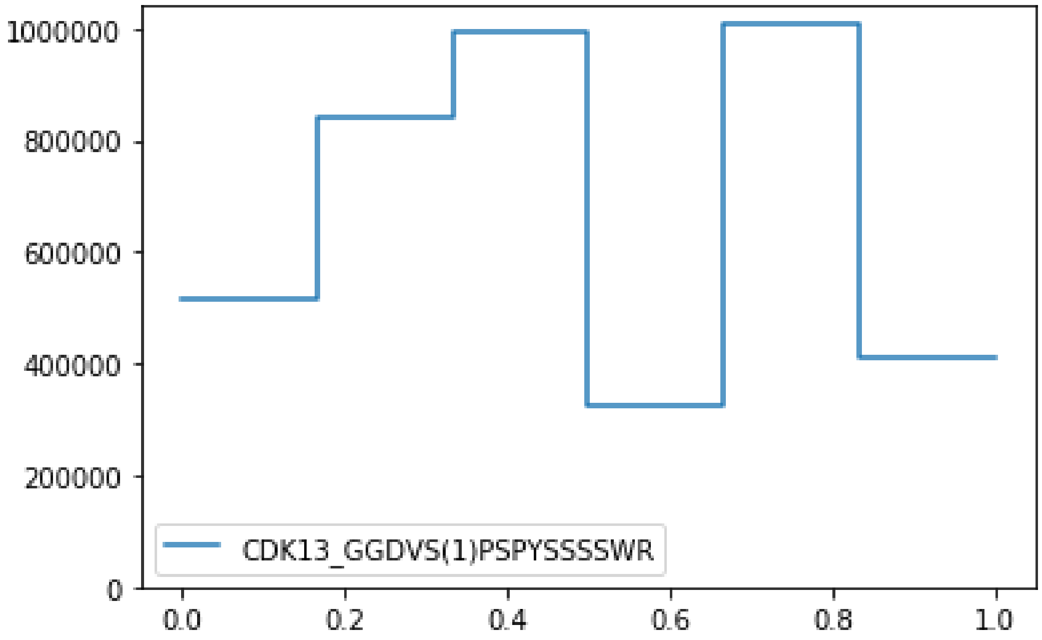

CCNE1_Y(1)MATQENVVK
600000
500000
400000
300000
200000
100000
0
0.0
0.2
0.4
0.6
0.8
1.0

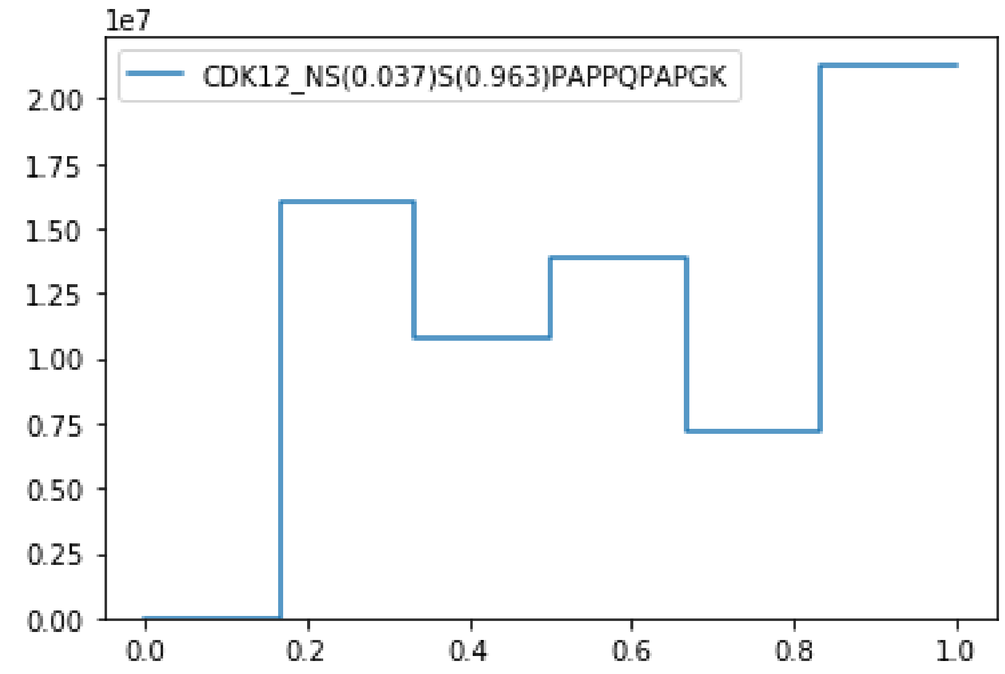

1e7
5
4
3
2
1
0
ANAPC4;ANAPC4_IKEEVLS(0.5)ES(0.5)EAENQQAGAAALAPEIVIK
0.0
0.2
0.4
0.6
0.8
1.0

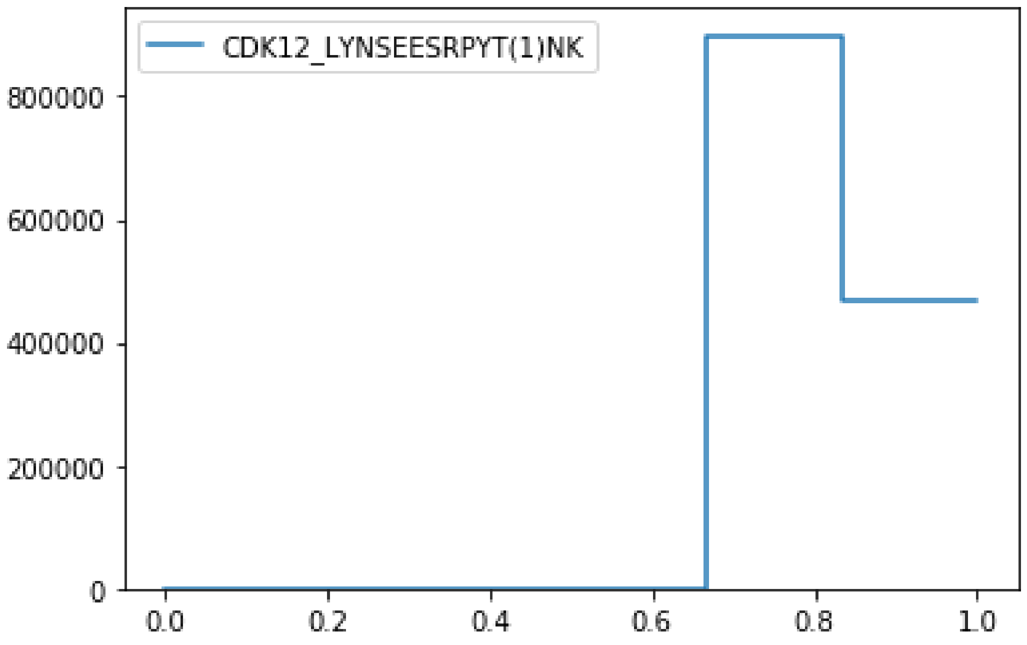

CDK11B;CDK11A;CDC2L2;CDC2L2_RGT(0.5)S(0.5)PRPPEGGLGYSQLGDDDLK
0.05
0.04
0.03
0.02
0.01
0.00
0.0
0.2
0.4
0.6
0.8
1.0

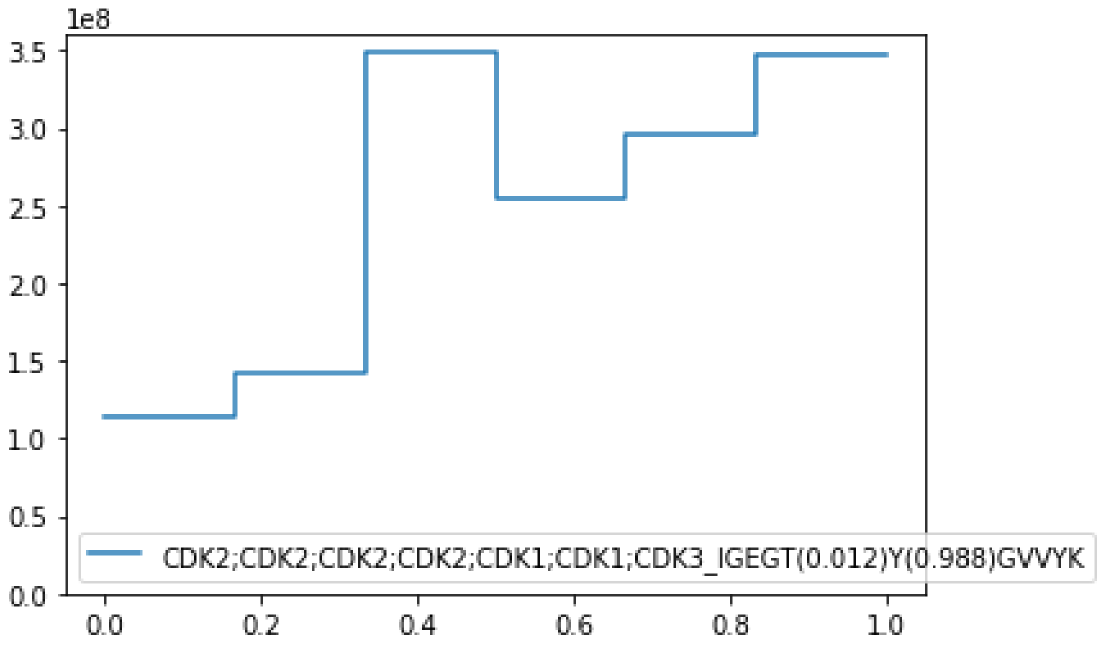

1e8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
CDK11B;CDK11A;CDC2L2;CDC2L2_VGDFGLAREYGS(1)PLK
0.0
0.2
0.4
0.6
0.8
1.0

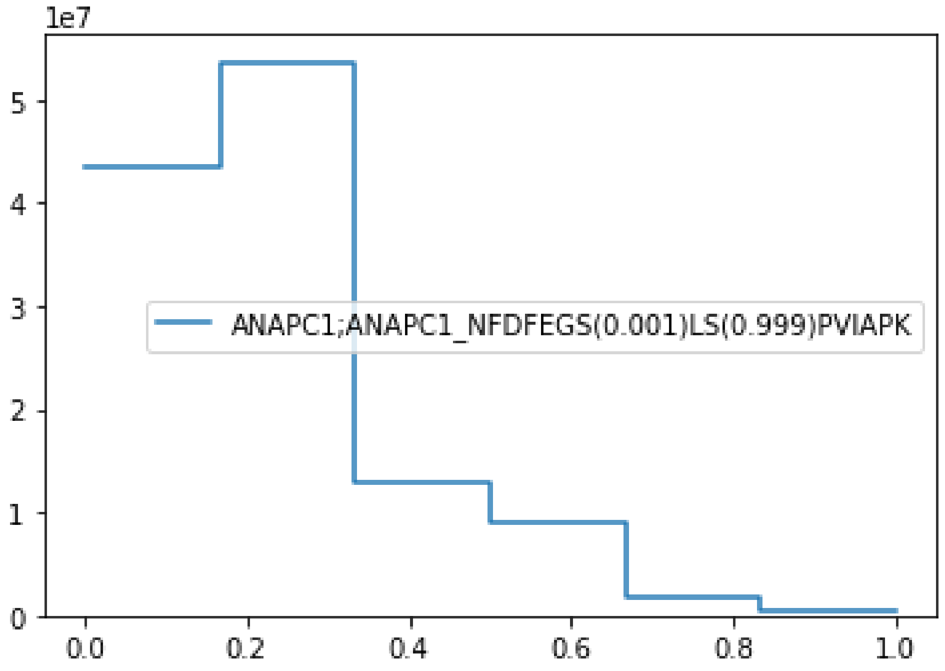

CDK13_AS(0.005)NT(0.024)S(0.441)T(0.441)PT(0.083)KGNT(0.007)ETSASASQTNHVK
0.05
0.04
0.03
0.02
0.01
0.00
0.0
0.2
0.4
0.6
0.8
1.0

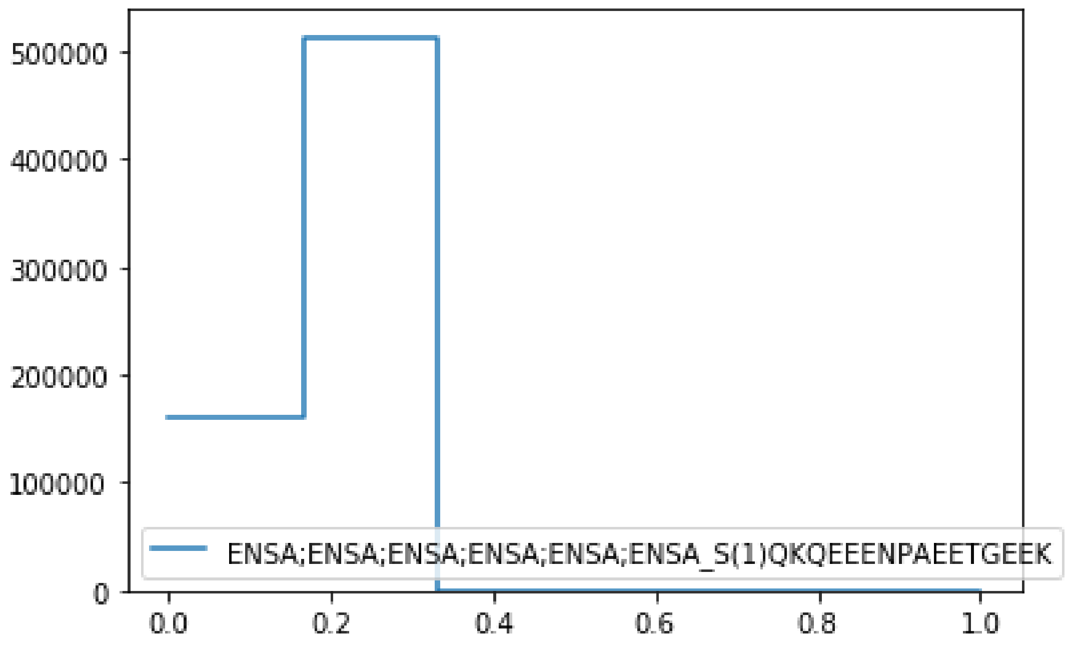

1000000
800000
600000
400000
200000
0
CDK13_GGDVS(1)PSPYSSSSWR
0.0
0.2
0.4
0.6
0.8
1.0

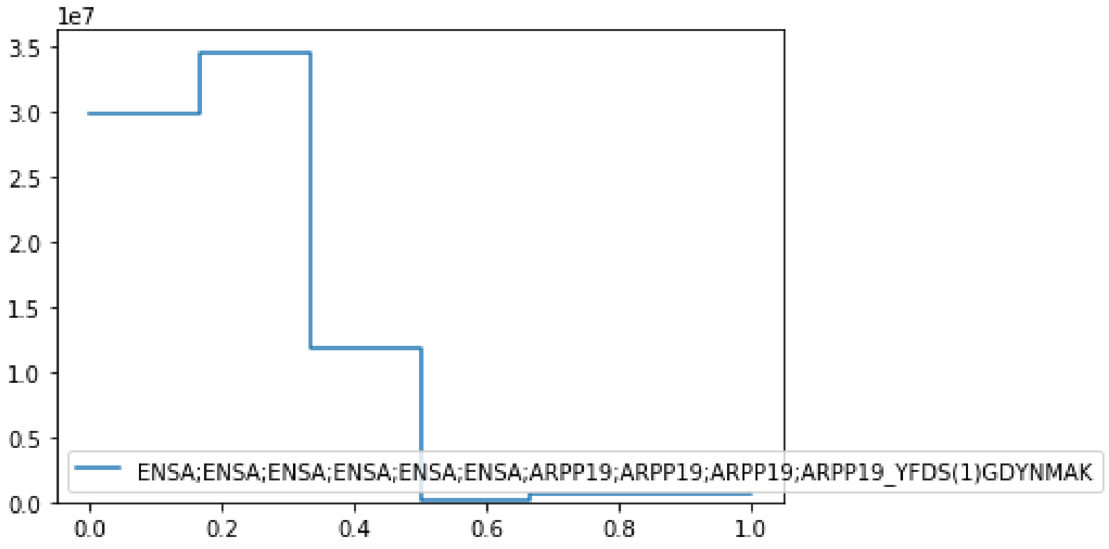

1e7
CDK12_NS(0.037)S(0.963)PAPPQPAPGK
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0


CDK12_LYNSEESRPYT(1)NK
800000
600000
400000
200000
0
0.0
0.2
0.4
0.6
0.8
1.0


1e8
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
CDK2;CDK2;CDK2;CDK2;CDK1;CDK1;CDK3_IGEGT(0.012)Y(0.988)GVVYK
0.0
0.2
0.4
0.6
0.8
1.0


1e7
5
4
3
2
1
0
ANAPC1;ANAPC1_NFDFEGS(0.001)LS(0.999)PVIAPK
0.0
0.2
0.4
0.6
0.8
1.0





Conclusion: * Some useful information on p53, CDC20, CCNE (the latter showing highest phosphorylation midway through the cell cycle, suggesting that it is not immediately degraded via SCF). * The measure phosphosites on RB, ENSA and APC do not look like the ones I modelled * Some peptides look erroneus

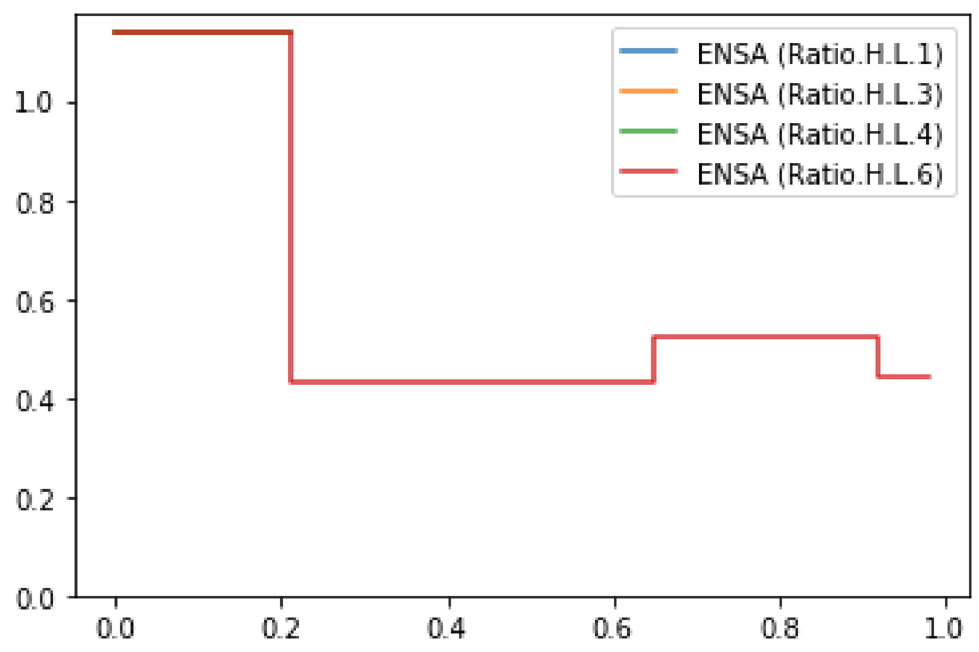
3 Ly et al. 2017 - FACS

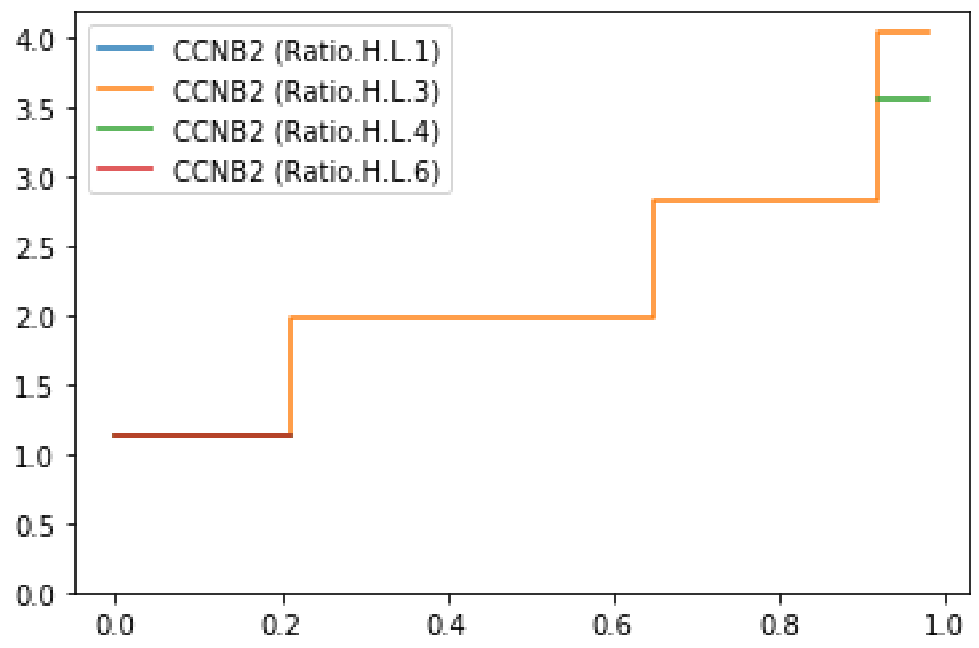
```
[ ]: ##################
     # FACS G1-S-G2-M
     ##################

     import pandas as pd
     import os
     import numpy as np
     import matplotlib.pyplot as plt
     from ast import literal_eval

     # Enter (part of) gene names to plot
     display = []
     # Do you want to plot all replicates or only the median?
     plot_median = False

     directory = os.path.abspath('')
     if not display:
         data_file = os.path.join(directory, 'merged.xlsx')
         df = pd.read_excel(data_file)
         display = list(set(df.loc[:, 'Gene name']))
         print(df.loc[:, ['Gene name', 'ccModel Paul ID']].drop_duplicates().
      ↪dropna())
         display = [val for val in display if isinstance(val, str)]

     # Fetch data of interest and prepare for plotting
     data_file = os.path.join(directory, 'elife-27574-supp2-v1.xlsx')
     df = pd.read_excel(data_file)

     genes_of_interest = []
     for gene in list(df.loc[:, 'gene_ids']):
         if gene in display:
             genes_of_interest.append(gene)

     rows_of_interest = [True if item in genes_of_interest else False for item in␣
      ↪list(df['gene_ids'])]
     df_of_interest = df.loc[rows_of_interest]

     time = [0, 0.21, 0.21, 0.65, 0.65, 0.92, 0.92, 0.98]

     data_to_plot = {}
     if plot_median:
         for gene in genes_of_interest:
             row = df_of_interest.loc[df_of_interest['gene_ids'] == gene]
             abundances = []
             for cc_phase in ['g1', 's', 'g2', 'm']:
```

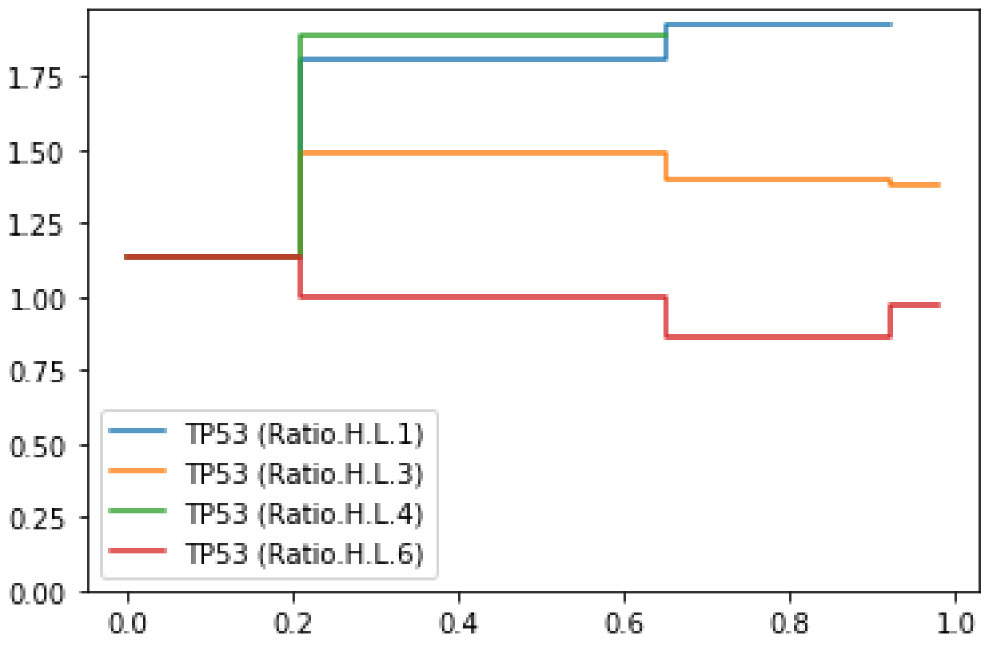
```
            column_name = '{}.median'.format(cc_phase)
            abundances.append(row.iloc[0][column_name])
        doubled = []
        for val in abundances:
            doubled.extend([val, val])
        data_to_plot[gene] = doubled

    for gene in data_to_plot:
        plt.figure()
        plt.plot(time, data_to_plot[gene])
        plt.gca().set_ylim(bottom=0)
        plt.gca().legend((gene,))

else:
    replicates = ['Ratio.H.L.1', 'Ratio.H.L.3', 'Ratio.H.L.4', 'Ratio.H.L.6']
    for gene in genes_of_interest:
        row = df_of_interest.loc[df_of_interest['gene_ids'] == gene]
        dict_of_replicates = {}
        for item in replicates:
            abundances = []
            for cc_phase in ['g1', 's', 'g2', 'm']:
                column_name = '{}.{}'.format(item, cc_phase)
                abundances.append(row.iloc[0][column_name])
            doubled = []
            for val in abundances:
                doubled.extend([val, val])
            dict_of_replicates[item] = doubled
        data_to_plot[gene] = dict_of_replicates

    for gene in data_to_plot:
        plt.figure()
        for replicate in replicates:
            plt.plot(time, data_to_plot[gene][replicate])
        plt.gca().set_ylim(bottom=0)
        plt.gca().legend((gene+' ('+replicates[0]+')', gene+'␣
↪('+replicates[1]+')',
                          gene+' ('+replicates[2]+')', gene+'␣
↪('+replicates[3]+')'))
```

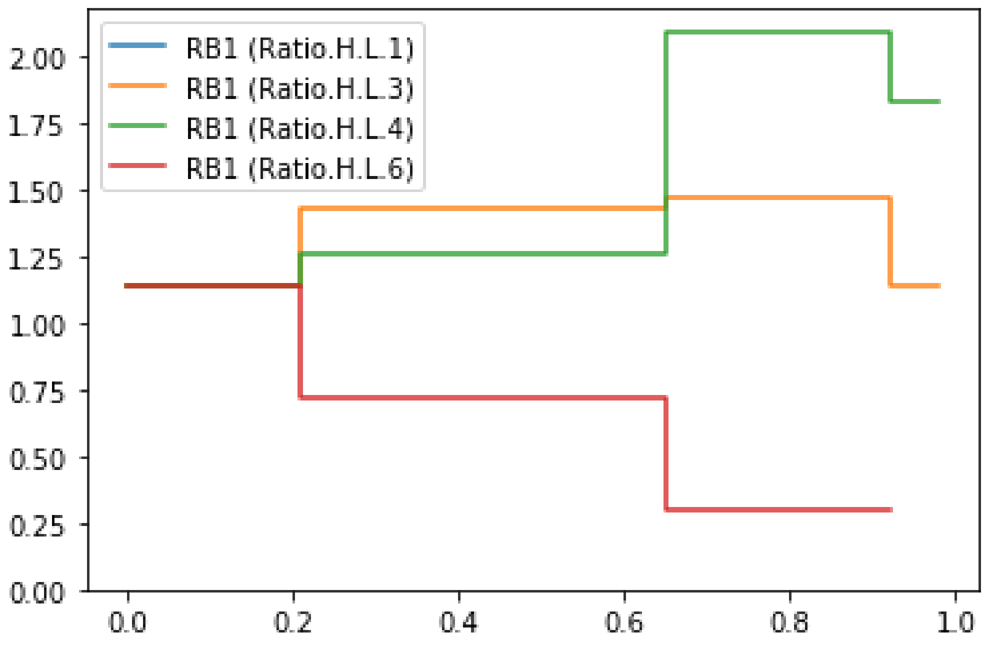
```
   Gene name ccModel Paul ID
0      CDC27             Apc
20      E2F2             E2f
22    ANAPC4             Apc
33   PPP2R2C             B55
42    ANAPC5             Apc
66     CDC23             Apc
75    CDC25B           Cdc25
```

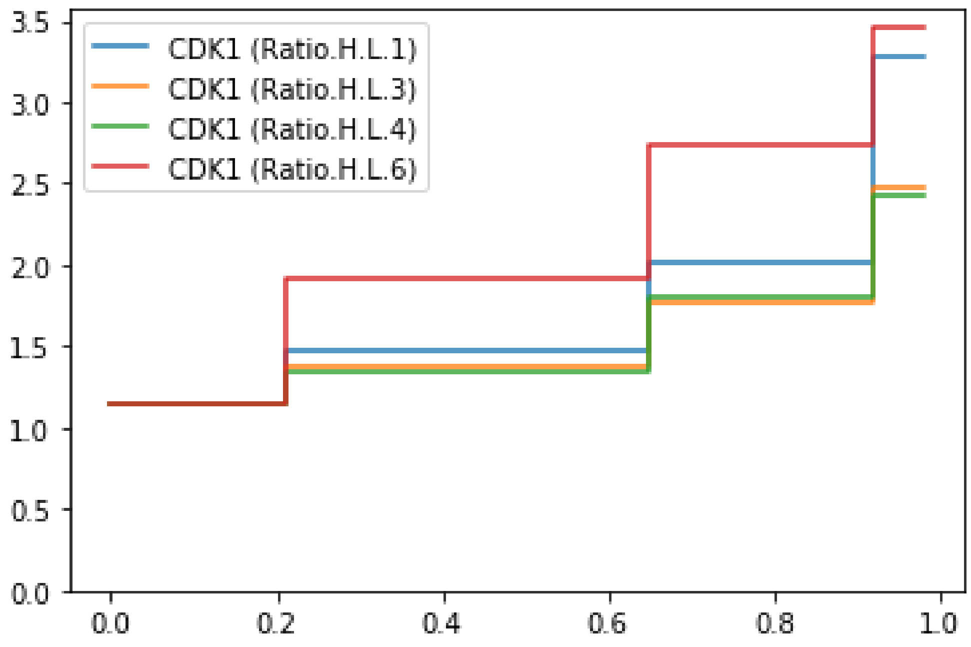
84	E2F1	E2f
85	CCNE1	Ce
92	FZR1	Cdh
100	CCND1	Cd
112	FOXM1	Fox
122	CDKN1B	p27
126	FBXO5	Emi
129	E2F3	E2f
132	CCND3	Cd
155	CDC20	Cdc20
159	CCND2	Cd
163	MASTL	Gw
167	CDKN1A	p21
175	ARPP19	Ensa
188	CDC16	Apc
200	CCNA1	Ca
207	CCNB1	Cb
214	RB1	Rb
220	TP53	p53
248	ANAPC11	Apc
271	ENSA	Ensa
286	CCNA2	Ca
288	CCNB3	Cb
296	ANAPC1	Apc
306	PPP2R2B	B55
329	CCNB2	Cb
334	CDC25C	Cdc25
345	CDC25A	Cdc25
350	ANAPC10	Apc
367	WEE1	Wee
375	CDK1	pCb
384	CCNE2	Ce
392	PPP2R2D	B55
400	ANAPC2	Apc
408	CDC26	Apc
412	ANAPC7	Apc
424	PPP2R2A	B55

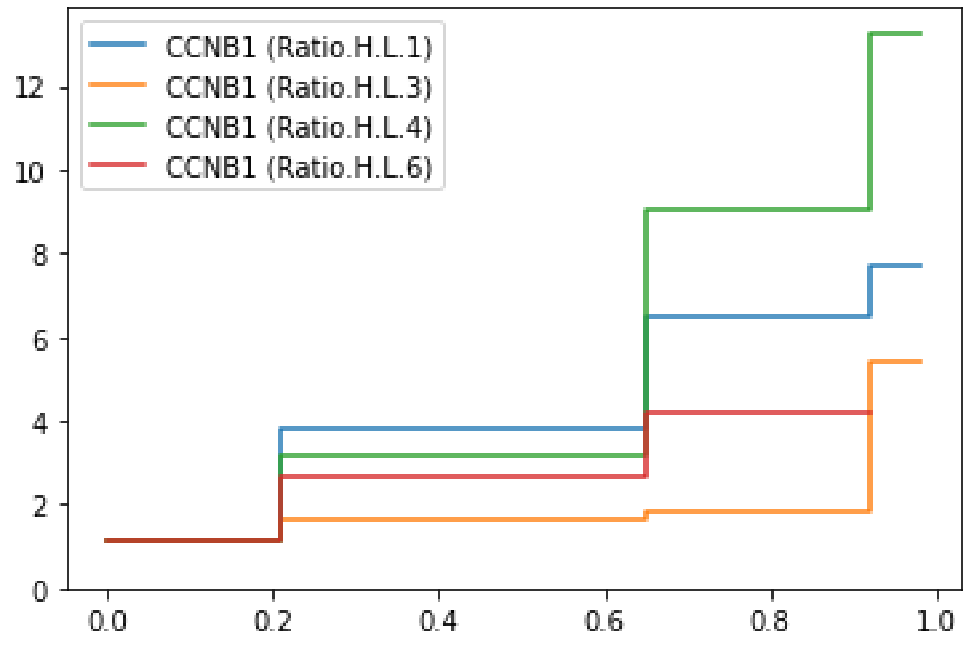

ENSA (Ratio.H.L.1)
ENSA (Ratio.H.L.3)
ENSA (Ratio.H.L.4)
ENSA (Ratio.H.L.6)
0.0
0.2
0.4
0.6
0.8
1.0

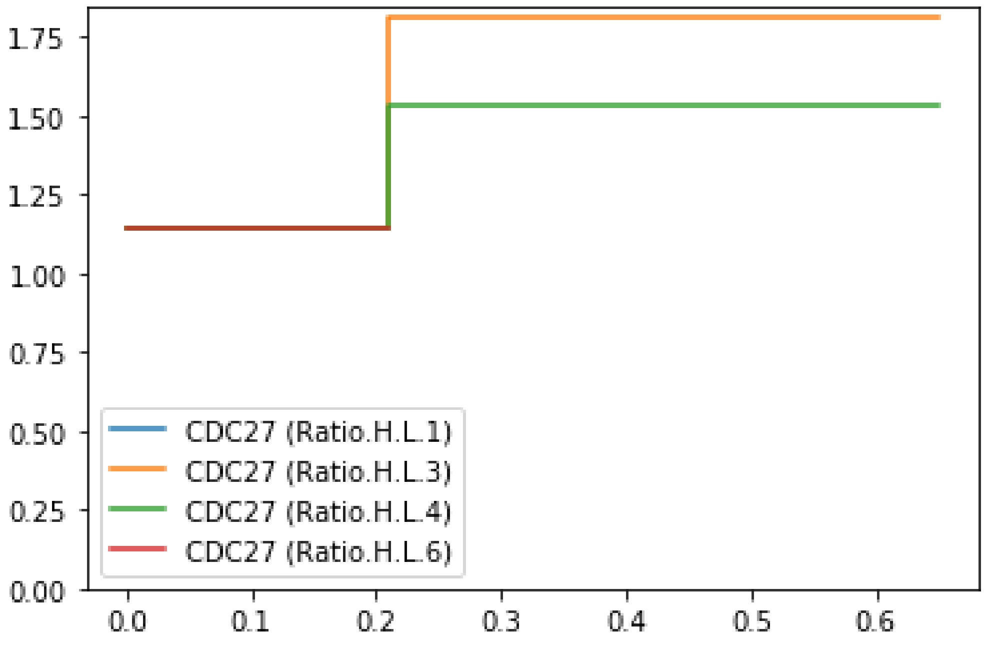

CCNB2 (Ratio.H.L.1)
CCNB2 (Ratio.H.L.3)
CCNB2 (Ratio.H.L.4)
CCNB2 (Ratio.H.L.6)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0.2
0.4
0.6
0.8

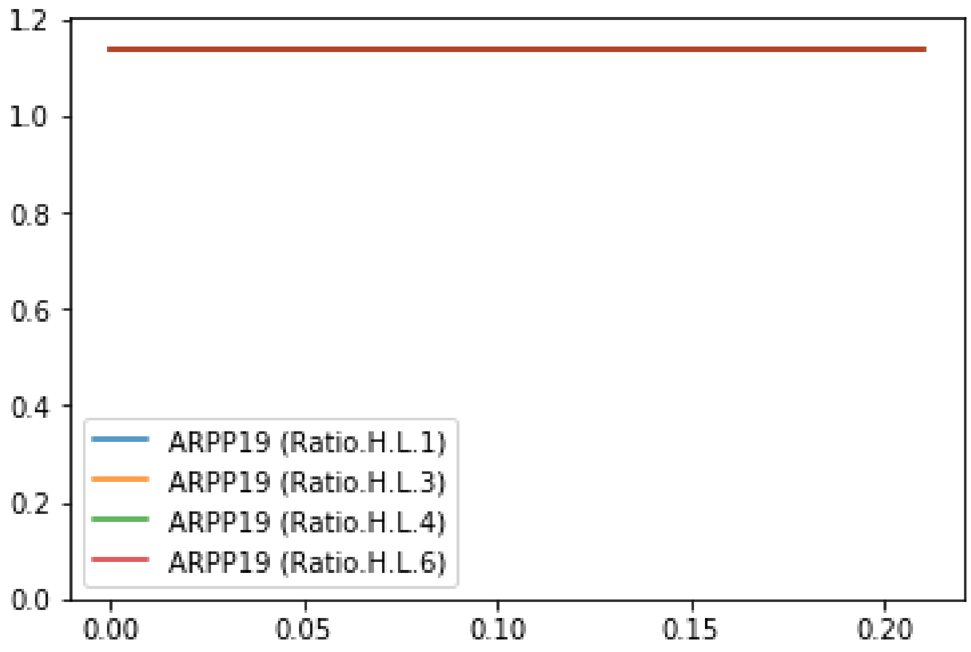

TP53 (Ratio.H.L.1)
TP53 (Ratio.H.L.3)
TP53 (Ratio.H.L.4)
TP53 (Ratio.H.L.6)
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0

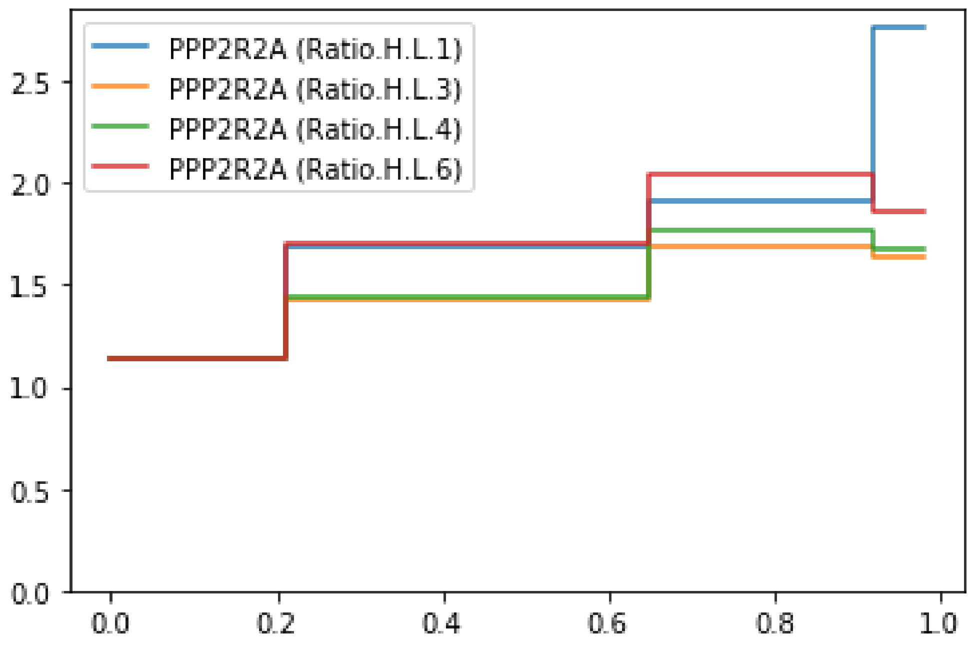

RB1 (Ratio.H.L.1)
RB1 (Ratio.H.L.3)
RB1 (Ratio.H.L.4)
RB1 (Ratio.H.L.6)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0


CDK1 (Ratio.H.L.1)
CDK1 (Ratio.H.L.3)
CDK1 (Ratio.H.L.4)
CDK1 (Ratio.H.L.6)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0.0
0.2
0.4
0.6
0.8
1.0

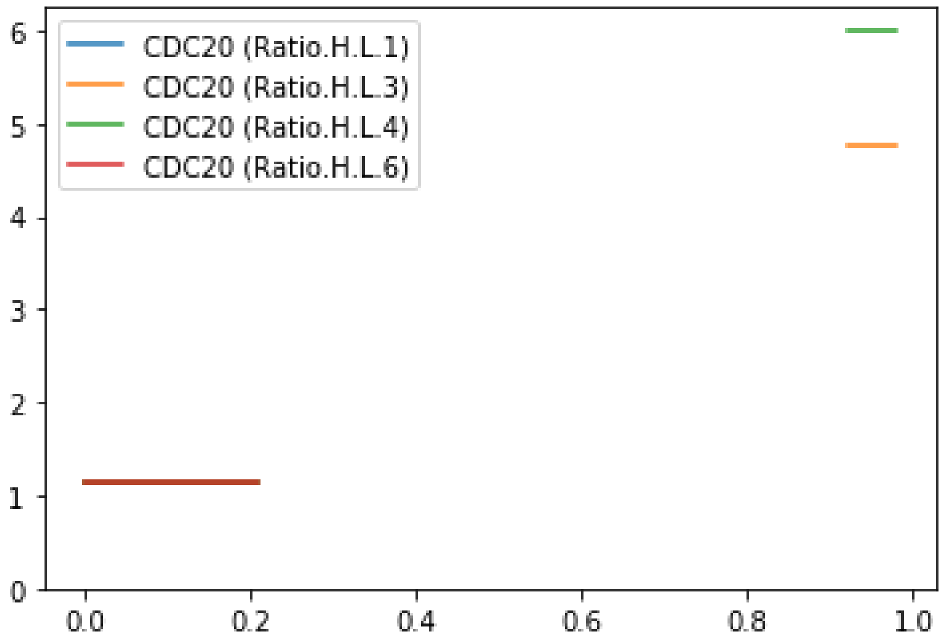

CCNB1 (Ratio.H.L.1)
CCNB1 (Ratio.H.L.3)
CCNB1 (Ratio.H.L.4)
CCNB1 (Ratio.H.L.6)
0
2
4
6
8
10
12
0.0
0.2
0.4
0.6
0.8
1.0


1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.1
0.2
0.3
0.4
0.5
0.6
CDC27 (Ratio.H.L.1)
CDC27 (Ratio.H.L.3)
CDC27 (Ratio.H.L.4)
CDC27 (Ratio.H.L.6)

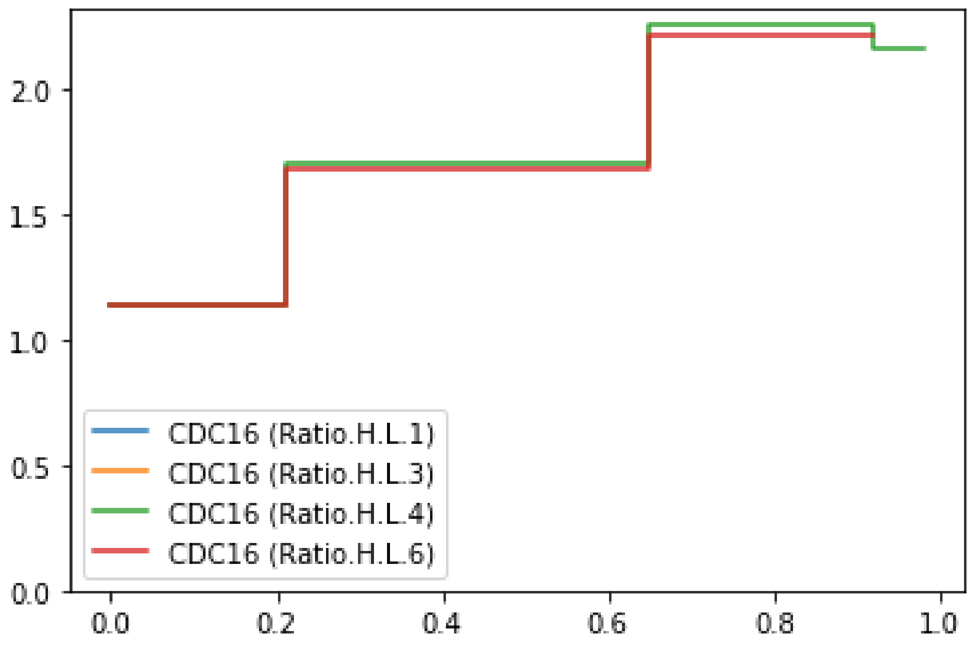

1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.05
0.10
0.15
0.20
ARPP19 (Ratio.H.L.1)
ARPP19 (Ratio.H.L.3)
ARPP19 (Ratio.H.L.4)
ARPP19 (Ratio.H.L.6)

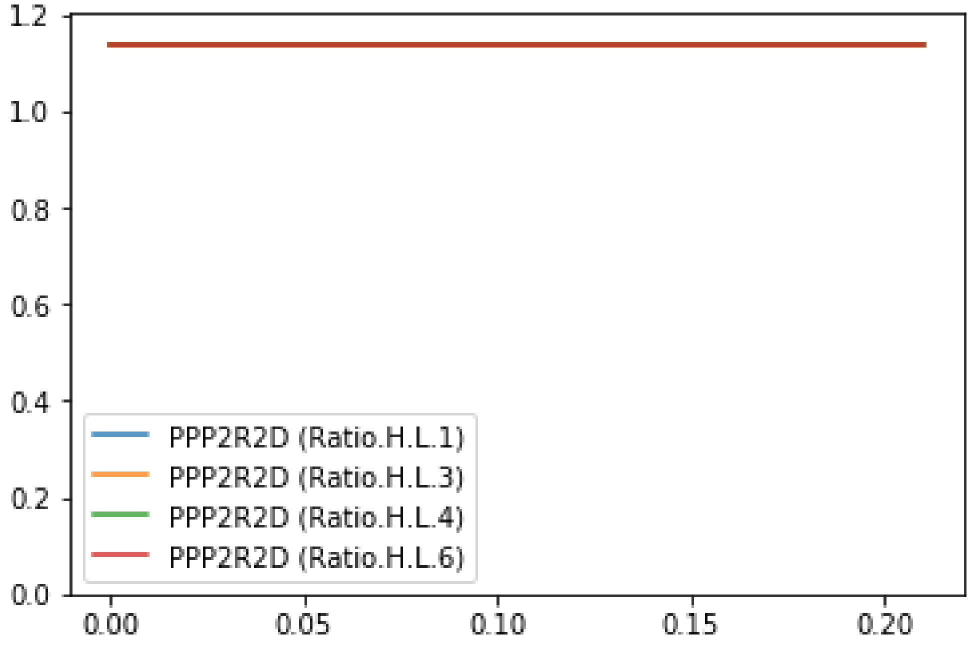


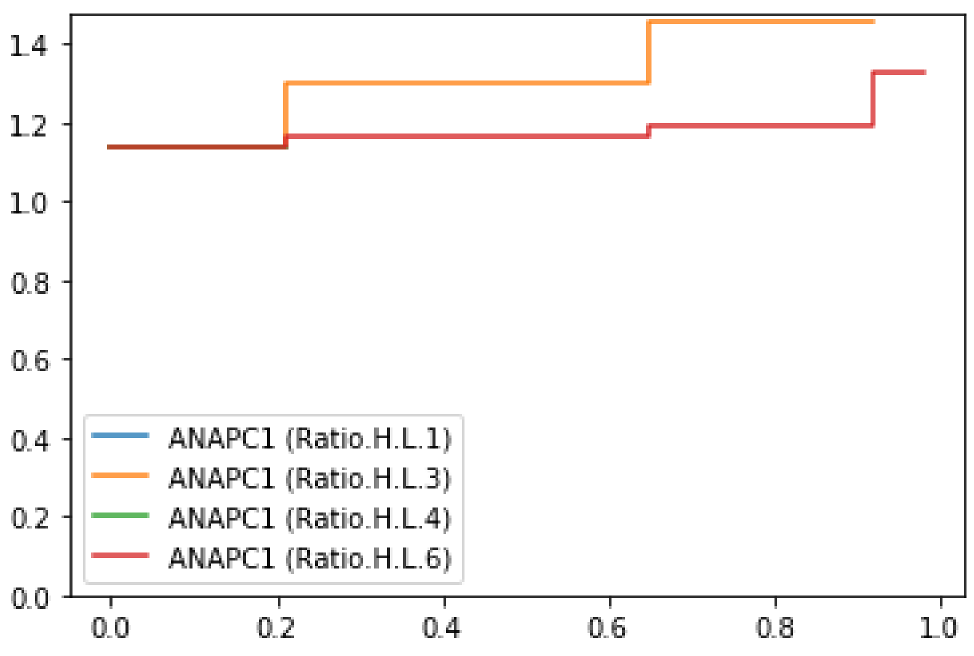
PPP2R2A (Ratio.H.L.1)
PPP2R2A (Ratio.H.L.3)
PPP2R2A (Ratio.H.L.4)
PPP2R2A (Ratio.H.L.6)
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0

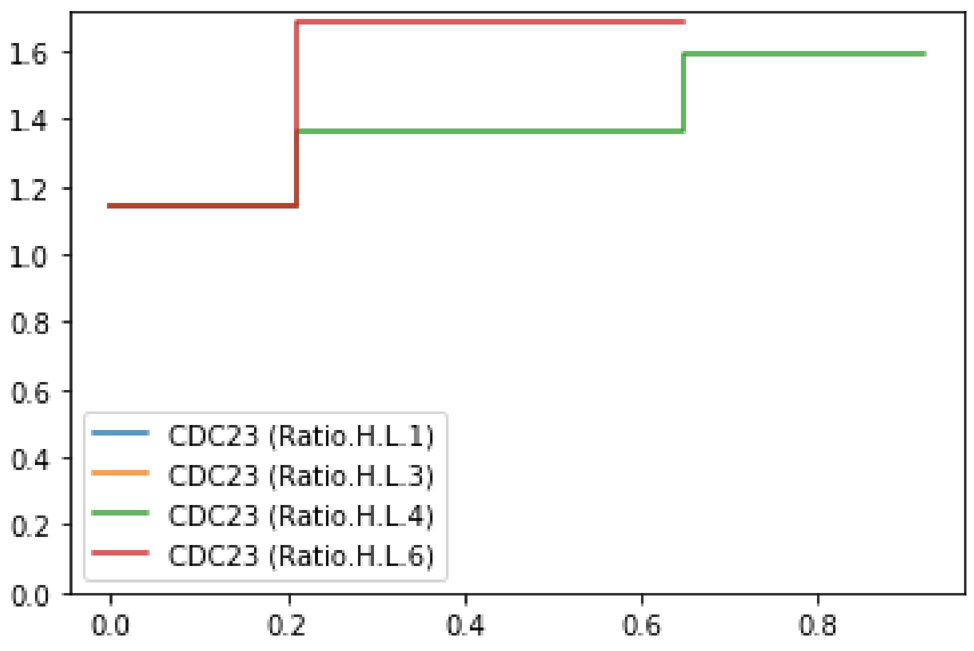


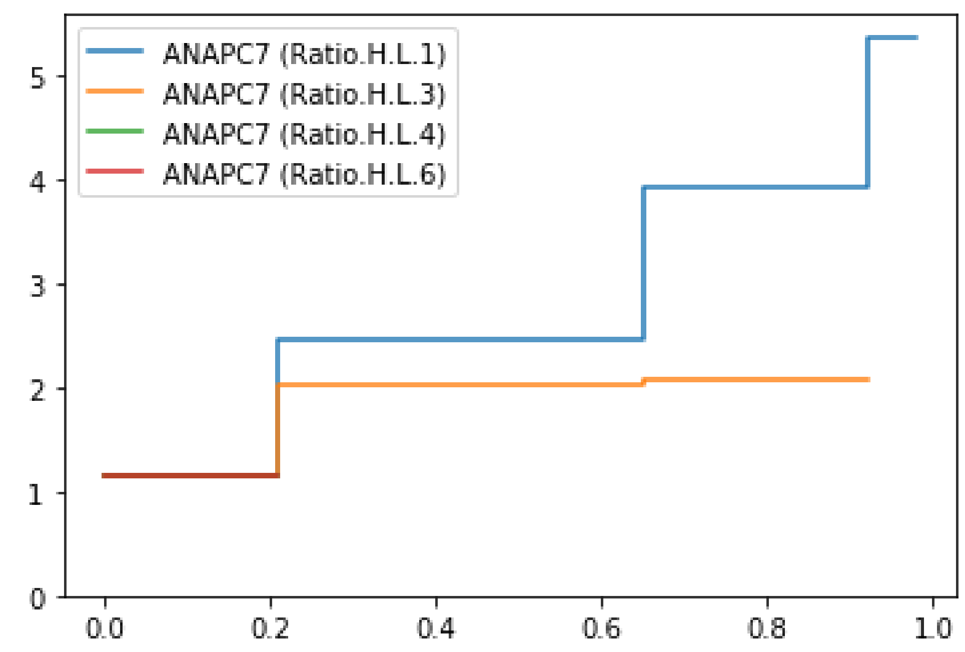
CDC20 (Ratio.H.L.1)
CDC20 (Ratio.H.L.3)
CDC20 (Ratio.H.L.4)
CDC20 (Ratio.H.L.6)
0
1
2
3
4
5
6
0.0
0.2
0.4
0.6
0.8
1.0

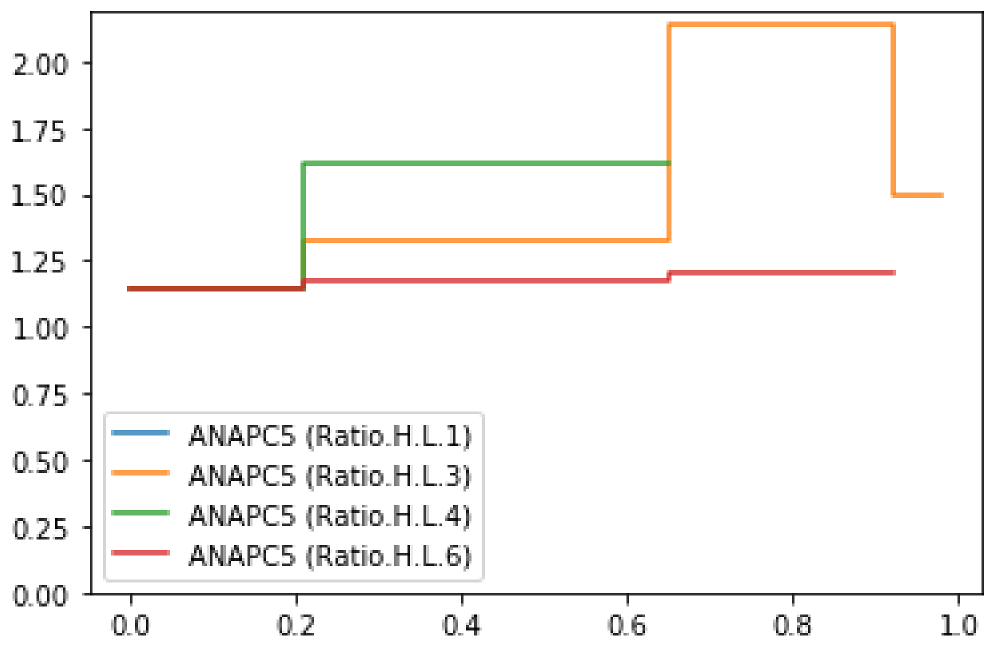

CDC16 (Ratio.H.L.1)
CDC16 (Ratio.H.L.3)
CDC16 (Ratio.H.L.4)
CDC16 (Ratio.H.L.6)
0.0
0.5
1.0
1.5
2.0
0.0
0.2
0.4
0.6
0.8
1.0

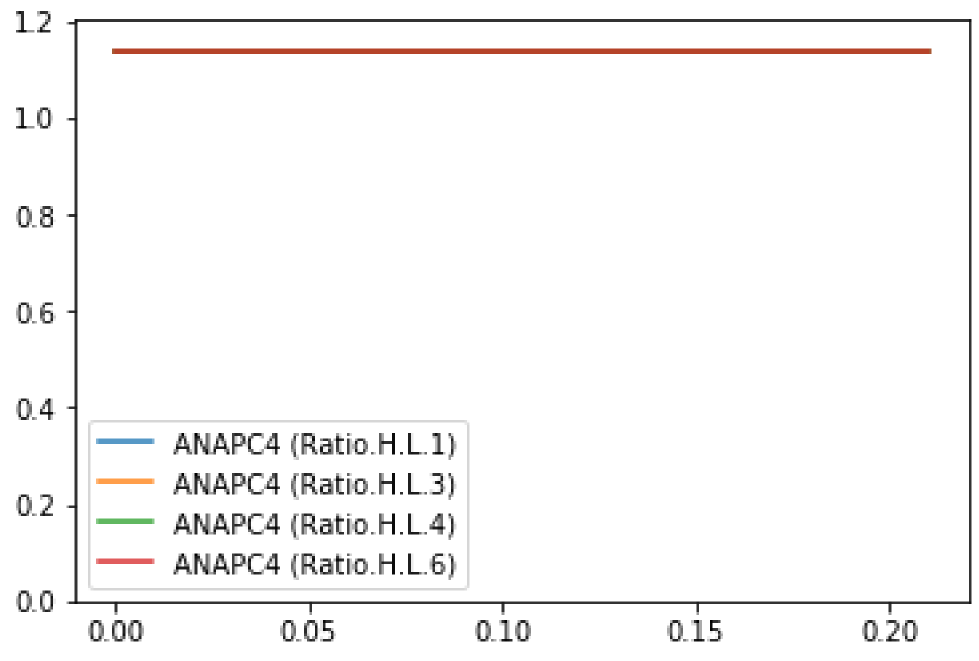

PPP2R2D (Ratio.H.L.1)
PPP2R2D (Ratio.H.L.3)
PPP2R2D (Ratio.H.L.4)
PPP2R2D (Ratio.H.L.6)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0.00
0.05
0.10
0.15
0.20

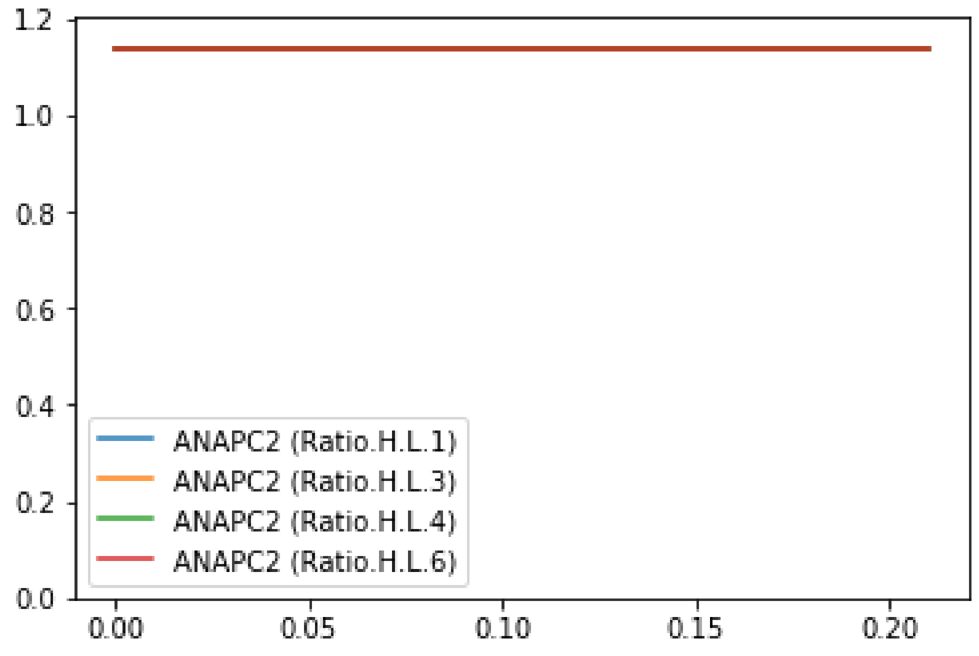

1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
ANAPC1 (Ratio.H.L.1)
ANAPC1 (Ratio.H.L.3)
ANAPC1 (Ratio.H.L.4)
ANAPC1 (Ratio.H.L.6)


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
CDC23 (Ratio.H.L.1)
CDC23 (Ratio.H.L.3)
CDC23 (Ratio.H.L.4)
CDC23 (Ratio.H.L.6)


ANAPC7 (Ratio.H.L.1)
ANAPC7 (Ratio.H.L.3)
ANAPC7 (Ratio.H.L.4)
ANAPC7 (Ratio.H.L.6)
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0


ANAPC5 (Ratio.H.L.1)
ANAPC5 (Ratio.H.L.3)
ANAPC5 (Ratio.H.L.4)
ANAPC5 (Ratio.H.L.6)
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
0.0
0.2
0.4
0.6
0.8
1.0





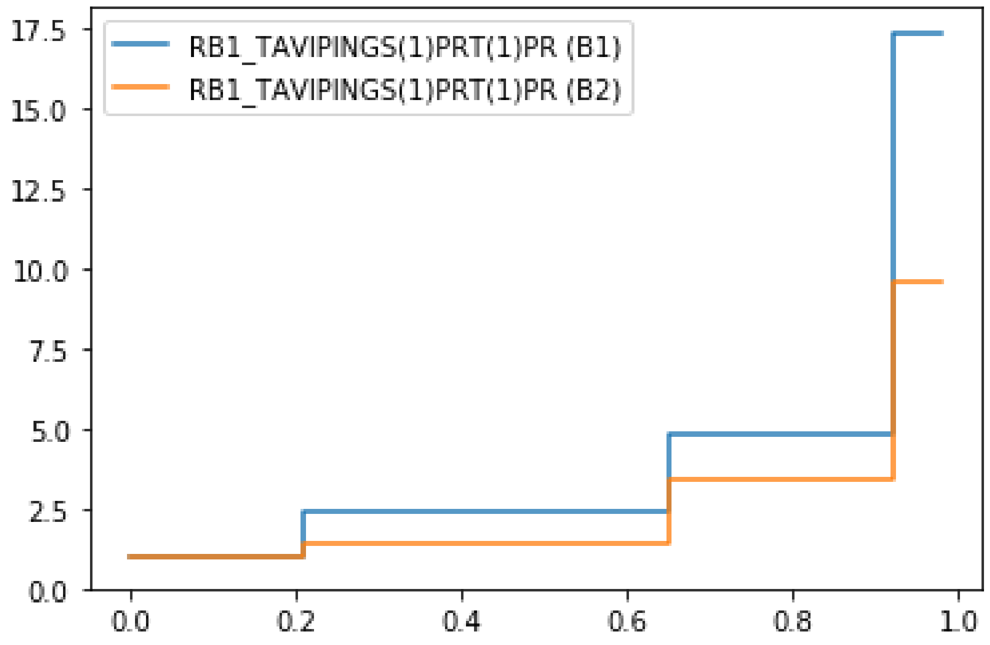
Conclusions: * Surprisingly ENSA drops and CDK1, PPP2R2A and CDC16 go up * No CCNA/E measured

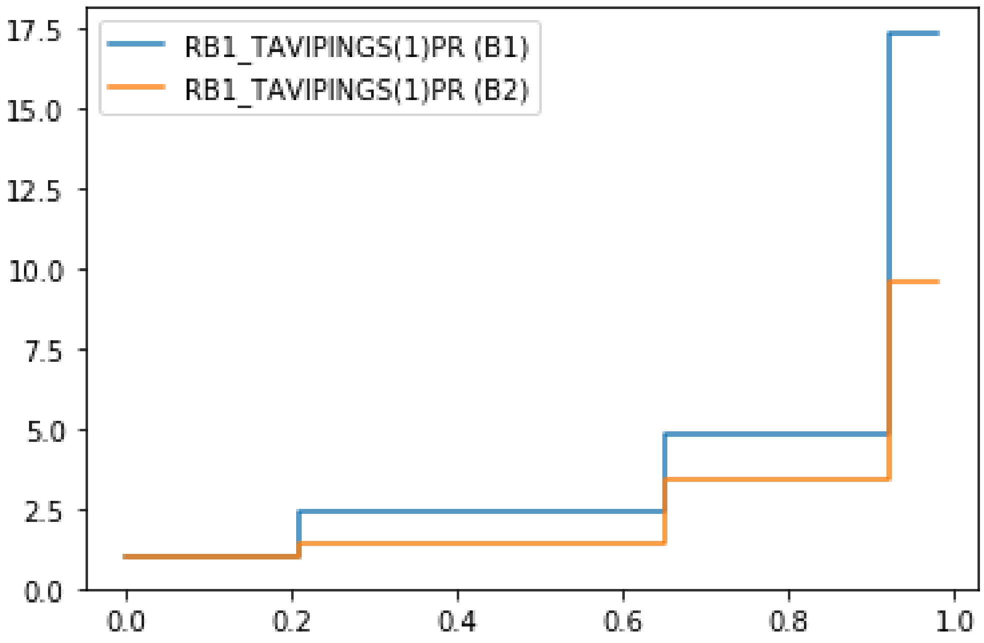
[ ]:
```
##################
# FACS G1-S-G2-M phospho
```

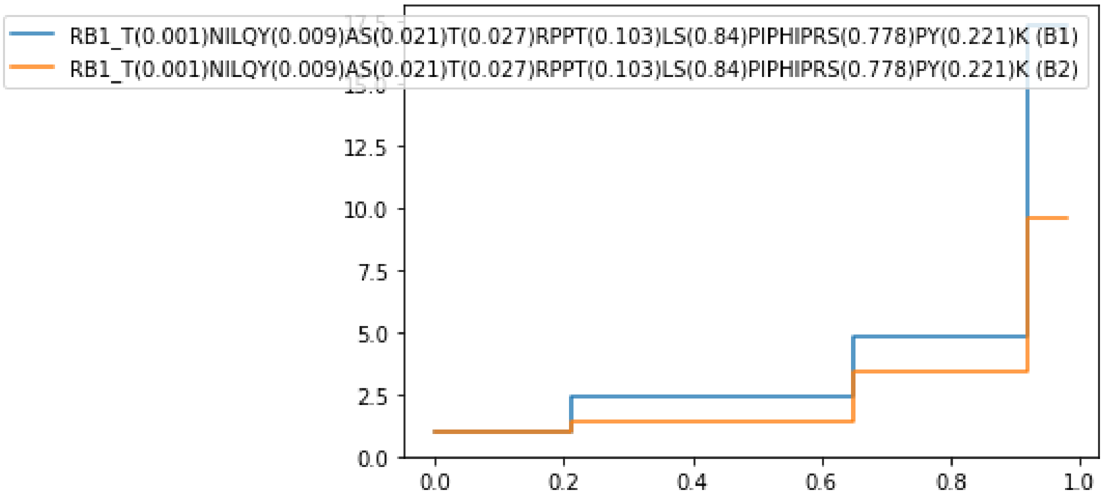
```
##################

import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
from ast import literal_eval

# Enter (part of) gene names to plot
display = []

directory = os.path.abspath('')
if not display:
    data_file = os.path.join(directory, 'merged.xlsx')
    df = pd.read_excel(data_file)
    display = list(set(df.loc[:, 'Gene name']))
    print(df.loc[:, ['Gene name', 'ccModel Paul ID']].drop_duplicates().
 ↪dropna())
    display = [val for val in display if isinstance(val, str)]

# Fetch data of interest and prepare for plotting
data_file = os.path.join(directory, 'elife-27574-supp3-v1.xlsx')
df = pd.read_excel(data_file)

genes_of_interest = []
sites_of_interest = []
for string in display:
    for gene, phospho_site in zip(list(df.loc[:, 'gene_ids']), list(df.loc[:,␣
 ↪'Phospho..STY..Probabilities'])):
        if string == str(gene):
            genes_of_interest.append(gene)
            sites_of_interest.append(phospho_site)

genes_and_sites = zip(genes_of_interest, sites_of_interest)

rows_of_interest = [True if item in genes_of_interest else False for item in␣
 ↪list(df['gene_ids'])]
df_of_interest = df.loc[rows_of_interest]

time = [0, 0.21, 0.21, 0.65, 0.65, 0.92, 0.92, 0.98]

data_to_plot = {}
replicates = ['B1', 'B2']
for gene, site in genes_and_sites:
    row = df_of_interest.loc[df_of_interest['gene_ids'] == gene]
    dict_of_replicates = {}
    for item in replicates:
        abundances = []
        for cc_phase in ['G1', 'S', 'G2', 'M']:
            column_name = '{}.{}'.format(item, cc_phase)
            abundances.append(row.iloc[0][column_name])
        doubled = []
        for val in abundances:
            doubled.extend([val, val])
        dict_of_replicates[item] = doubled
    data_to_plot[gene+'_'+site] = dict_of_replicates

# Plot
for item in data_to_plot:
    plt.figure()
    for replicate in replicates:
        plt.plot(time, data_to_plot[item][replicate])
    plt.gca().set_ylim(bottom=0)
    plt.gca().legend((item+' ('+replicates[0]+')', item+' ('+replicates[1]+')'))
```

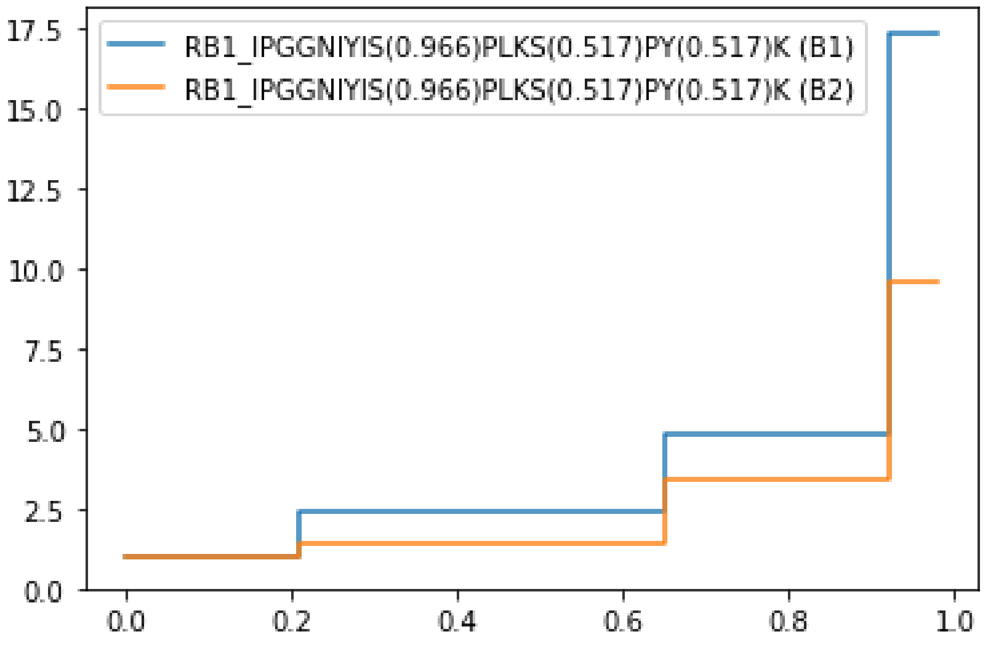
```
    Gene name ccModel Paul ID
0       CDC27             Apc
20       E2F2             E2f
22     ANAPC4             Apc
33    PPP2R2C             B55
42     ANAPC5             Apc
66      CDC23             Apc
75     CDC25B           Cdc25
84       E2F1             E2f
85      CCNE1              Ce
92       FZR1             Cdh
100     CCND1              Cd
112     FOXM1             Fox
122    CDKN1B             p27
126     FBXO5             Emi
129      E2F3             E2f
132     CCND3              Cd
155     CDC20           Cdc20
159     CCND2              Cd
163     MASTL              Gw
167    CDKN1A             p21
175    ARPP19            Ensa
188     CDC16             Apc
200     CCNA1              Ca
207     CCNB1              Cb
214       RB1              Rb
220      TP53             p53
248   ANAPC11             Apc
271      ENSA            Ensa
```

```
286     CCNA2            Ca
288     CCNB3            Cb
296    ANAPC1           Apc
306   PPP2R2B           B55
329     CCNB2            Cb
334    CDC25C         Cdc25
345    CDC25A         Cdc25
350   ANAPC10           Apc
367      WEE1           Wee
375      CDK1           pCb
384     CCNE2            Ce
392   PPP2R2D           B55
400    ANAPC2           Apc
408     CDC26           Apc
412    ANAPC7           Apc
424   PPP2R2A           B55
```

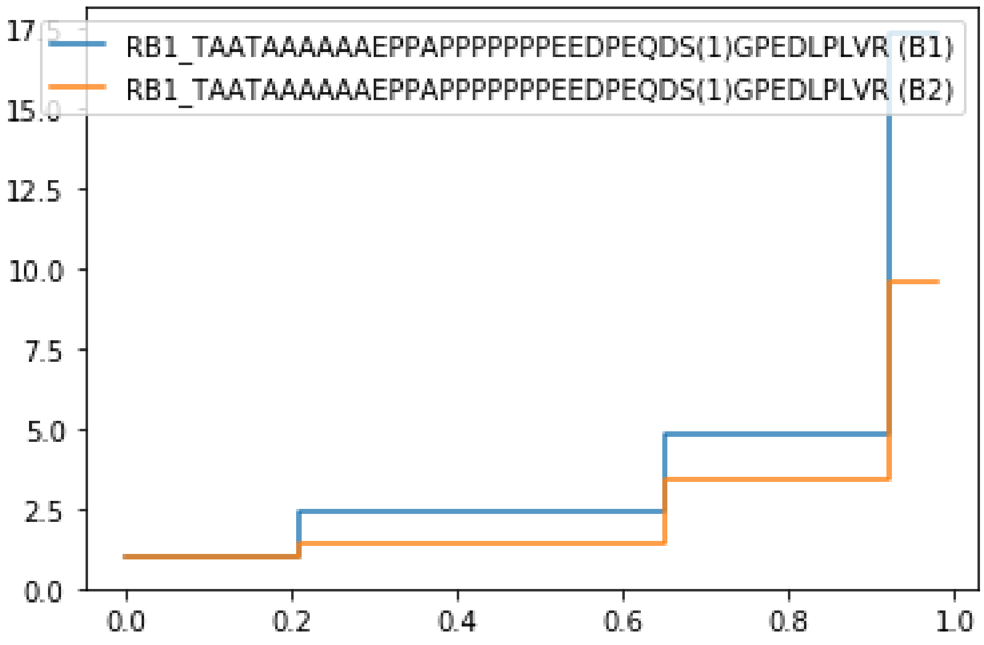



RB1_TAVIPINGS(1)PR (B1)
RB1_TAVIPINGS(1)PR (B2)

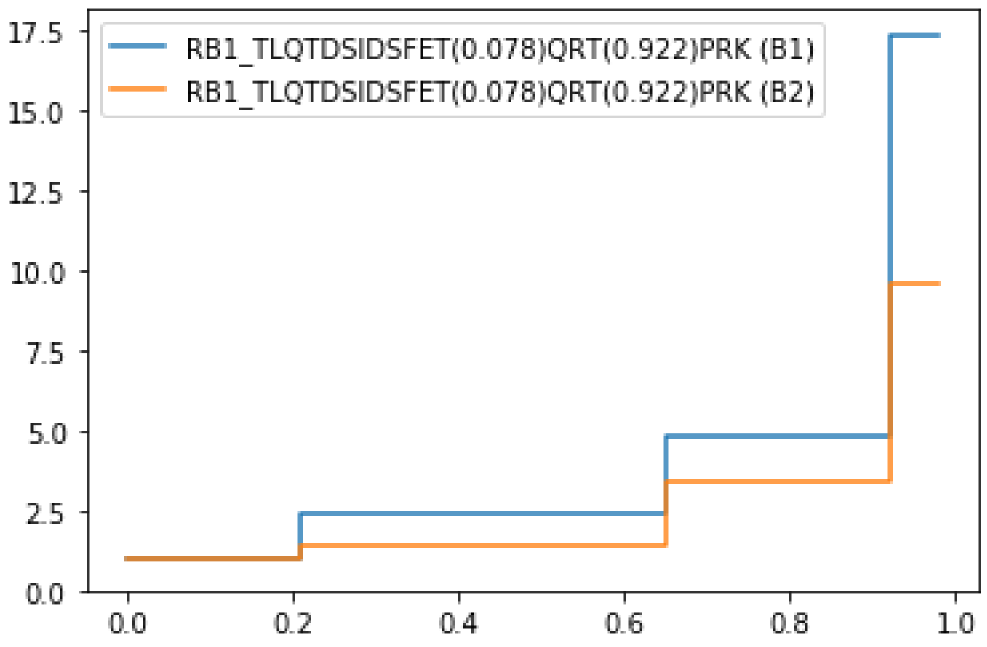

RB1_T(0.001)NILQY(0.009)AS(0.021)T(0.027)RPPT(0.103)LS(0.84)PIPHIPRS(0.778)PY(0.221)K (B1)
RB1_T(0.001)NILQY(0.009)AS(0.021)T(0.027)RPPT(0.103)LS(0.84)PIPHIPRS(0.778)PY(0.221)K (B2)

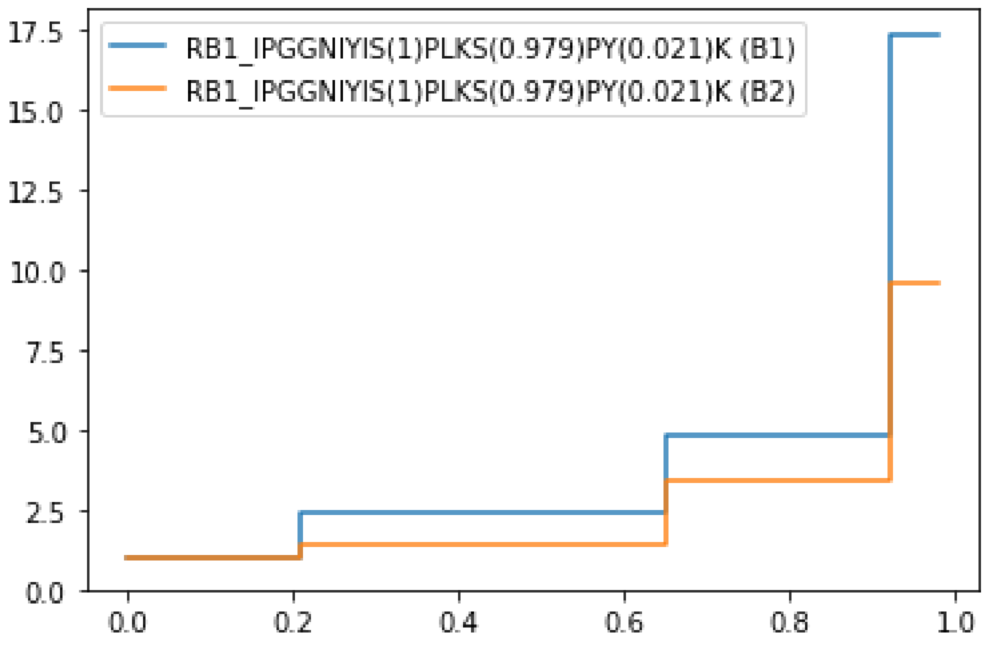


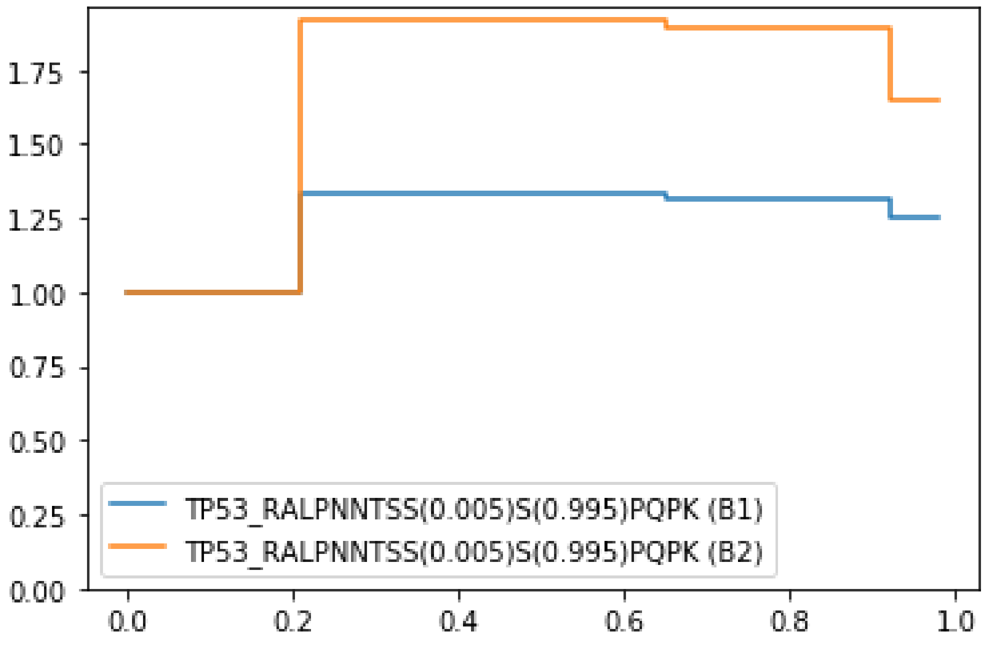
RB1_IPGGNIYIS(0.966)PLKS(0.517)PY(0.517)K (B1)
RB1_IPGGNIYIS(0.966)PLKS(0.517)PY(0.517)K (B2)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.0
0.2
0.4
0.6
0.8
1.0



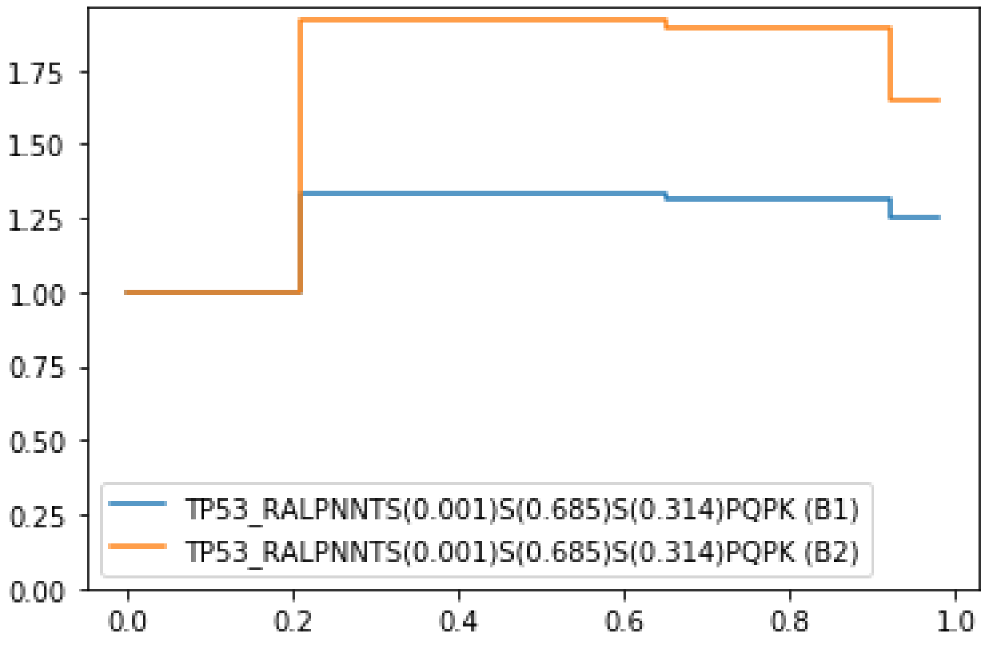
RB1_TAATAAAAAAEPPAPPPPPPPEEDPEQDS(1)GPEDLPLVR (B1)
RB1_TAATAAAAAAEPPAPPPPPPPEEDPEQDS(1)GPEDLPLVR (B2)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.0
0.2
0.4
0.6
0.8
1.0


RB1_TLQTDSIDSFET(0.078)QRT(0.922)PRK (B1)
RB1_TLQTDSIDSFET(0.078)QRT(0.922)PRK (B2)
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

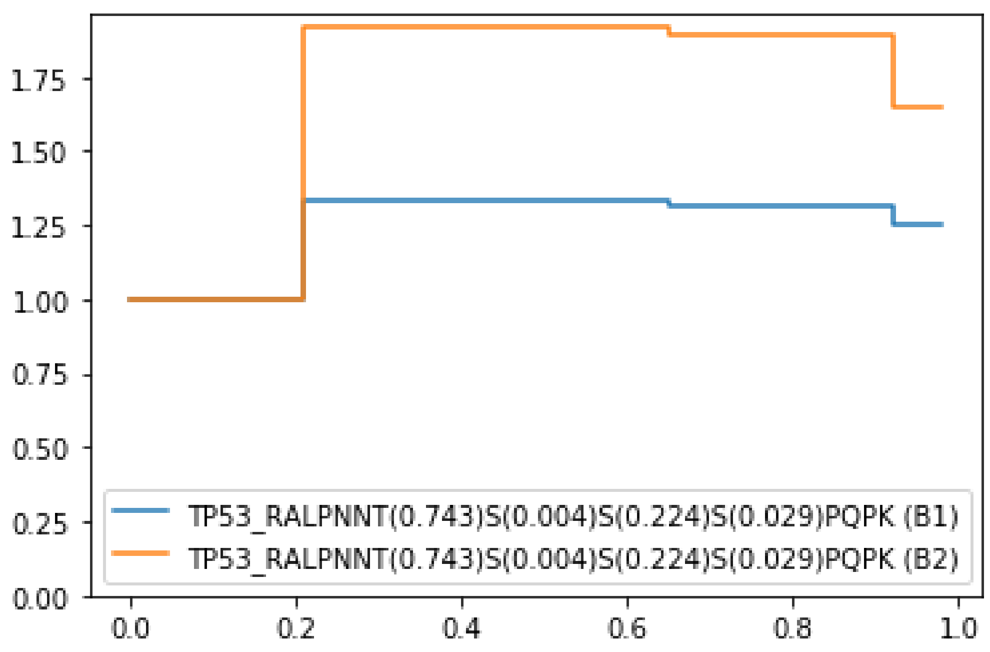

RB1_IPGGNIYIS(1)PLKS(0.979)PY(0.021)K (B1)
RB1_IPGGNIYIS(1)PLKS(0.979)PY(0.021)K (B2)
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

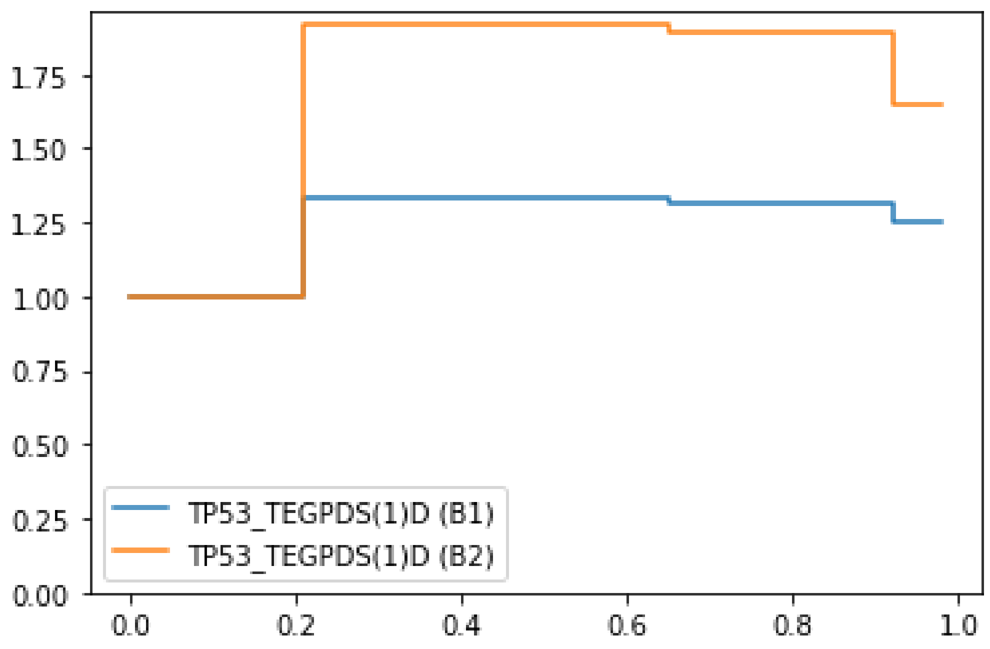


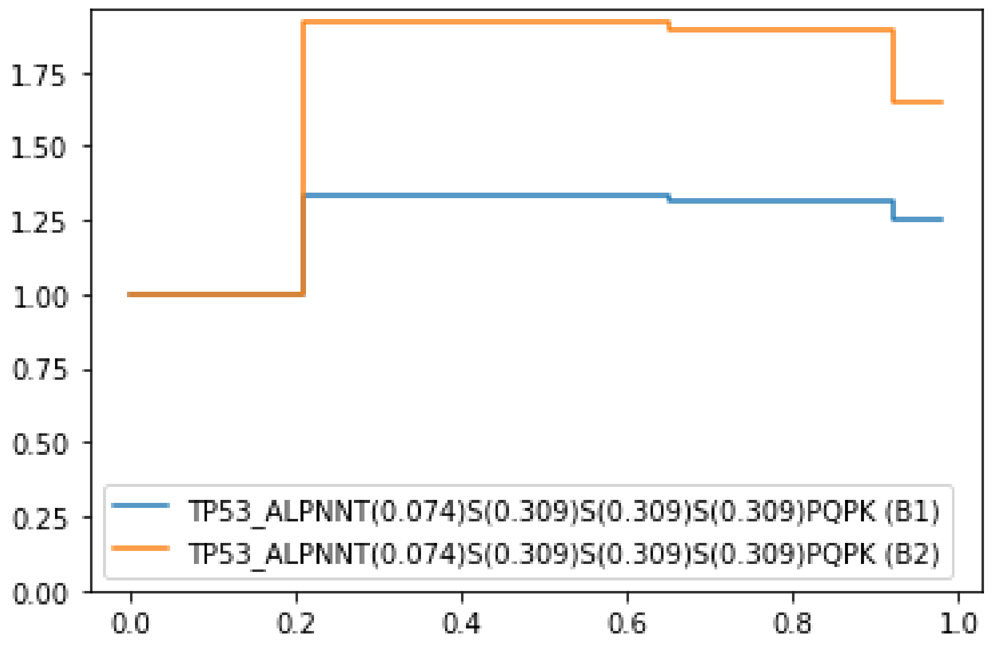
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0
TP53_RALPNNTSS(0.005)S(0.995)PQPK (B1)
TP53_RALPNNTSS(0.005)S(0.995)PQPK (B2)



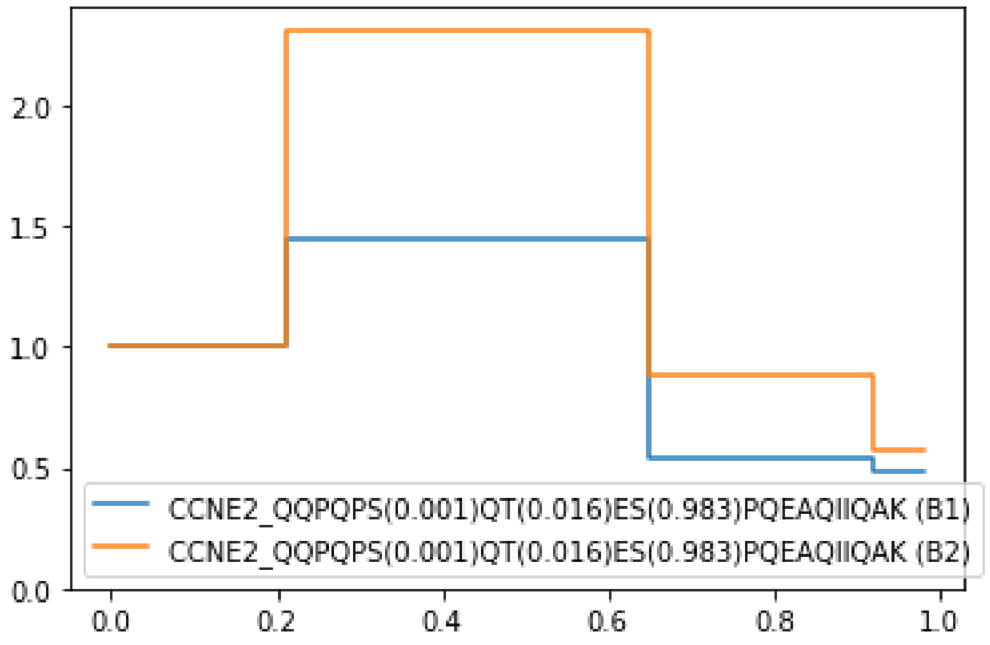
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0
TP53_RALPNNTS(0.001)S(0.685)S(0.314)PQPK (B1)
TP53_RALPNNTS(0.001)S(0.685)S(0.314)PQPK (B2)


1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0
TP53_RALPNNT(0.743)S(0.004)S(0.224)S(0.029)PQPK (B1)
TP53_RALPNNT(0.743)S(0.004)S(0.224)S(0.029)PQPK (B2)

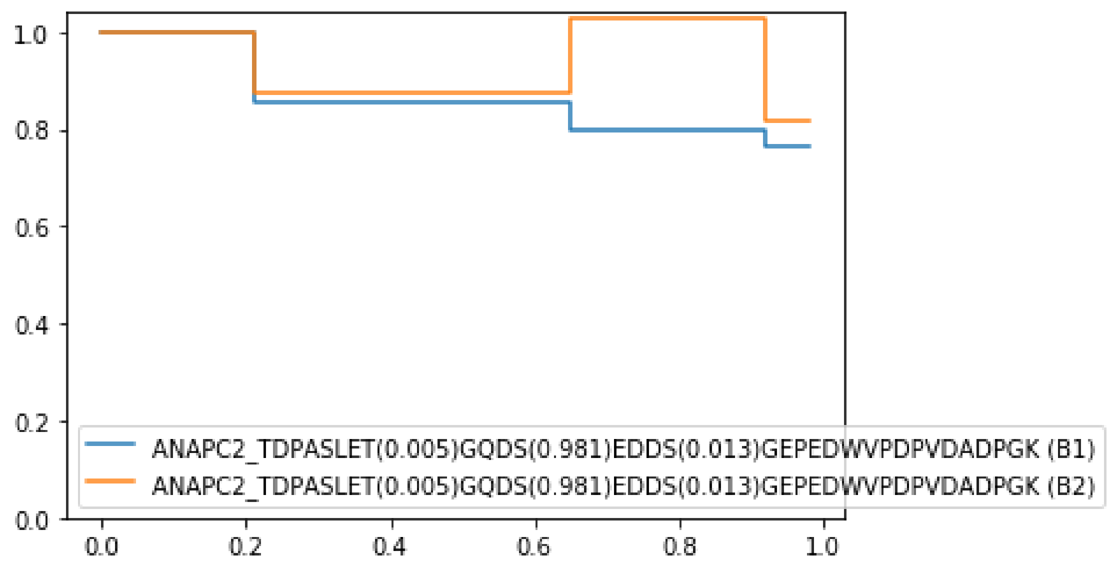

1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0
TP53_TEGPDS(1)D (B1)
TP53_TEGPDS(1)D (B2)



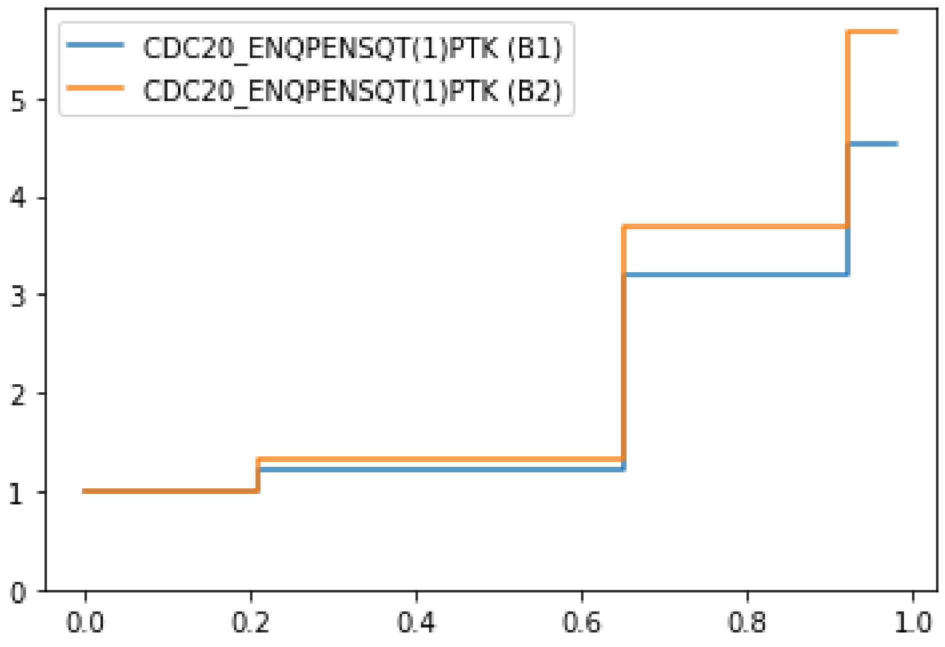
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0
TP53_ALPNNT(0.074)S(0.309)S(0.309)S(0.309)PQPK (B1)
TP53_ALPNNT(0.074)S(0.309)S(0.309)S(0.309)PQPK (B2)



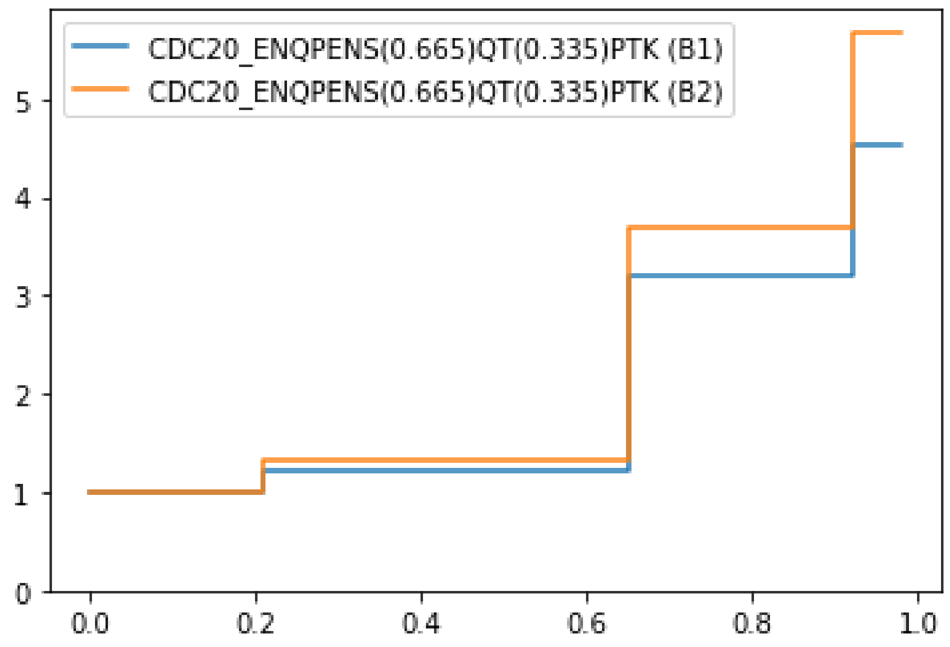
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
CCNE2_QQPQPS(0.001)QT(0.016)ES(0.983)PQEAQIIQAK (B1)
CCNE2_QQPQPS(0.001)QT(0.016)ES(0.983)PQEAQIIQAK (B2)

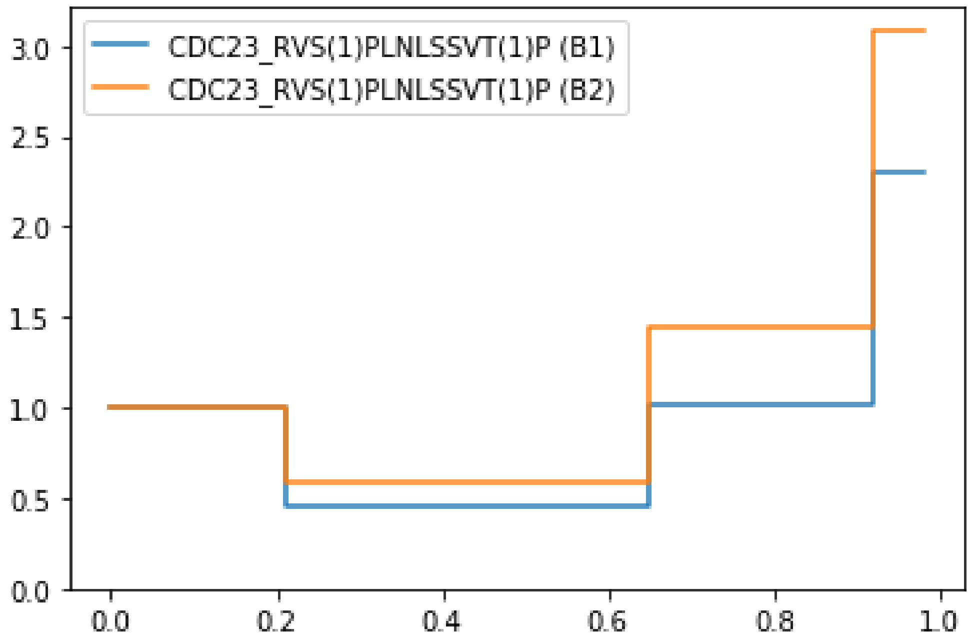


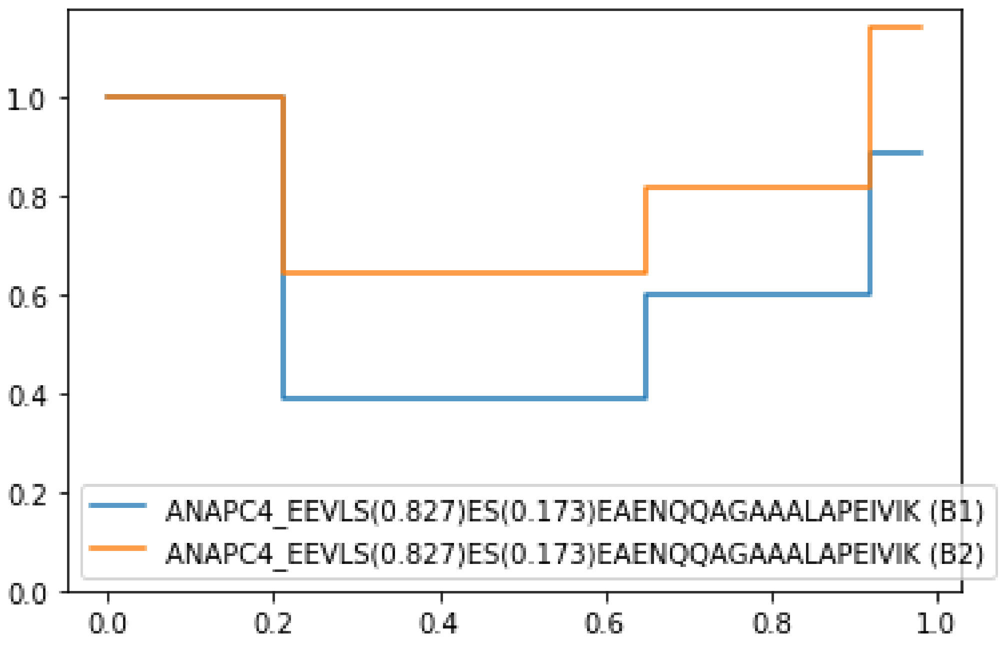
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
ANAPC2_TDPASLET(0.005)GQDS(0.981)EDDS(0.013)GEPEDWVPDPVDADPGK (B1)
ANAPC2_TDPASLET(0.005)GQDS(0.981)EDDS(0.013)GEPEDWVPDPVDADPGK (B2)

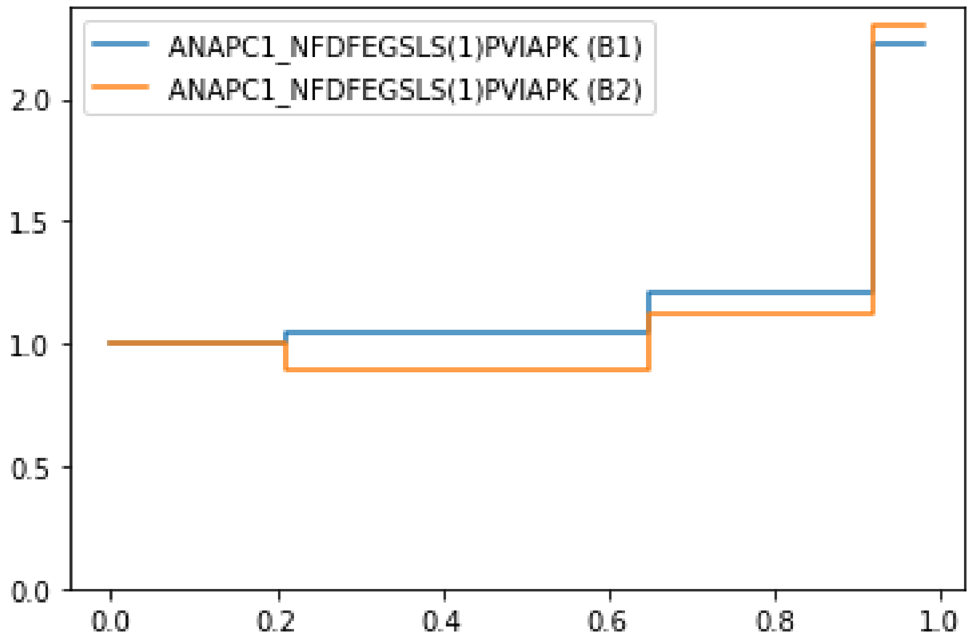


CDC20_ENQPENSQT(1)PTK (B1)
CDC20_ENQPENSQT(1)PTK (B2)
5
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0


CDC20_ENQPENS(0.665)QT(0.335)PTK (B1)
CDC20_ENQPENS(0.665)QT(0.335)PTK (B2)
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0


CDC23_RVS(1)PLNLSSVT(1)P (B1)
CDC23_RVS(1)PLNLSSVT(1)P (B2)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.0
0.2
0.4
0.6
0.8
1.0





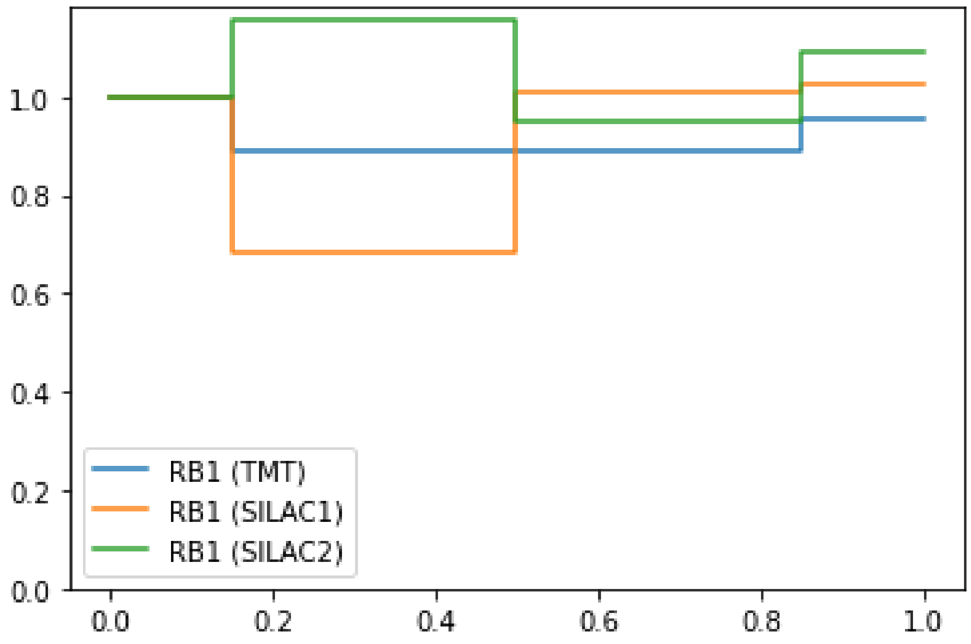
Conclusions: * Interestingly, a CCNE peptide is detected here (highest phosphorylation mid-way through the cell cycle, suggesting that it is not immediately degraded via SCF). * Interestingly, RB1 becomes more and more phosphorylated through the cell cycle

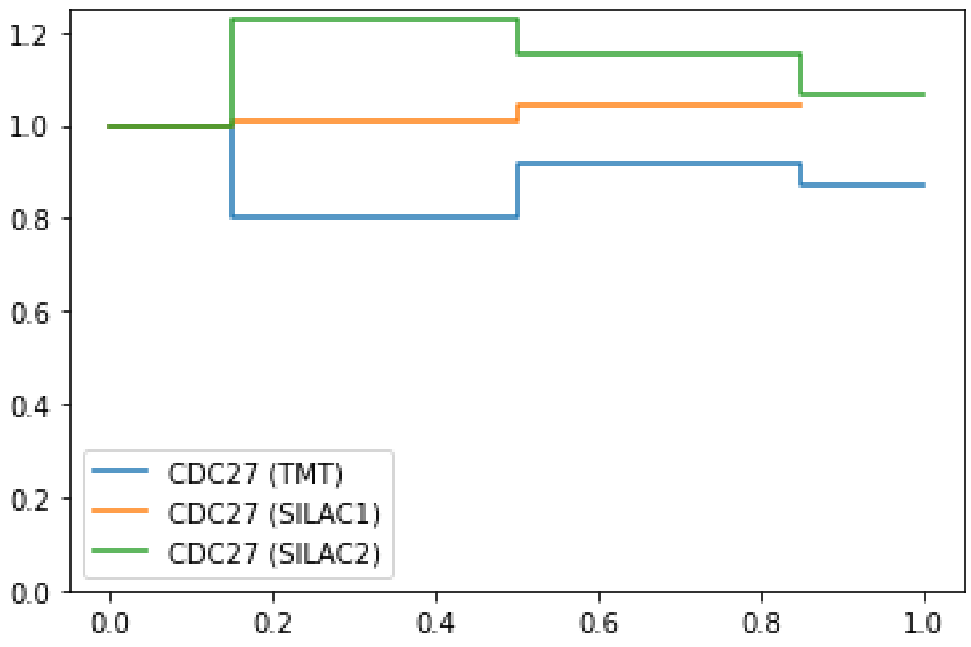
```
[ ]: ##################
     # FACS P-PM1-PM2-A
     ##################

     import pandas as pd
     import os
     import numpy as np
     import matplotlib.pyplot as plt
     from ast import literal_eval

     # Enter (part of) gene names to plot
     display = []

     directory = os.path.abspath('')
     if not display:
         data_file = os.path.join(directory, 'merged.xlsx')
         df = pd.read_excel(data_file)
         display = list(set(df.loc[:, 'Gene name']))
         print(df.loc[:, ['Gene name', 'ccModel Paul ID']].drop_duplicates().
      ↪dropna())
         display = [val for val in display if isinstance(val, str)]

     # Fetch data of interest and prepare for plotting
     data_file = os.path.join(directory, 'elife-27574-supp4-v1.xlsx')
     df = pd.read_excel(data_file)

     genes_of_interest = []
     for string in display:
         for item in list(df.loc[:, 'Gene.names']):
             if string == str(item):
                 genes_of_interest.append(item)

     rows_of_interest = [True if item in genes_of_interest else False for item in␣
      ↪list(df['Gene.names'])]
     df_of_interest = df.loc[rows_of_interest]

     time = [0, 0.15, 0.15, 0.5, 0.5, 0.85, 0.85, 1]

     data_to_plot = {}
     replicates = ['TMT', 'SILAC1', 'SILAC2']
     for gene in genes_of_interest:
         row = df_of_interest.loc[df_of_interest['Gene.names'] == gene]
         dict_of_replicates = {}
         for item in replicates:
             abundances = [1]
             for cc_phase in ['PM1', 'PM2', 'Ana']:
                 column_name = '{}.{}'.format(item, cc_phase)
```

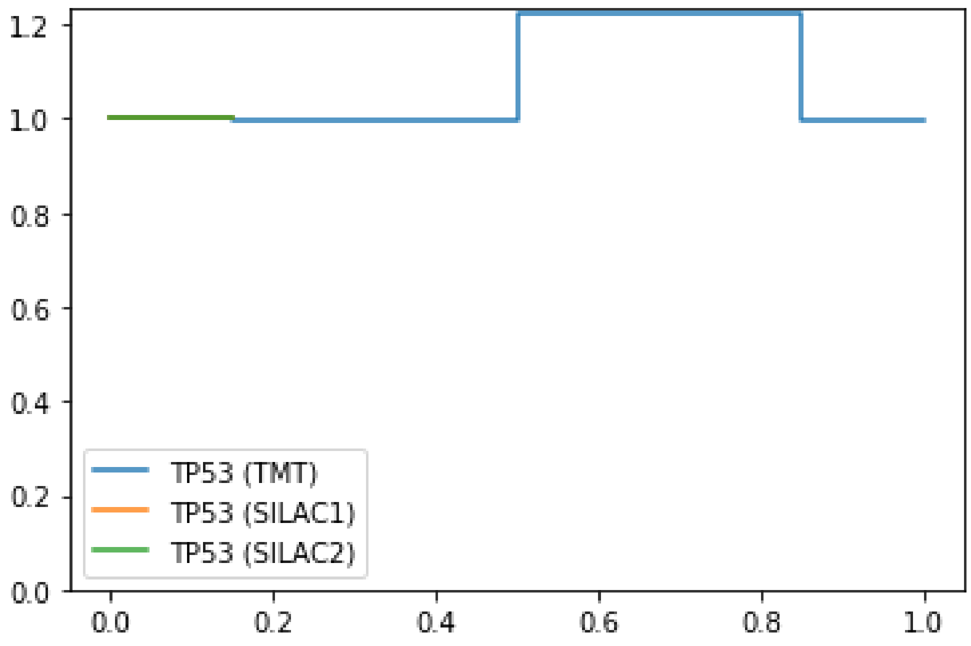
```
            abundances.append(row.iloc[0][column_name])
        doubled = []
        for val in abundances:
            doubled.extend([val, val])
        dict_of_replicates[item] = doubled
    data_to_plot[gene] = dict_of_replicates

# Plot
for gene in data_to_plot:
    plt.figure()
    for replicate in replicates:
        plt.plot(time, data_to_plot[gene][replicate])
    plt.gca().set_ylim(bottom=0)
    plt.gca().legend((gene+' ('+replicates[0]+')', gene+' ('+replicates[1]+')',␣
 ↪gene+' ('+replicates[2]+')'))
```

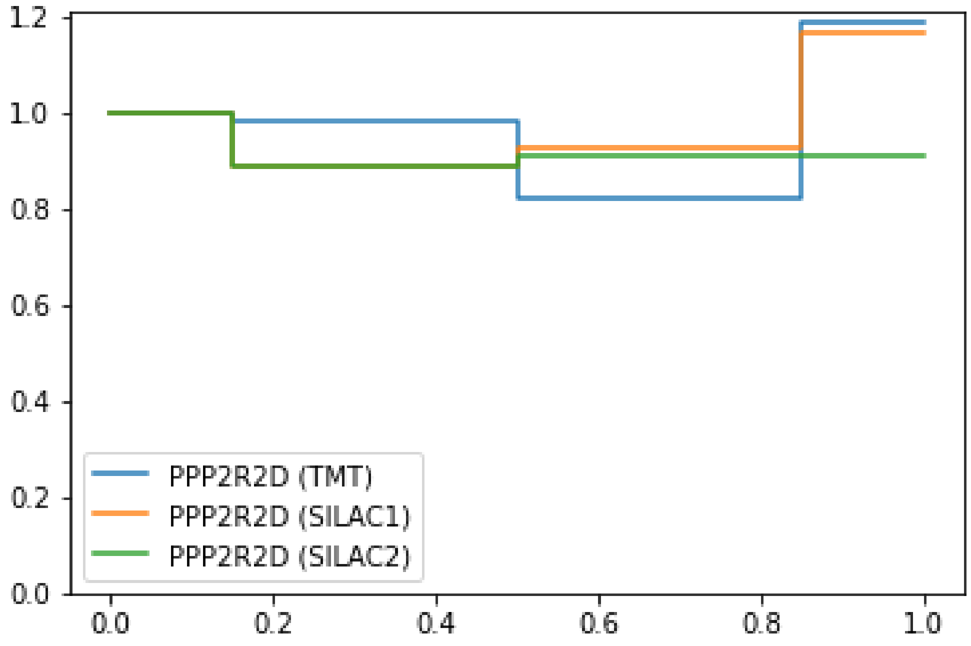
```
    Gene name ccModel Paul ID
0       CDC27             Apc
20       E2F2             E2f
22     ANAPC4             Apc
33    PPP2R2C             B55
42     ANAPC5             Apc
66      CDC23             Apc
75     CDC25B           Cdc25
84       E2F1             E2f
85      CCNE1              Ce
92       FZR1             Cdh
100     CCND1              Cd
112     FOXM1             Fox
122    CDKN1B             p27
126     FBXO5             Emi
129      E2F3             E2f
132     CCND3              Cd
155     CDC20           Cdc20
159     CCND2              Cd
163     MASTL              Gw
167    CDKN1A             p21
175    ARPP19            Ensa
188     CDC16             Apc
200     CCNA1              Ca
207     CCNB1              Cb
214       RB1              Rb
220      TP53             p53
248   ANAPC11             Apc
271      ENSA            Ensa
286     CCNA2              Ca
288     CCNB3              Cb
```

```
296    ANAPC1         Apc
306   PPP2R2B         B55
329     CCNB2          Cb
334    CDC25C       Cdc25
345    CDC25A       Cdc25
350   ANAPC10         Apc
367      WEE1         Wee
375      CDK1         pCb
384     CCNE2          Ce
392   PPP2R2D         B55
400    ANAPC2         Apc
408     CDC26         Apc
412    ANAPC7         Apc
424   PPP2R2A         B55
```

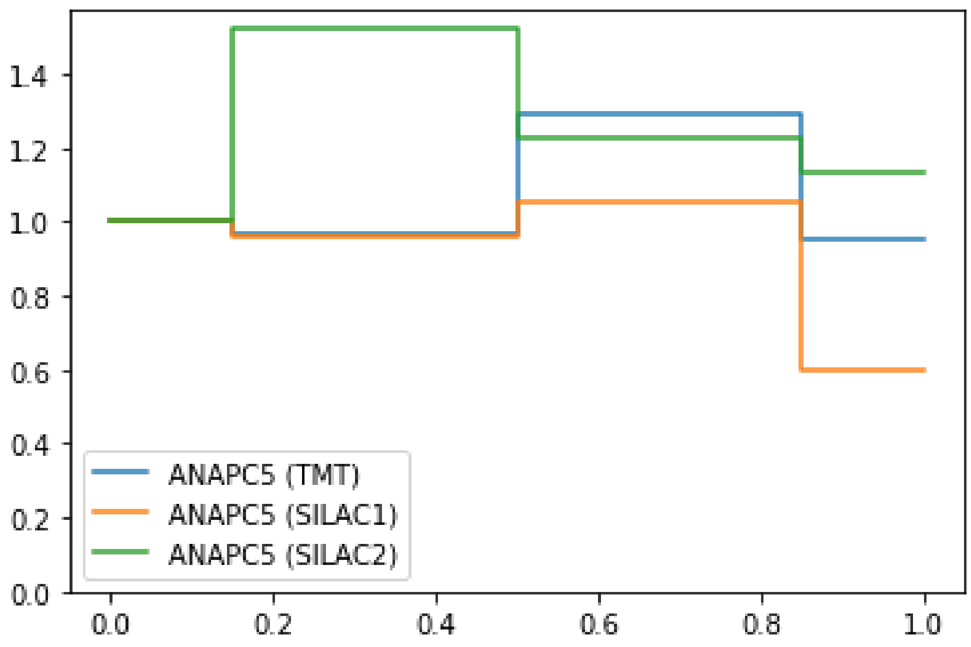



1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
CDC27 (TMT)
CDC27 (SILAC1)
CDC27 (SILAC2)

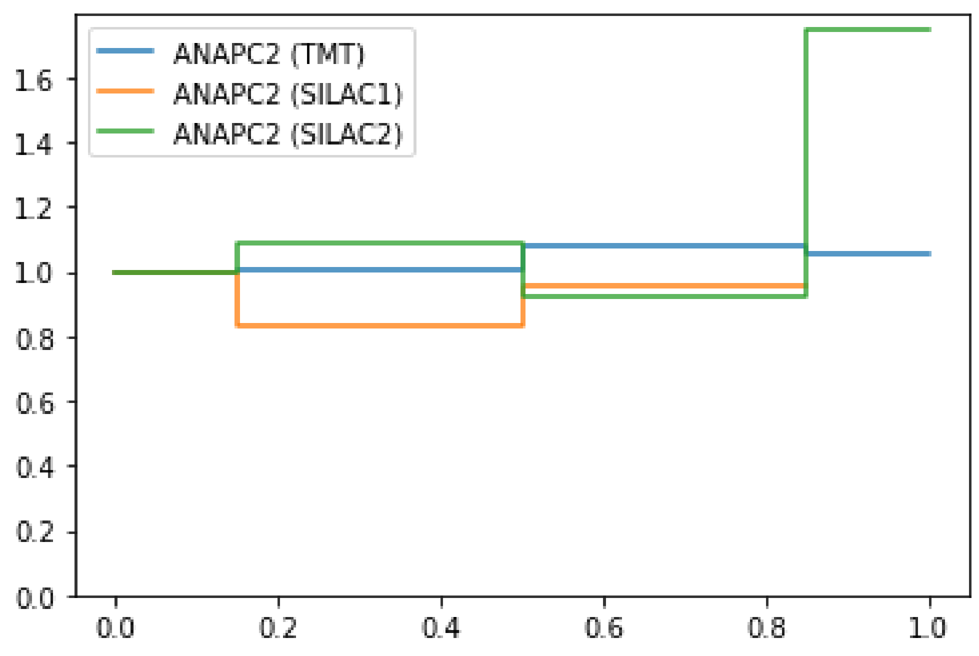

1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
TP53 (TMT)
TP53 (SILAC1)
TP53 (SILAC2)

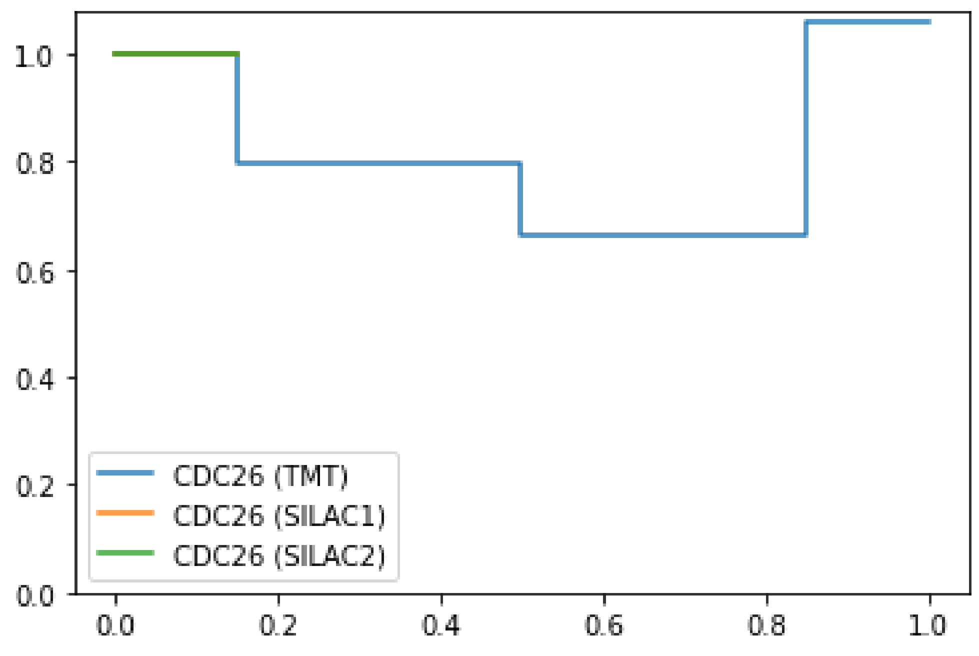


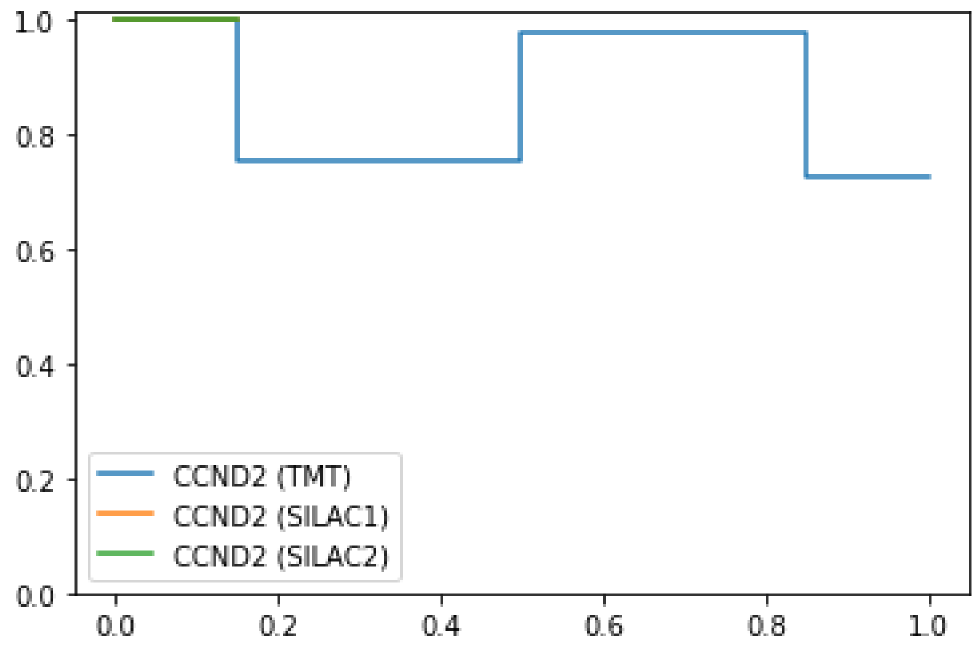
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
PPP2R2D (TMT)
PPP2R2D (SILAC1)
PPP2R2D (SILAC2)



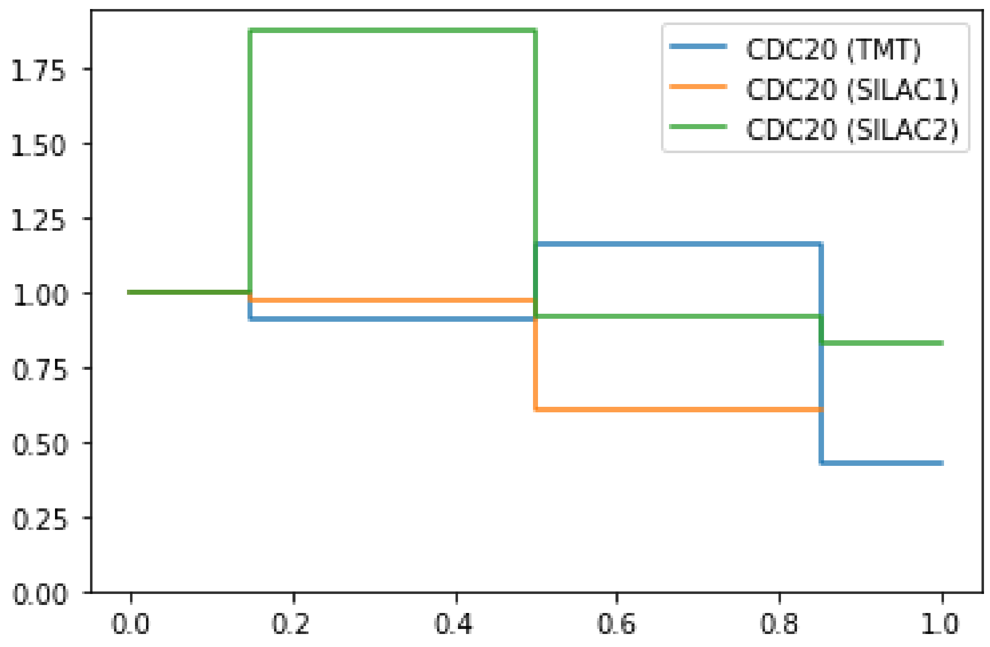
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
ANAPC5 (TMT)
ANAPC5 (SILAC1)
ANAPC5 (SILAC2)


ANAPC2 (TMT)
ANAPC2 (SILAC1)
ANAPC2 (SILAC2)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0.0
0.2
0.4
0.6
0.8
1.0

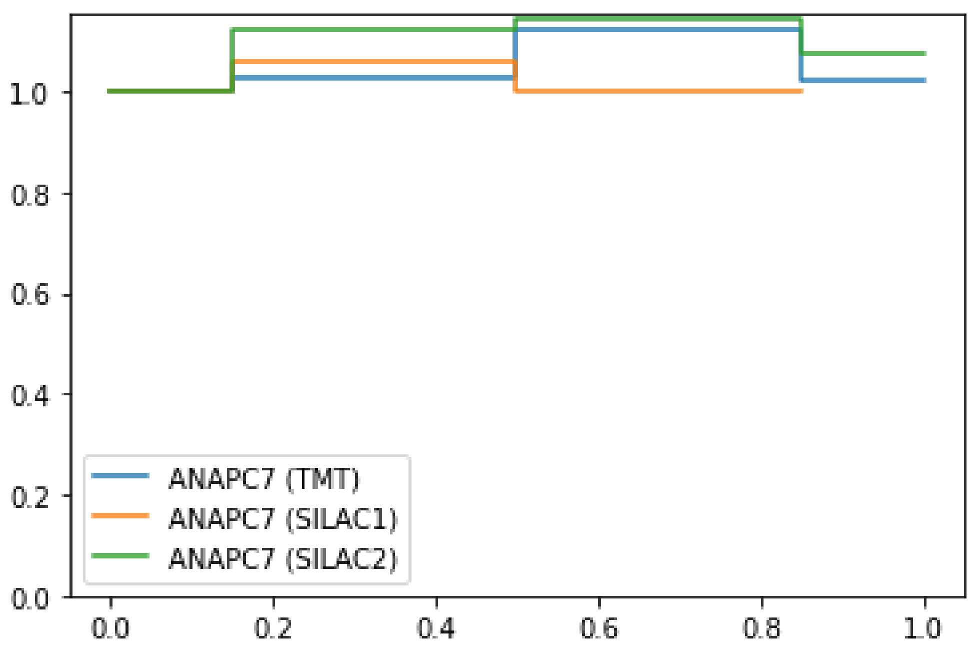

CDC26 (TMT)
CDC26 (SILAC1)
CDC26 (SILAC2)
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

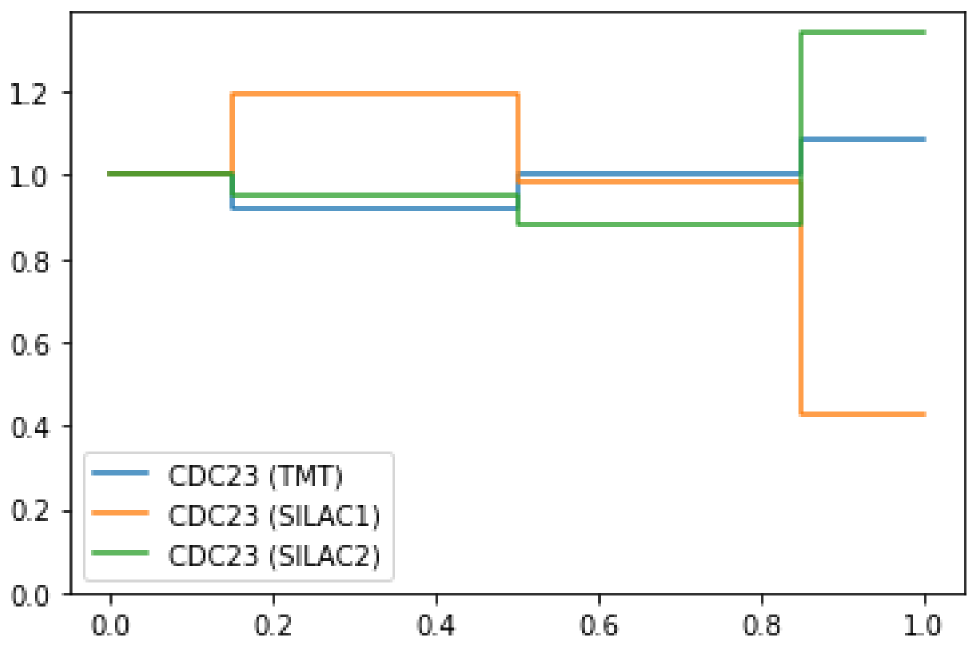


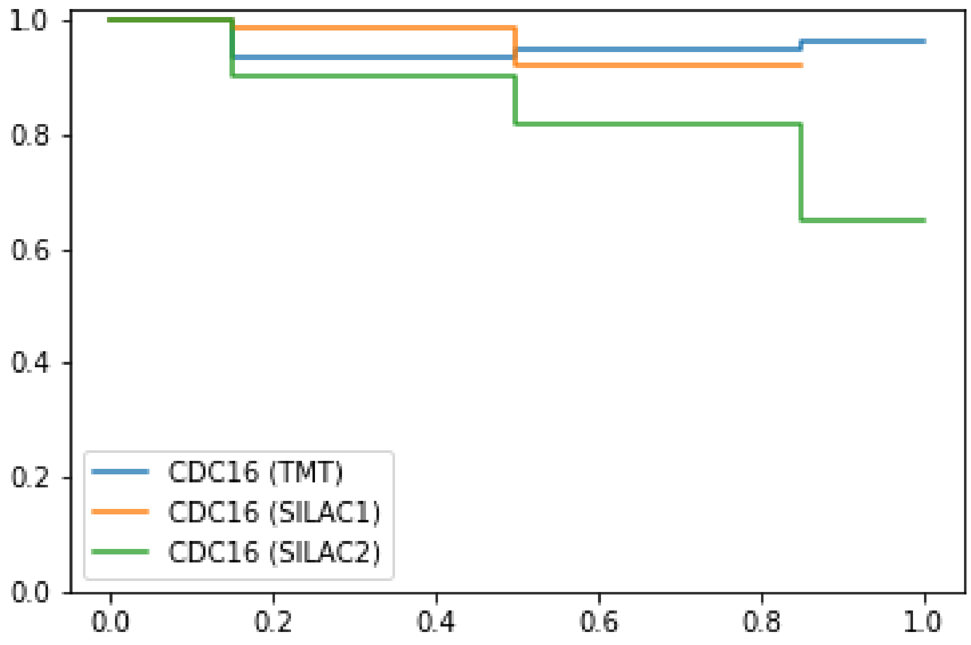
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
CCND2 (TMT)
CCND2 (SILAC1)
CCND2 (SILAC2)

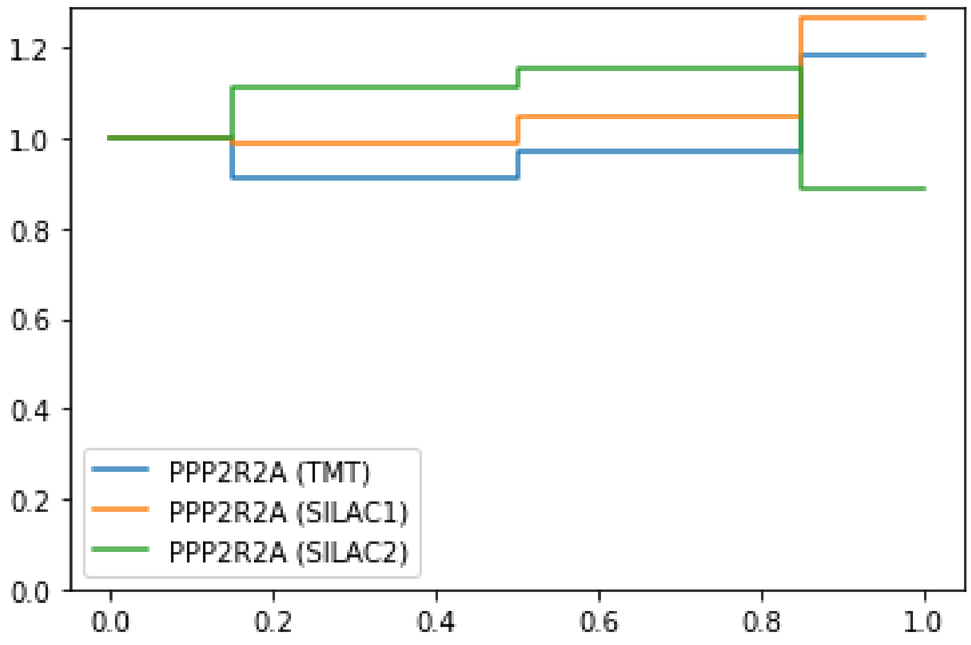


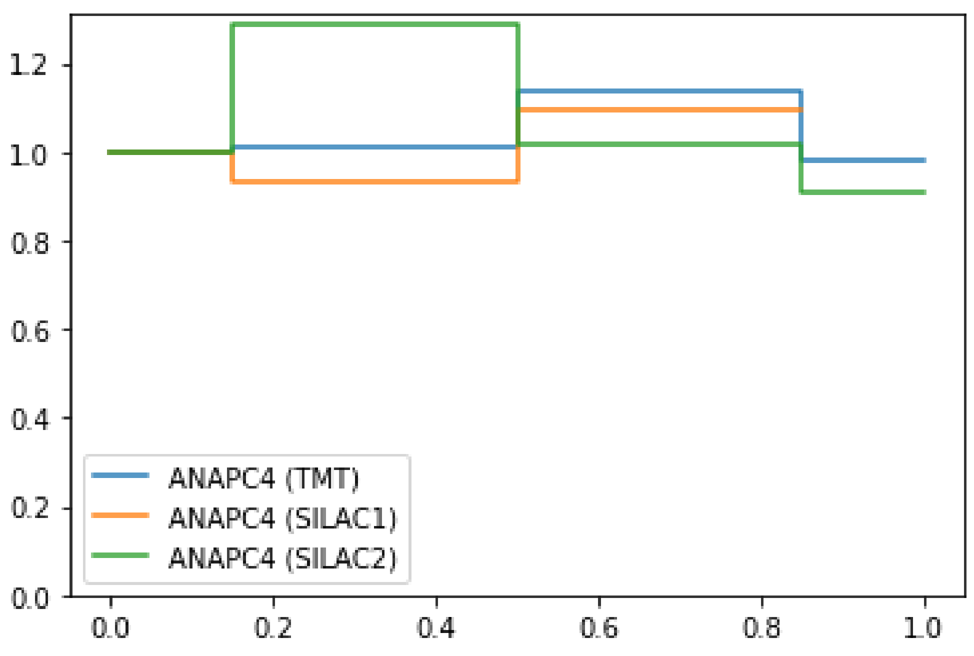
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0
CDC20 (TMT)
CDC20 (SILAC1)
CDC20 (SILAC2)


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
ANAPC7 (TMT)
ANAPC7 (SILAC1)
ANAPC7 (SILAC2)

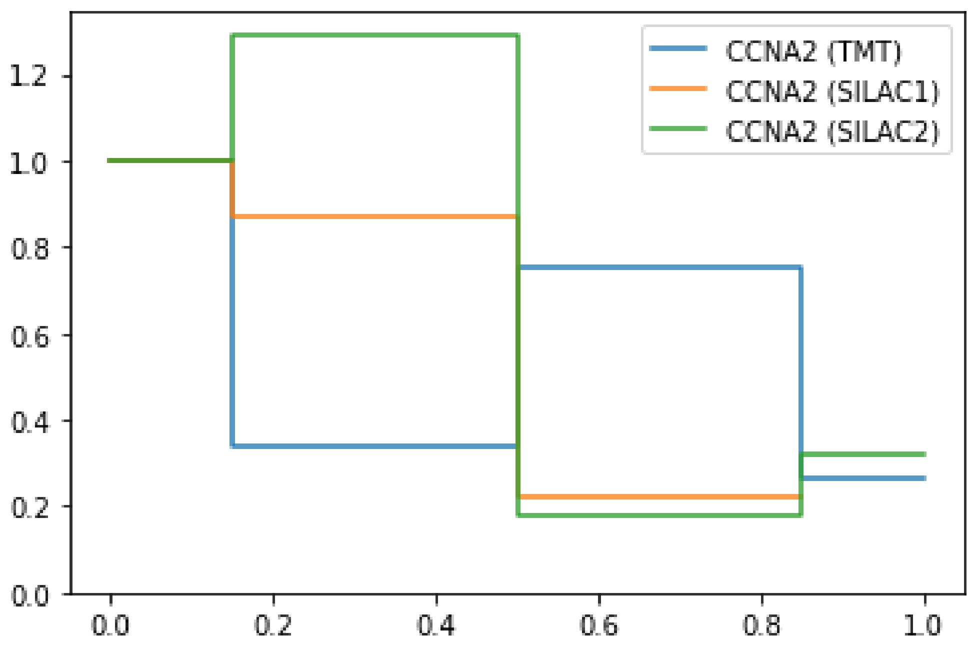

1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
CDC23 (TMT)
CDC23 (SILAC1)
CDC23 (SILAC2)


CDC16 (TMT)
CDC16 (SILAC1)
CDC16 (SILAC2)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

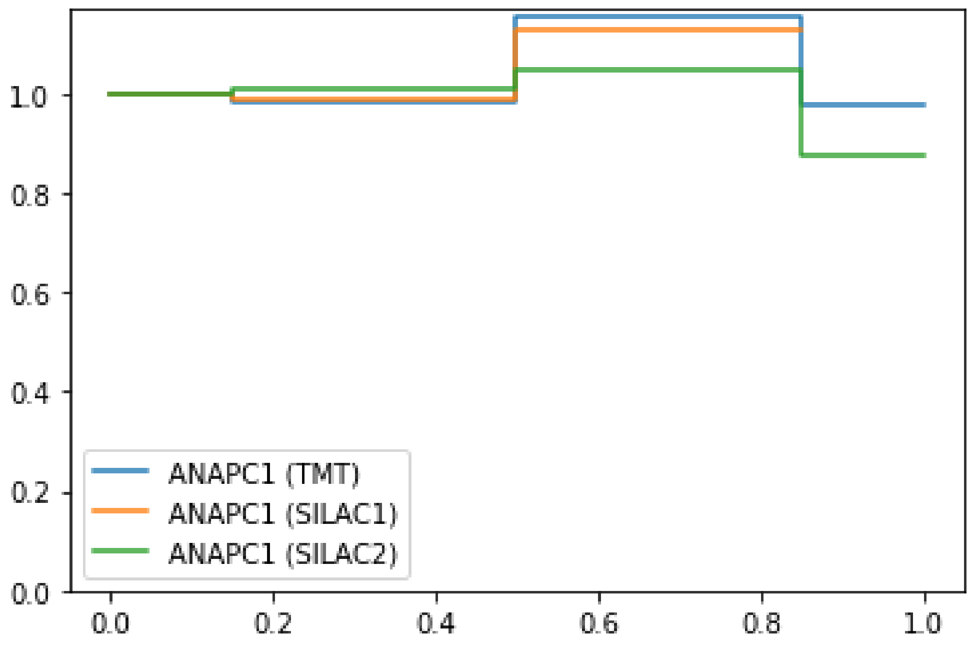

PPP2R2A (TMT)
PPP2R2A (SILAC1)
PPP2R2A (SILAC2)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0



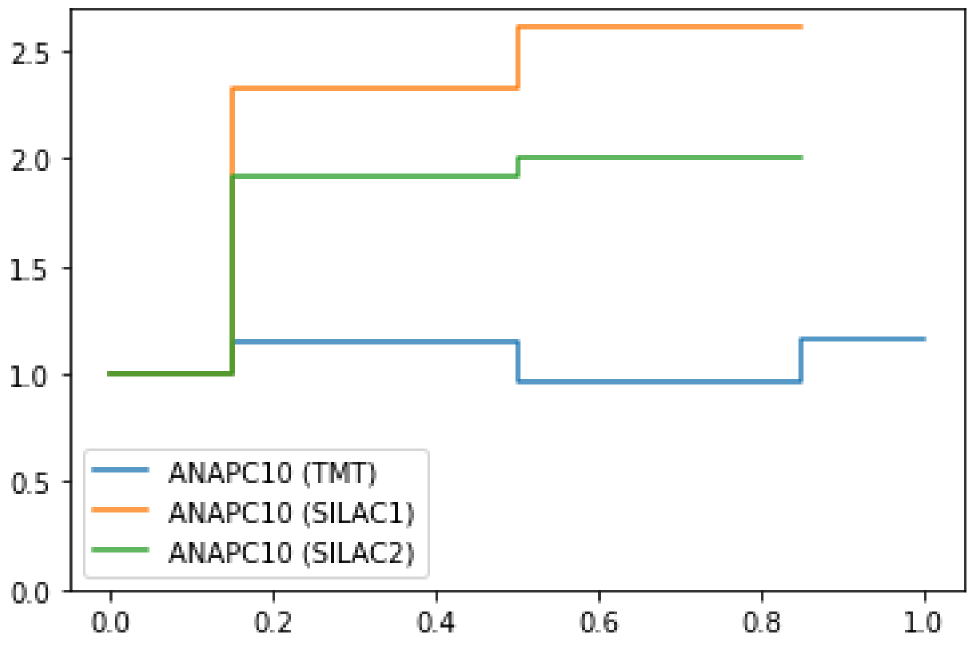
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
ANAPC4 (TMT)
ANAPC4 (SILAC1)
ANAPC4 (SILAC2)



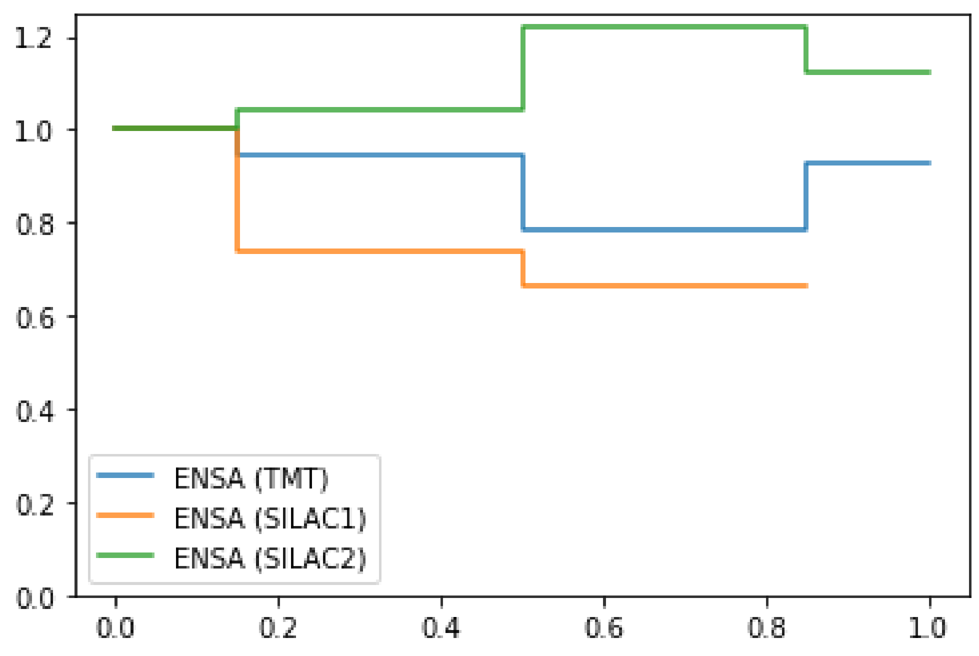
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
CCNA2 (TMT)
CCNA2 (SILAC1)
CCNA2 (SILAC2)

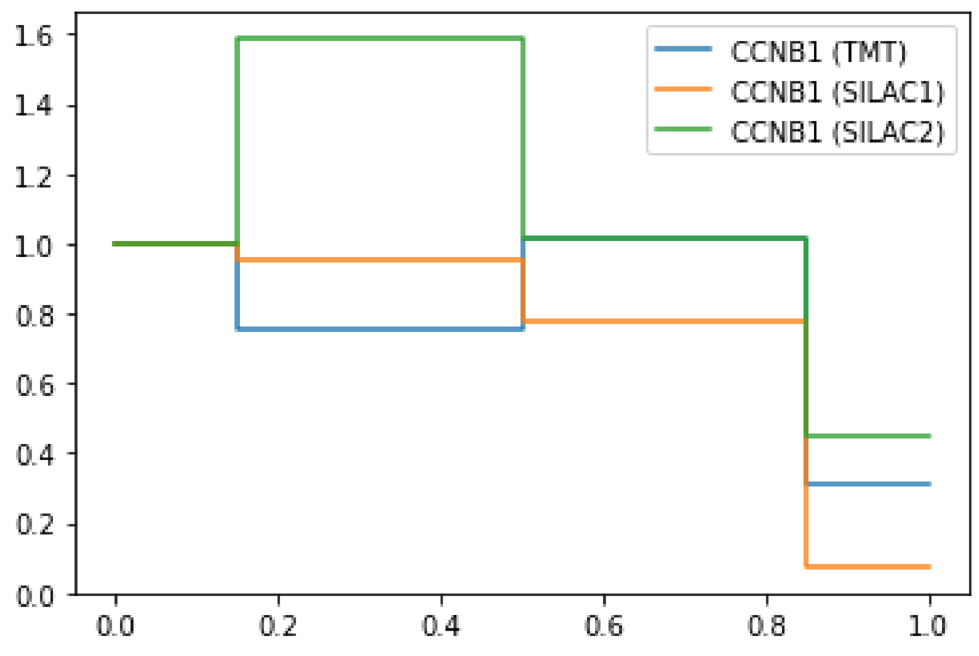


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
ANAPC1 (TMT)
ANAPC1 (SILAC1)
ANAPC1 (SILAC2)



2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
ANAPC10 (TMT)
ANAPC10 (SILAC1)
ANAPC10 (SILAC2)





Conclusions: * limited precision * CCNB1 and CCNA2 (and CDC20) become degraded

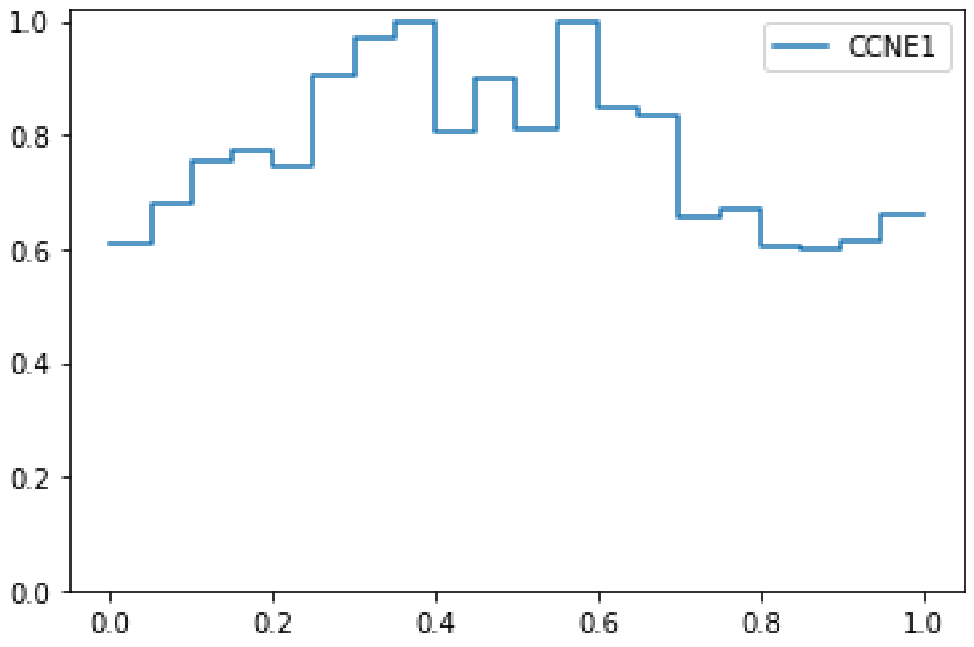
[ ]:
```
##################
# FACS P-PM1-PM2-A phospho
##################
```

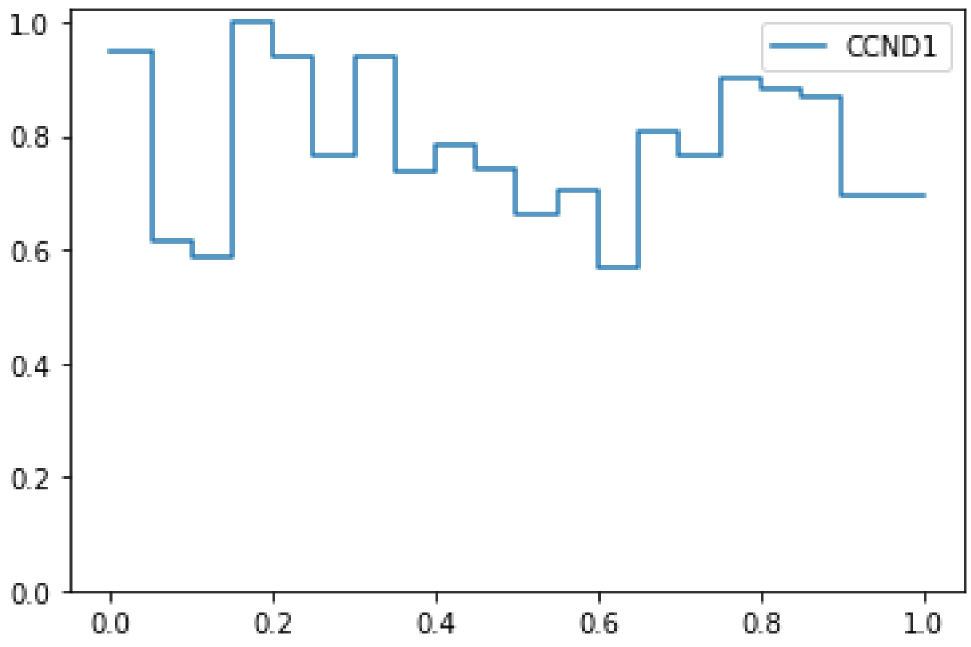
```
# There is no data for that
```

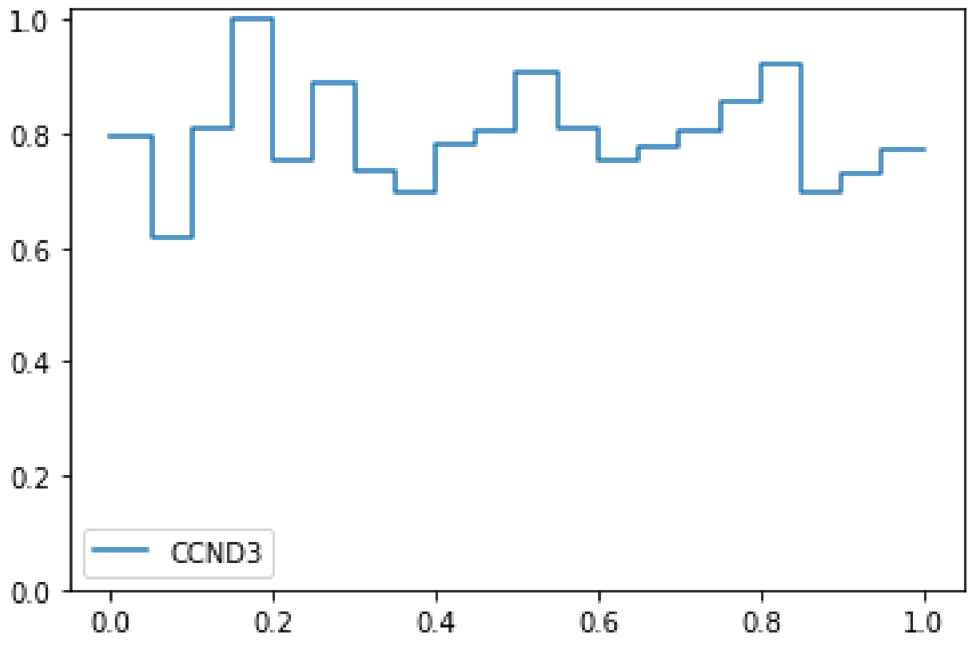
4 Mahdessian et al. 2019 - FUCCI based immunufluorescence trajectory

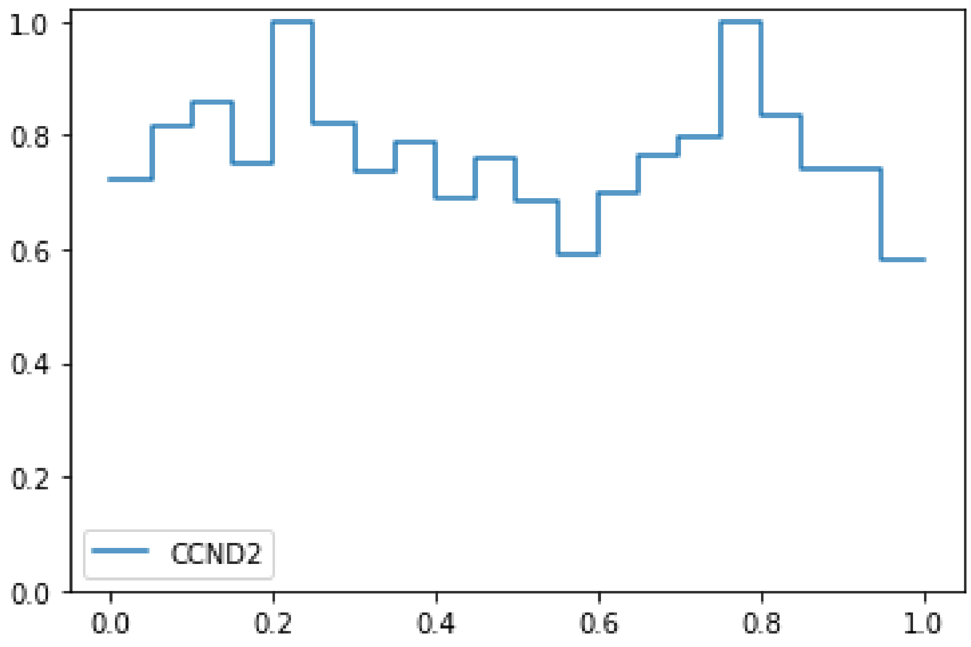
[ ]:
```
##################
# HPA_Fig3A
# Data does not include cells in mitosis
##################

import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
from ast import literal_eval

# Enter (part of) ENSEMBL ID to plot
display = [] # e.g. CCNA2 is 'ENSG00000145386'

directory = os.path.abspath('')
data_file = os.path.join(directory, 'merged.xlsx')
df = pd.read_excel(data_file)
translation_table = df.loc[:, ['ENSEMBL ID', 'Gene name']].drop_duplicates()
if not display:
    # print(translation_table)
    display = [val for val in translation_table.loc[:, 'ENSEMBL ID'] if␣
 ↪isinstance(val, str)]

# Fetch data of interest and prepare for plotting
data_file = os.path.join(directory, 'HPA_Fig3A_all.xlsx')
names = ['ENSEMBL ID', 'cell cycle time', 'abundance']
df = pd.read_excel(data_file, header=None, names=names)
df = df.pivot(index='ENSEMBL ID', columns='cell cycle time').reset_index()

genes_of_interest = []
for string in display:
    for item in list(df.loc[:, 'ENSEMBL ID']):
        if string == str(item):
            genes_of_interest.append(item)

rows_of_interest = [True if item in genes_of_interest else False for item in␣
 ↪list(df.loc[:, 'ENSEMBL ID'])]
df_of_interest = df.loc[rows_of_interest]
time = df.columns.get_level_values(1)
time = [val for val in time if isinstance(val, float)]
time = [val for val in time for _ in (0, 1)]
```

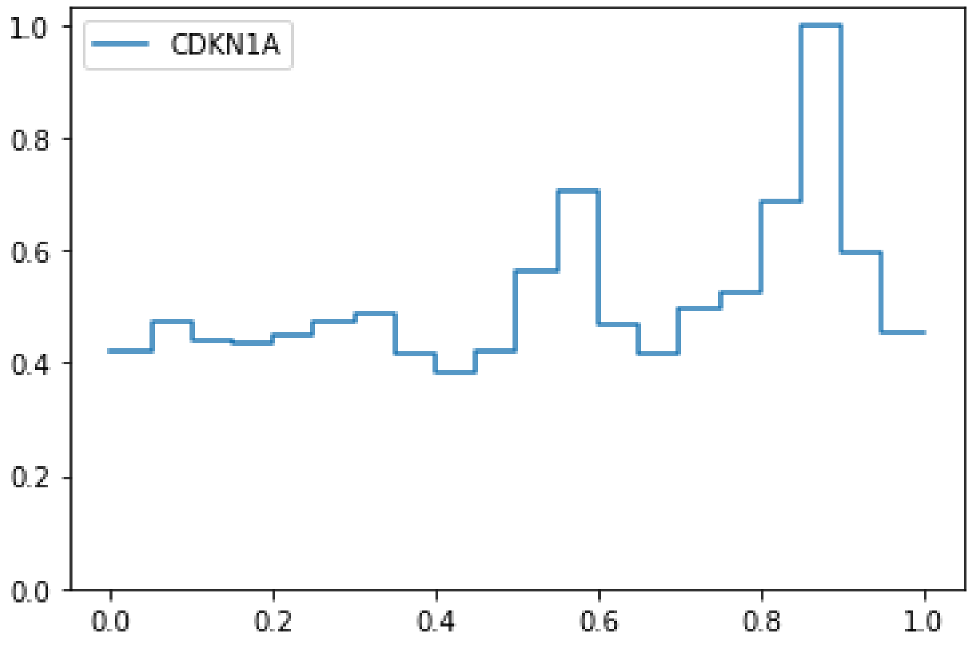
```python
time.insert(0, 0)
del time[-1]

data_to_plot = {}
for gene in genes_of_interest:
    abundances = df_of_interest.loc[df_of_interest['ENSEMBL ID'] == gene]
    abundances = list(abundances.iloc[0, 1:])
    doubled = []
    for val in abundances:
        doubled.extend([val, val])
    data_to_plot[gene] = doubled

# Plot
for gene in data_to_plot:
    plt.figure()
    plt.plot(time, data_to_plot[gene])
    plt.gca().set_ylim(bottom=0)
    label = translation_table.loc[translation_table['ENSEMBL ID'] == gene]
    label = label.iloc[0, 1]
    plt.gca().legend((label,))
```

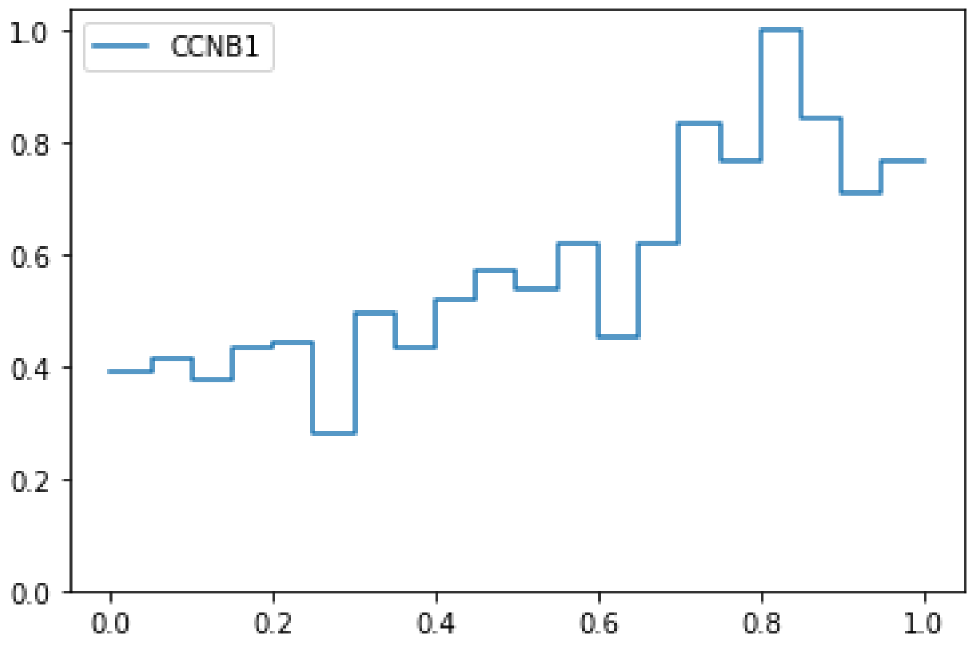


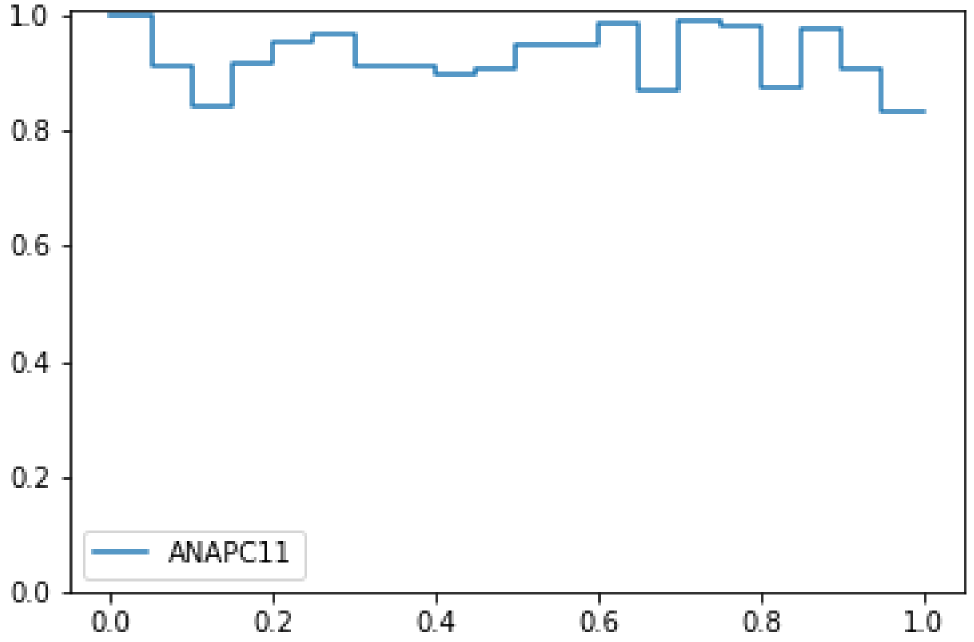

CCND1
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

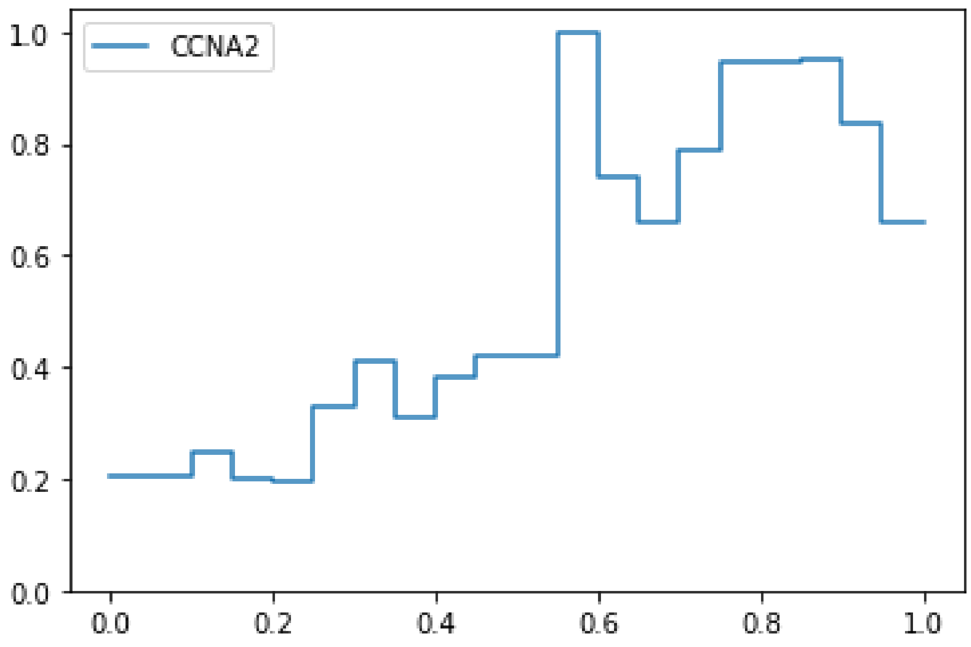

CCND3
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

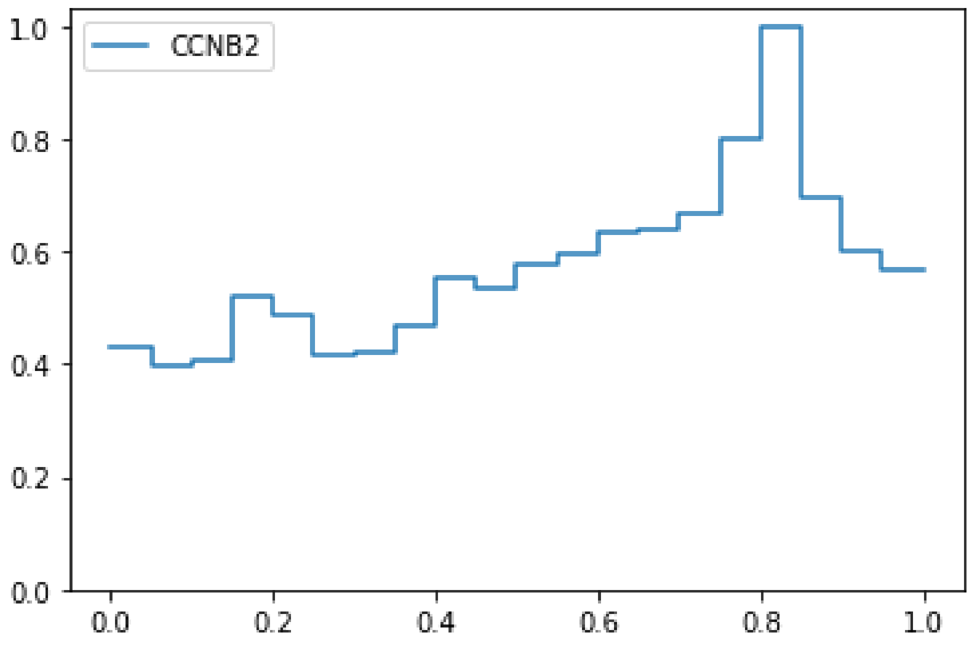

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
CCND2

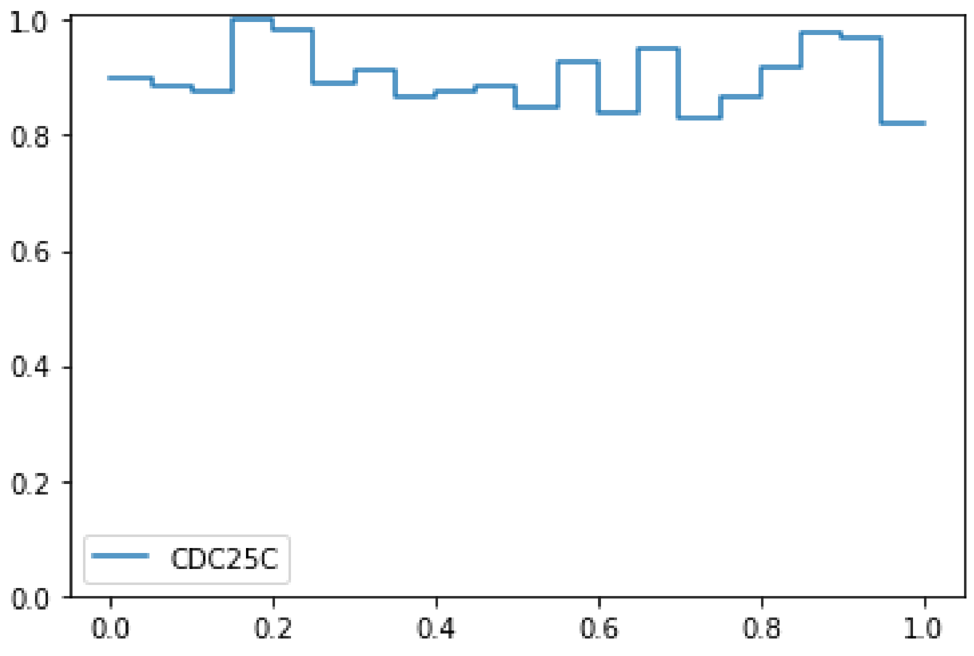

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
CDKN1A

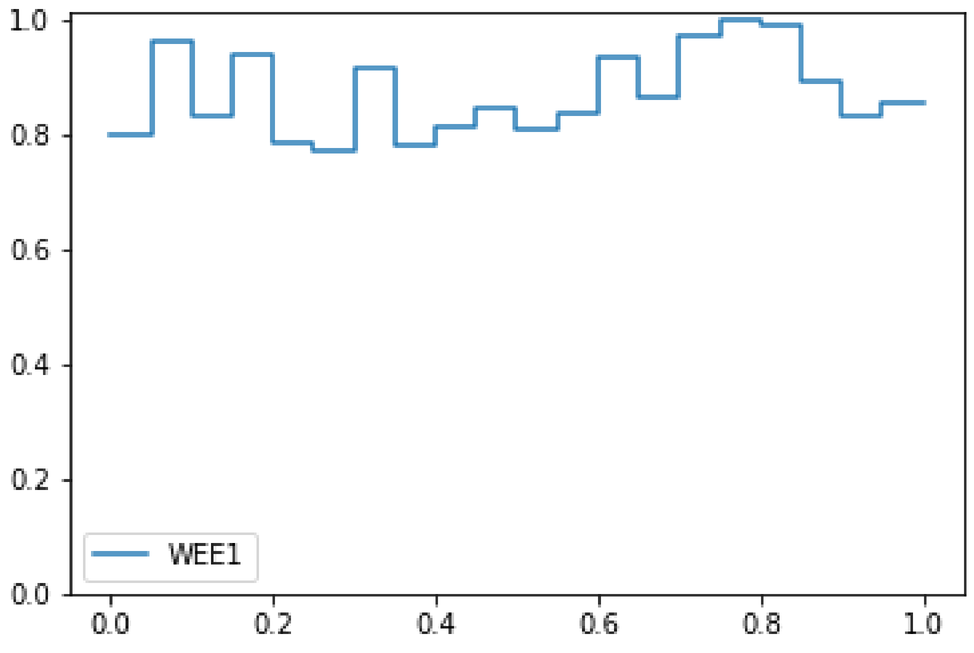

CCNB1
0.0
0.2
0.4
0.6
0.8
1.0


ANAPC11
0.0
0.2
0.4
0.6
0.8
1.0


CCNA2
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


CCNB2
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0





Conclusions: * A and B type cyclins get degraded earlier than I would have expected (perhaps due to contamination with mitotic cells?).

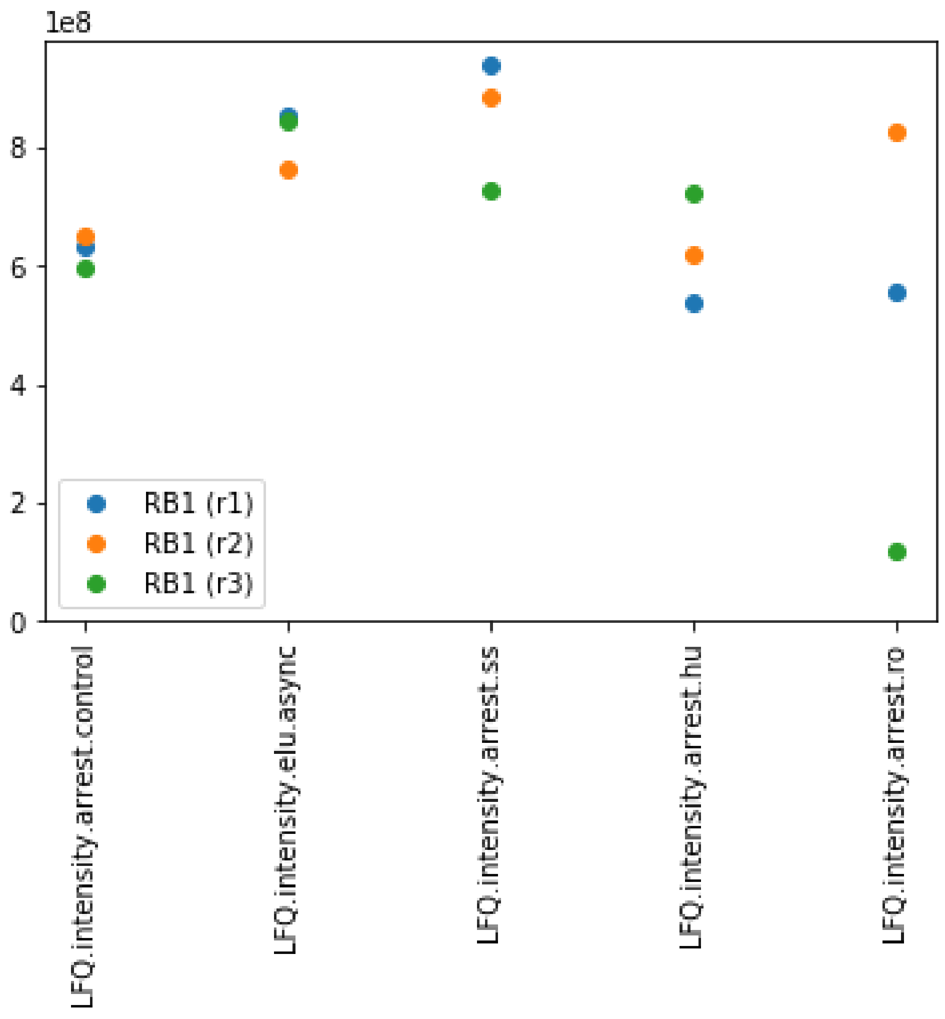
5

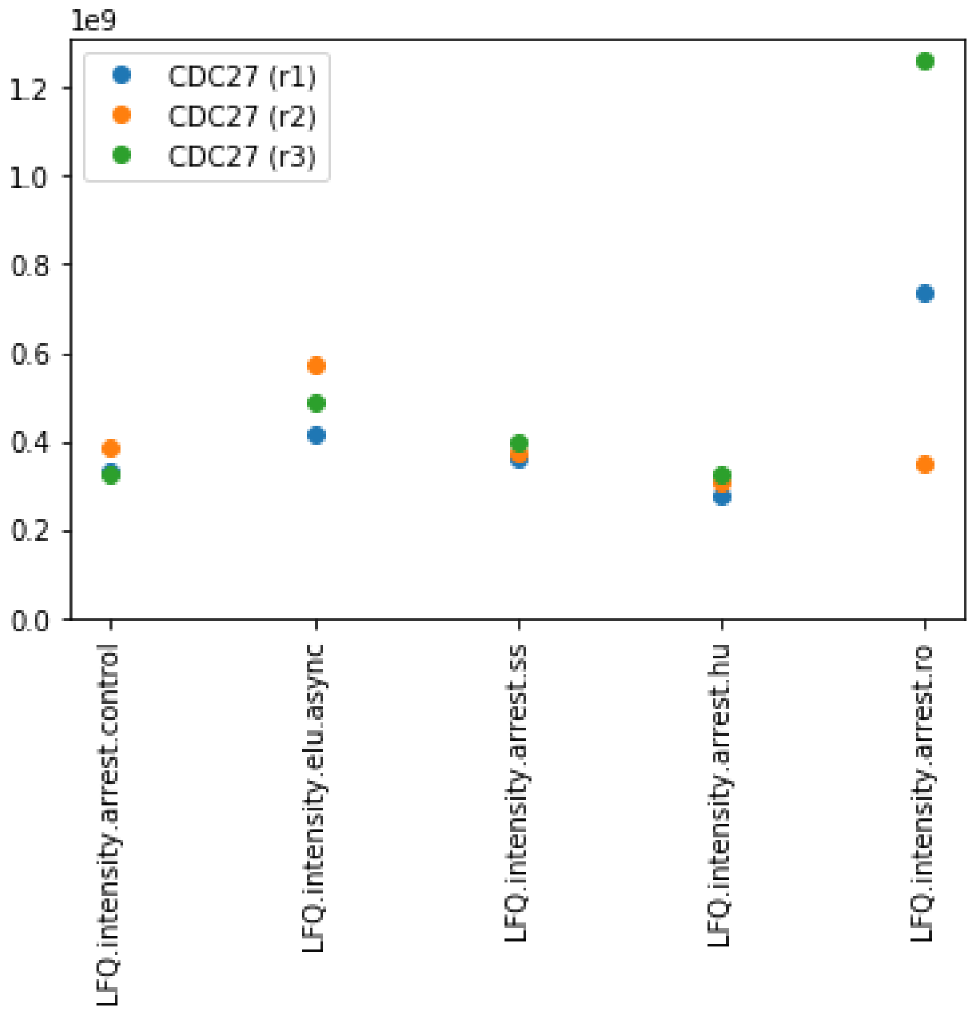
6 Ly et al. 2015 - Cell cycle arrests

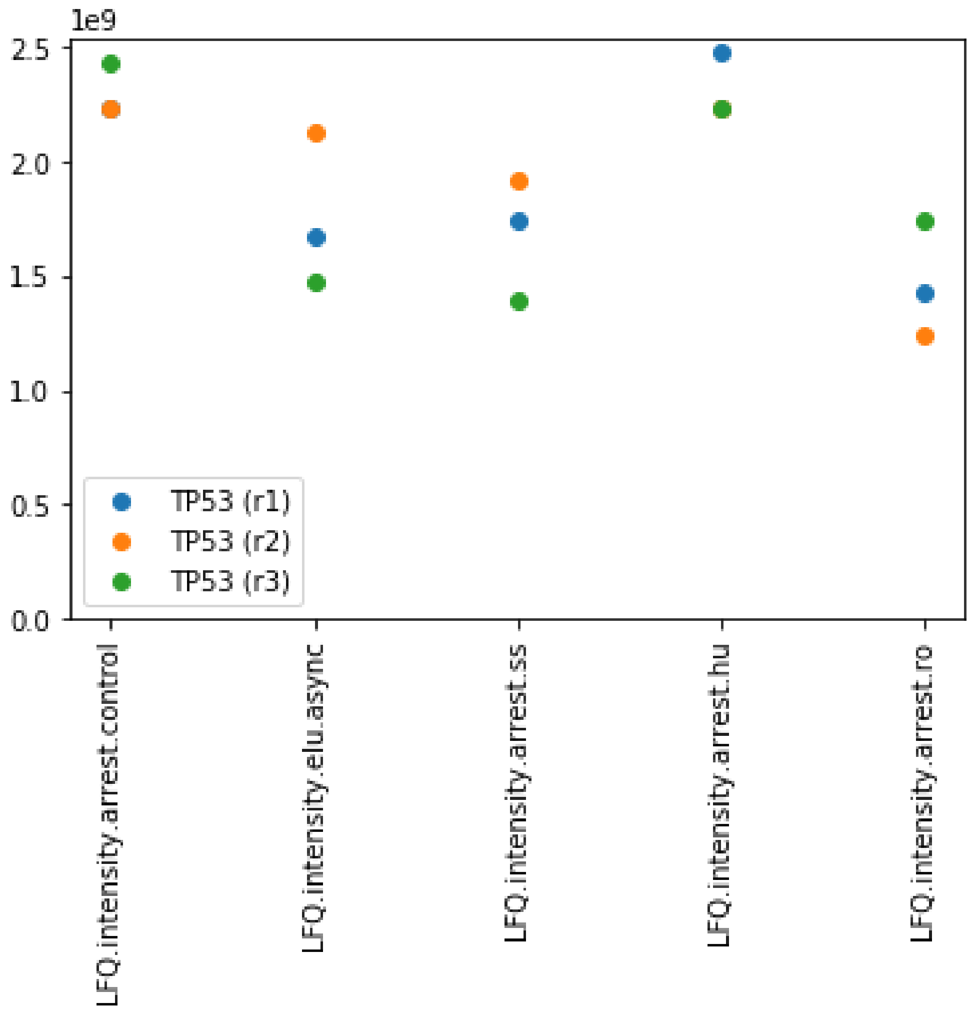
[ ]:
```
##################
# CC arest
# asynchromous cells this study
# asynchronous cells elutriation study
# 48 h serum starvation
# 18 h hydroxyurea (depletes deoxynucleotides)
# 18 h RO-3306 (specifically inhibits CDK1 activity)
##################

import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
from ast import literal_eval

# Enter (part of) gene names to plot
display = []

directory = os.path.abspath('')
if not display:
    data_file = os.path.join(directory, 'merged.xlsx')
    df = pd.read_excel(data_file)
    display = list(set(df.loc[:, 'Gene name']))
    print(df.loc[:, ['Gene name', 'ccModel Paul ID']].drop_duplicates().
 ↪dropna())
    display = [val for val in display if isinstance(val, str)]

# Fetch data of interest and prepare for plotting
data_file = os.path.join(directory, 'elife-04534-supp1-v1.xlsx')
df = pd.read_excel(data_file)

genes_of_interest = []
for string in display:
    for item in list(df.loc[:, 'Gene.names']):
        if string == str(item):
            genes_of_interest.append(item)

rows_of_interest = [True if item in genes_of_interest else False for item in␣
 ↪list(df['Gene.names'])]
df_of_interest = df.loc[rows_of_interest]

time = ['LFQ.intensity.arrest.control', 'LFQ.intensity.elu.async', 'LFQ.
 ↪intensity.arrest.ss',
        'LFQ.intensity.arrest.hu', 'LFQ.intensity.arrest.ro']

data_to_plot = {}
replicates = ['r1', 'r2', 'r3']
for gene in genes_of_interest:
    row = df_of_interest.loc[df_of_interest['Gene.names'] == gene]
    dict_of_replicates = {}
    for item in replicates:
        abundances = []
        for cc_phase in time:
            column_name = '{}.{}'.format(cc_phase, item)
            abundances.append(row.iloc[0][column_name])

        dict_of_replicates[item] = abundances
    data_to_plot[gene] = dict_of_replicates

# Plot
for gene in data_to_plot:
    plt.figure()
    for replicate in replicates:
        plt.plot(time, data_to_plot[gene][replicate], linestyle="", marker="o")
    plt.gca().set_ylim(bottom=0)
    plt.gca().legend((gene+' ('+replicates[0]+')', gene+' ('+replicates[1]+')',␣
 ↪gene+' ('+replicates[2]+')'), loc='best')
    for tick in plt.gca().get_xticklabels():
        tick.set_rotation(90)
```

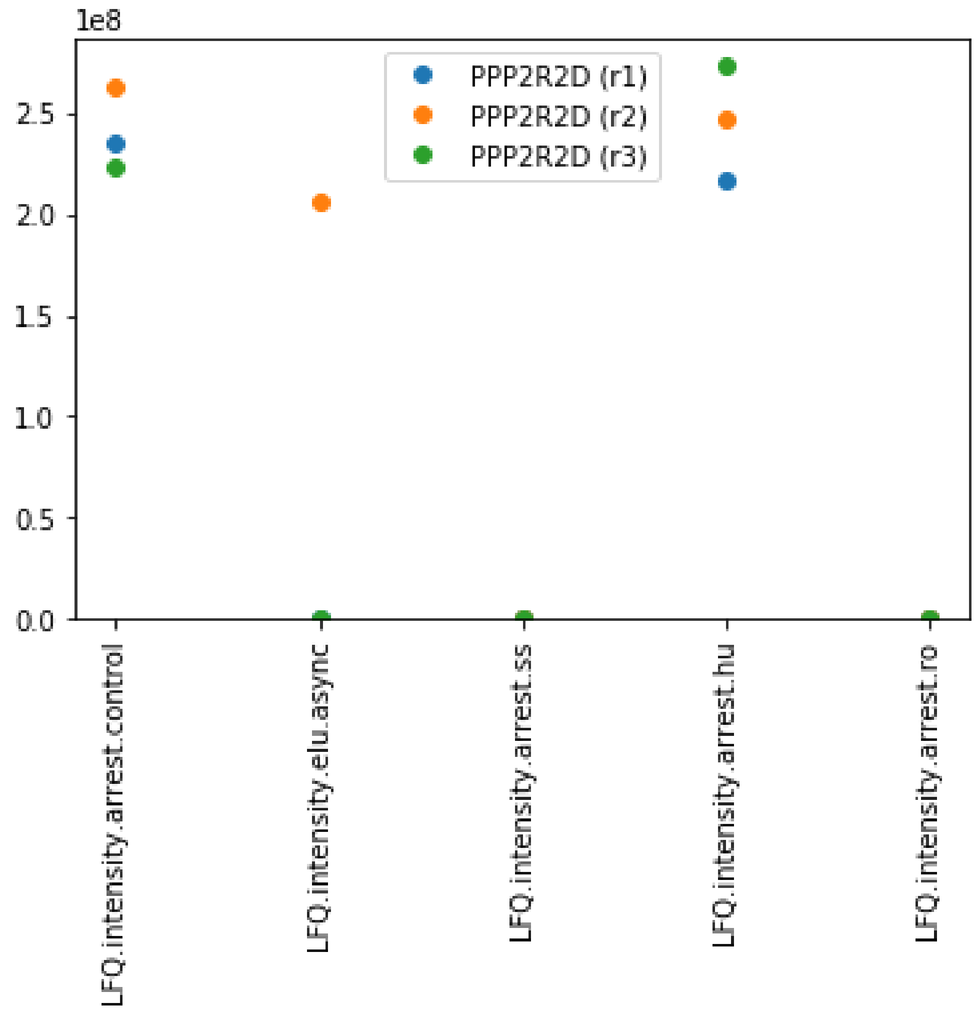
```
   Gene name ccModel Paul ID
0      CDC27             Apc
20      E2F2             E2f
22    ANAPC4             Apc
33   PPP2R2C             B55
42    ANAPC5             Apc
66     CDC23             Apc
75    CDC25B           Cdc25
84      E2F1             E2f
85     CCNE1              Ce
92      FZR1             Cdh
100    CCND1              Cd
112    FOXM1             Fox
122   CDKN1B             p27
126    FBXO5             Emi
129     E2F3             E2f
132    CCND3              Cd
155    CDC20           Cdc20
159    CCND2              Cd
163    MASTL              Gw
```

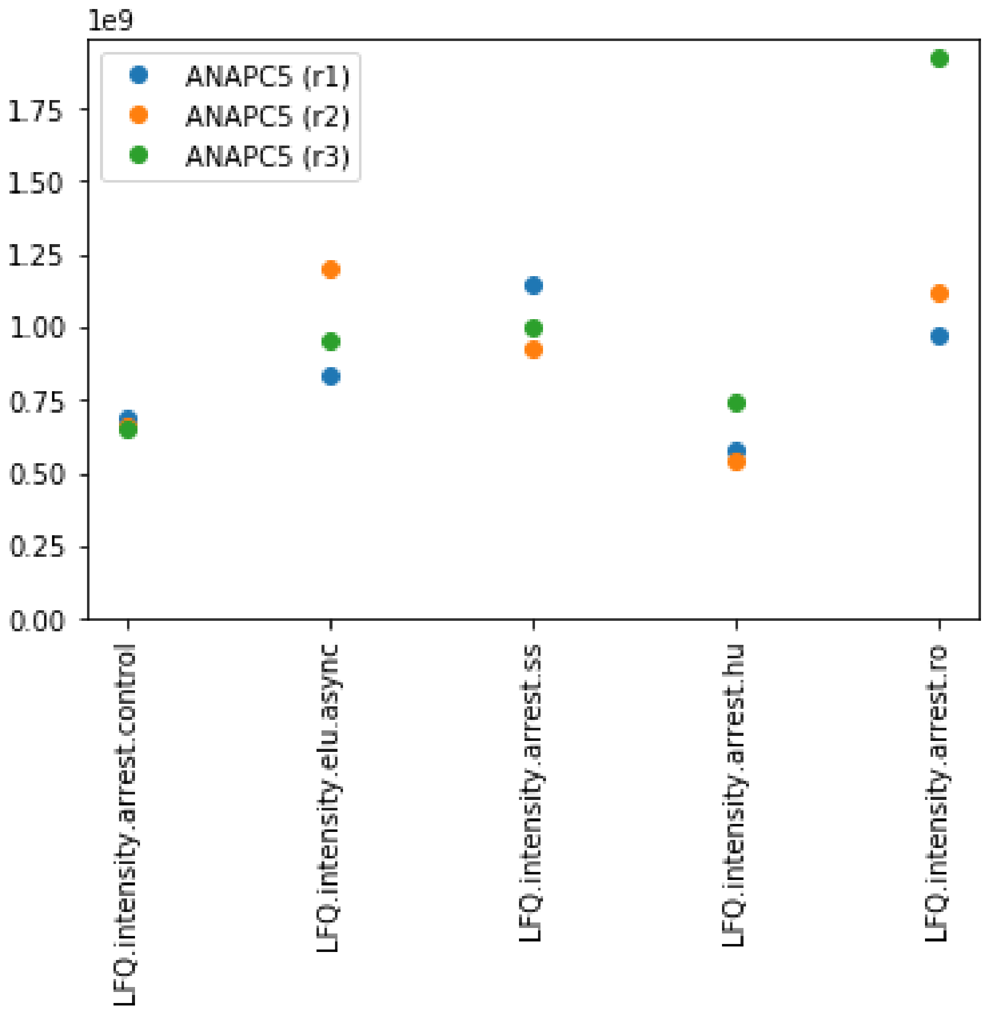
167	CDKN1A	p21
175	ARPP19	Ensa
188	CDC16	Apc
200	CCNA1	Ca
207	CCNB1	Cb
214	RB1	Rb
220	TP53	p53
248	ANAPC11	Apc
271	ENSA	Ensa
286	CCNA2	Ca
288	CCNB3	Cb
296	ANAPC1	Apc
306	PPP2R2B	B55
329	CCNB2	Cb
334	CDC25C	Cdc25
345	CDC25A	Cdc25
350	ANAPC10	Apc
367	WEE1	Wee
375	CDK1	pCb
384	CCNE2	Ce
392	PPP2R2D	B55
400	ANAPC2	Apc
408	CDC26	Apc
412	ANAPC7	Apc
424	PPP2R2A	B55

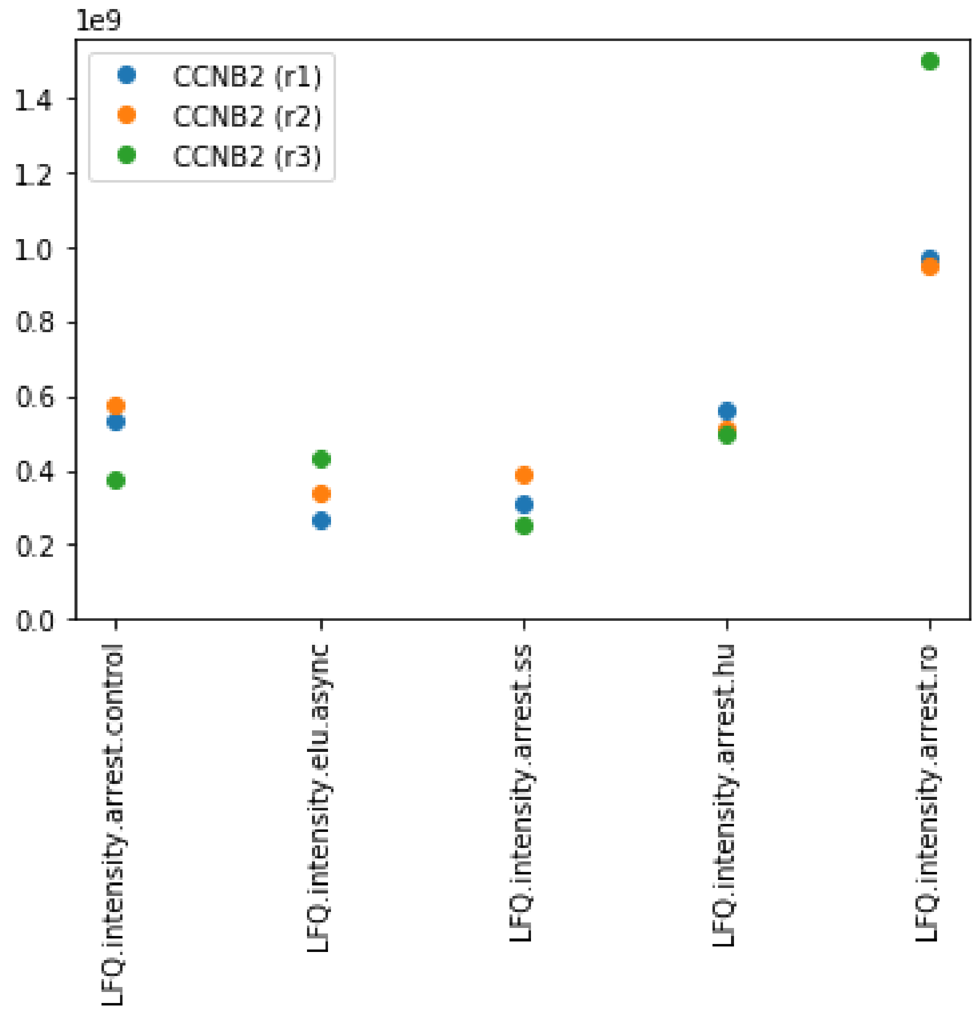

1e8
8
6
4
2
0
RB1 (r1)
RB1 (r2)
RB1 (r3)
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro

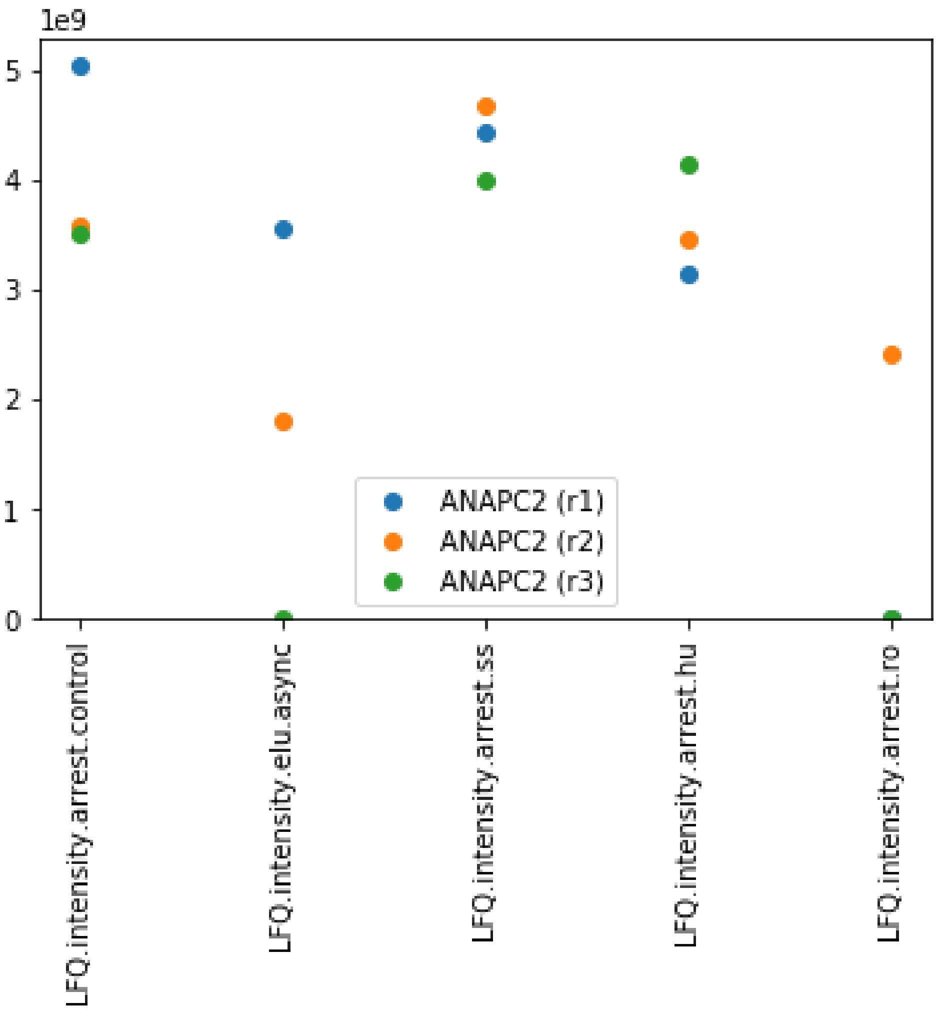

1e9
CDC27 (r1)
CDC27 (r2)
CDC27 (r3)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro

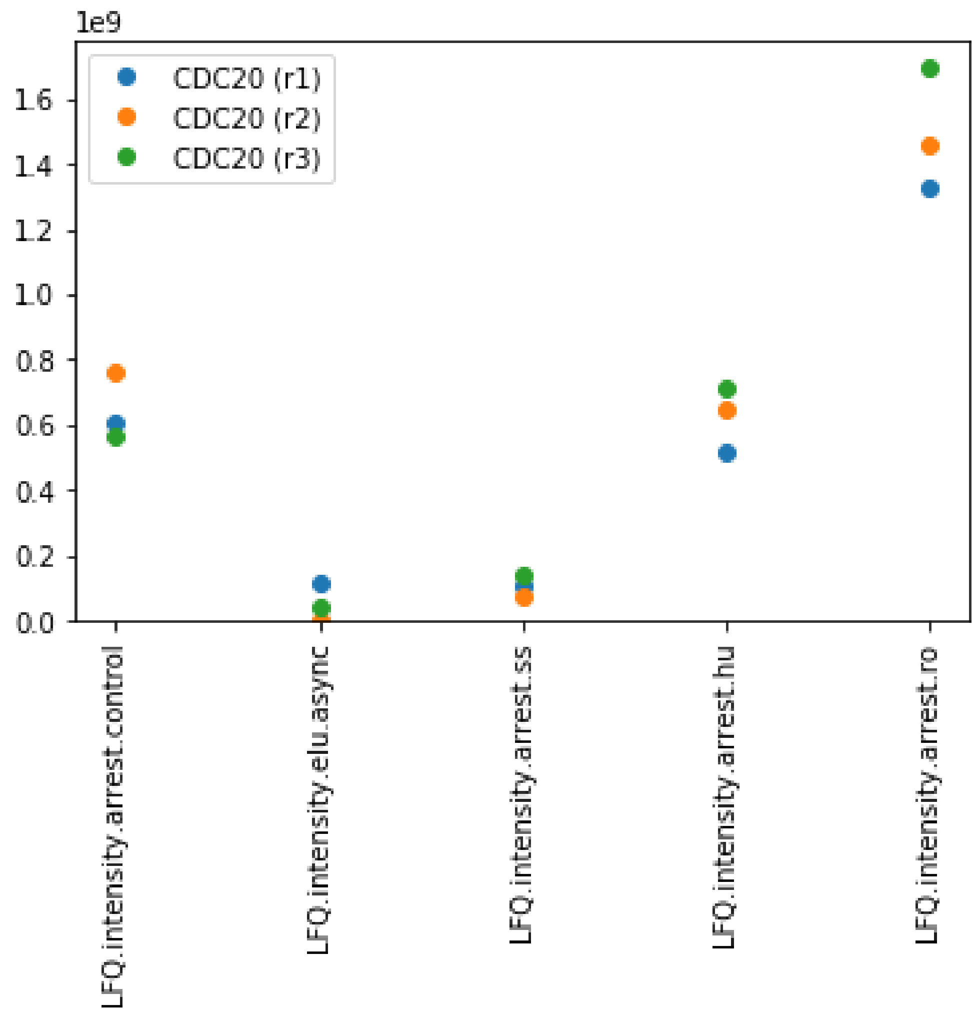

1e9
2.5
2.0
1.5
1.0
0.5
0.0
TP53 (r1)
TP53 (r2)
TP53 (r3)
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro

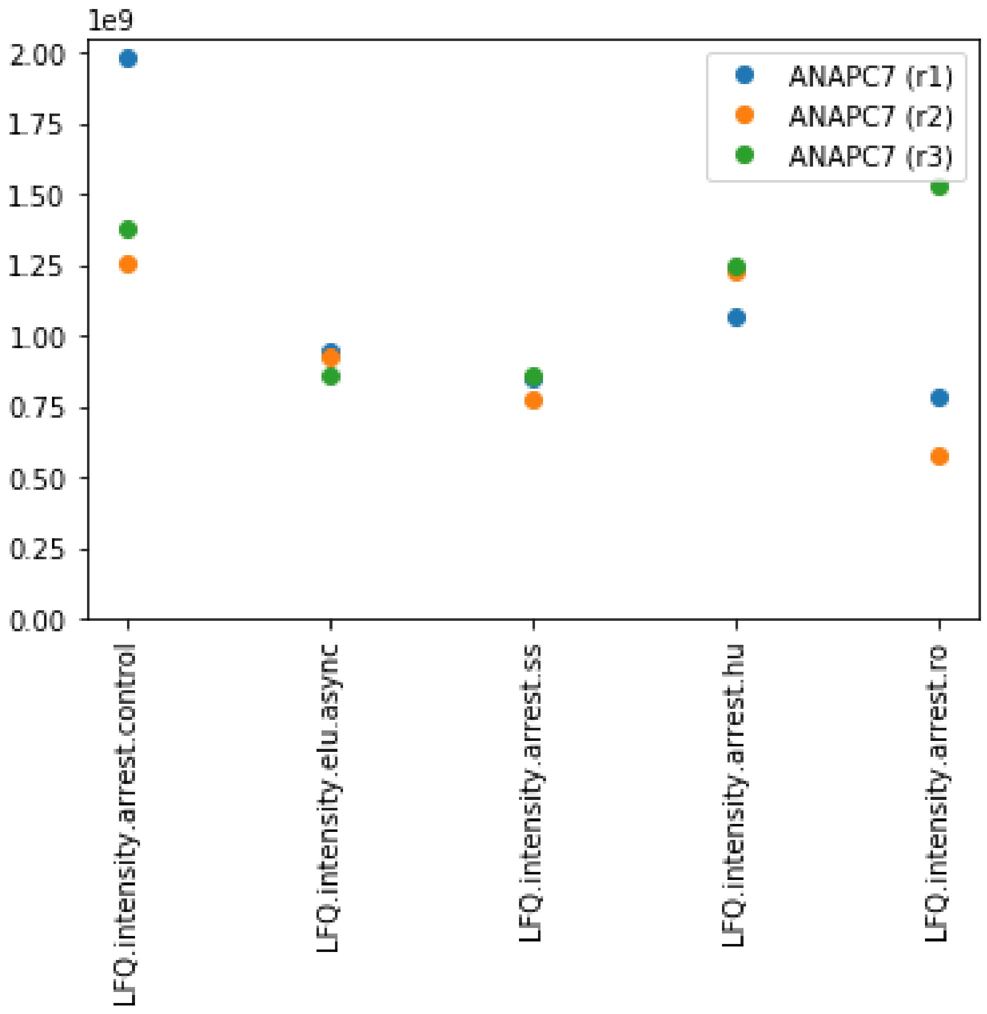

1e8
2.5
2.0
1.5
1.0
0.5
0.0
PPP2R2D (r1)
PPP2R2D (r2)
PPP2R2D (r3)
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro

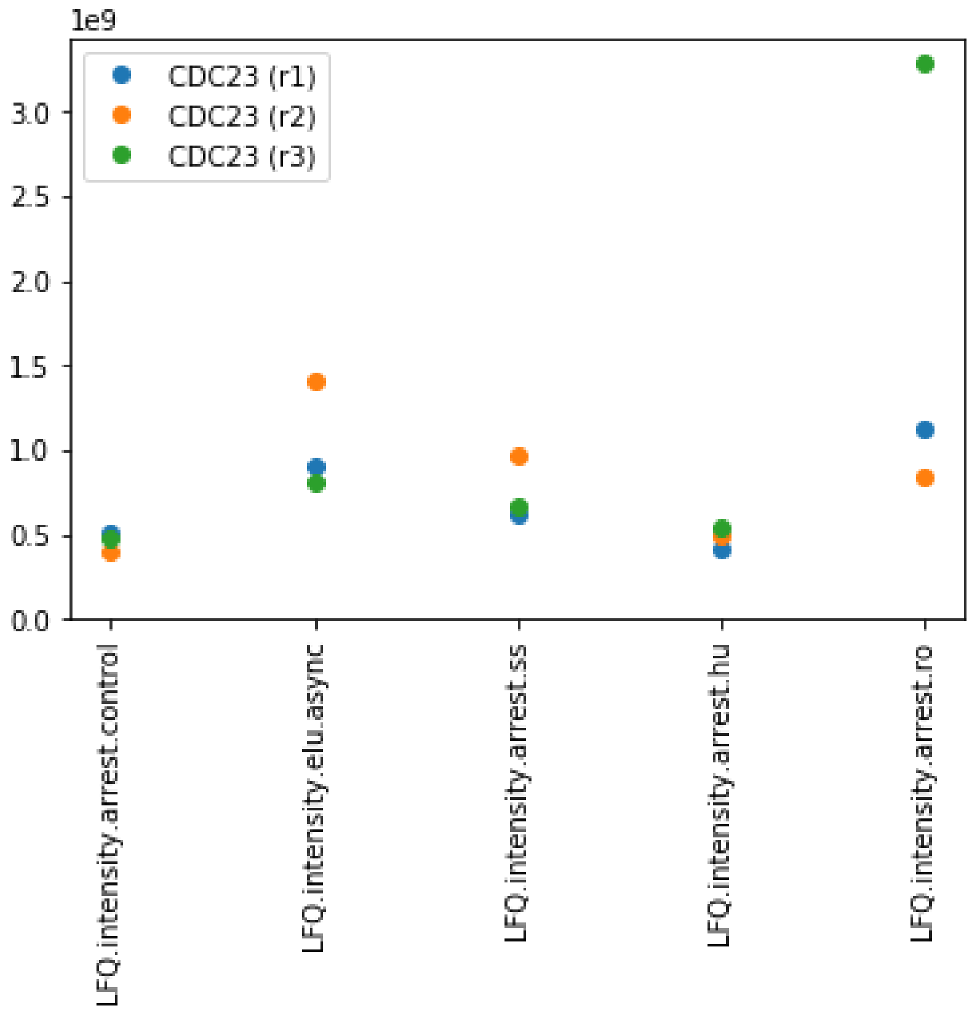

1e9
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
ANAPC5 (r1)
ANAPC5 (r2)
ANAPC5 (r3)
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro

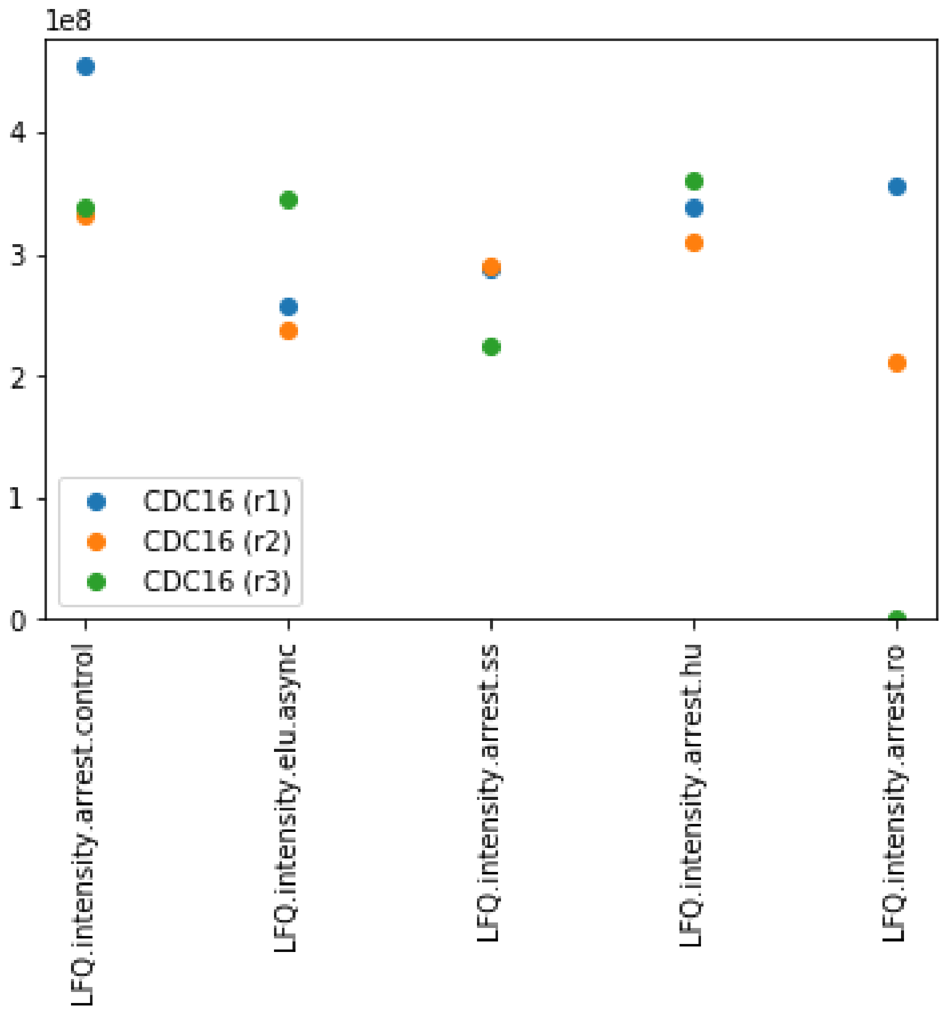

1e9
CCNB2 (r1)
CCNB2 (r2)
CCNB2 (r3)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro

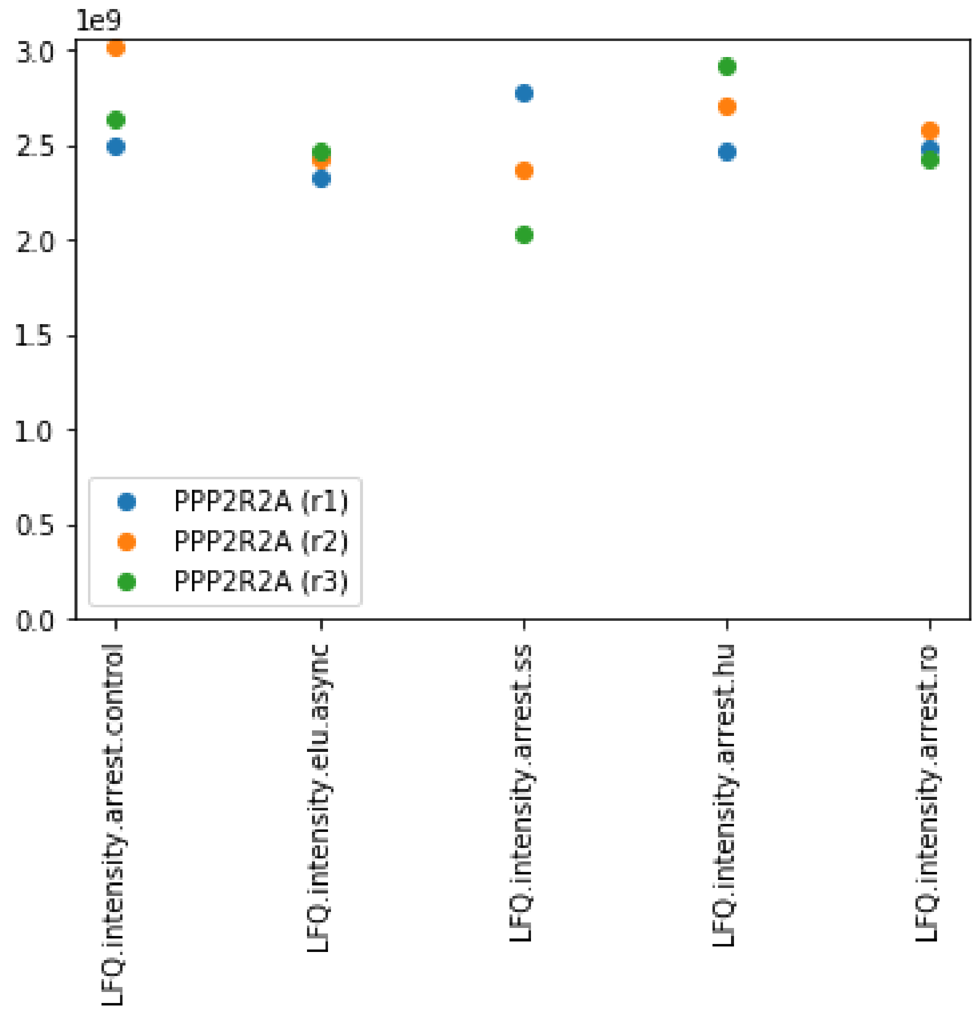

1e9
5
4
3
2
1
0
ANAPC2 (r1)
ANAPC2 (r2)
ANAPC2 (r3)
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro

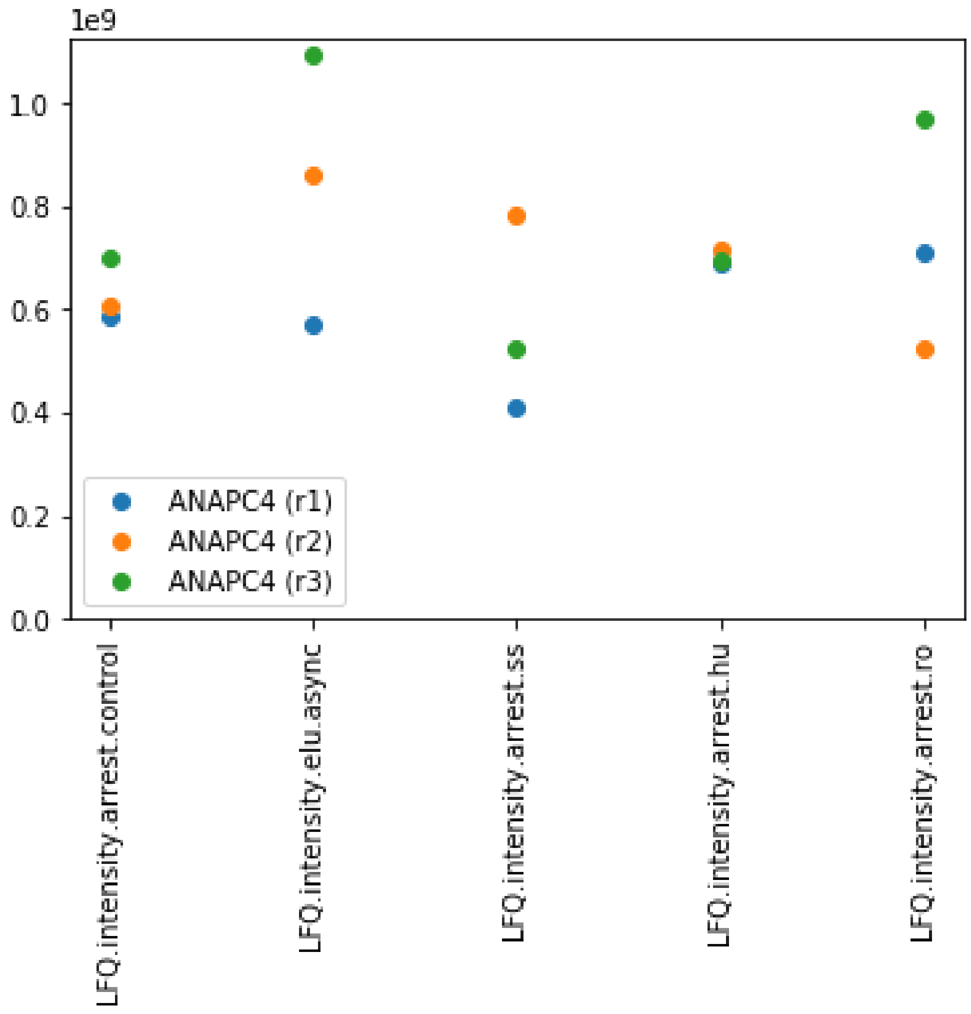

1e9
CDC20 (r1)
CDC20 (r2)
CDC20 (r3)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro


1e9
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
ANAPC7 (r1)
ANAPC7 (r2)
ANAPC7 (r3)
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro

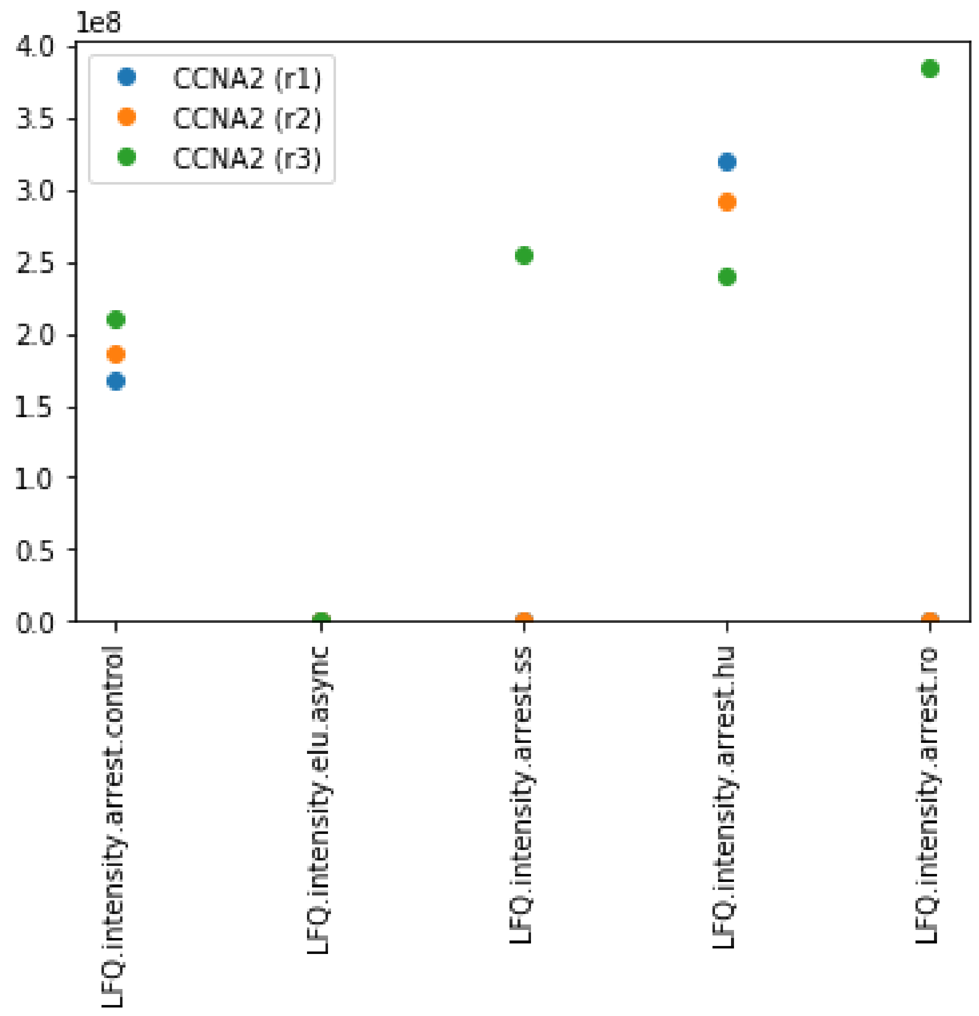

1e9
CDC23 (r1)
CDC23 (r2)
CDC23 (r3)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro

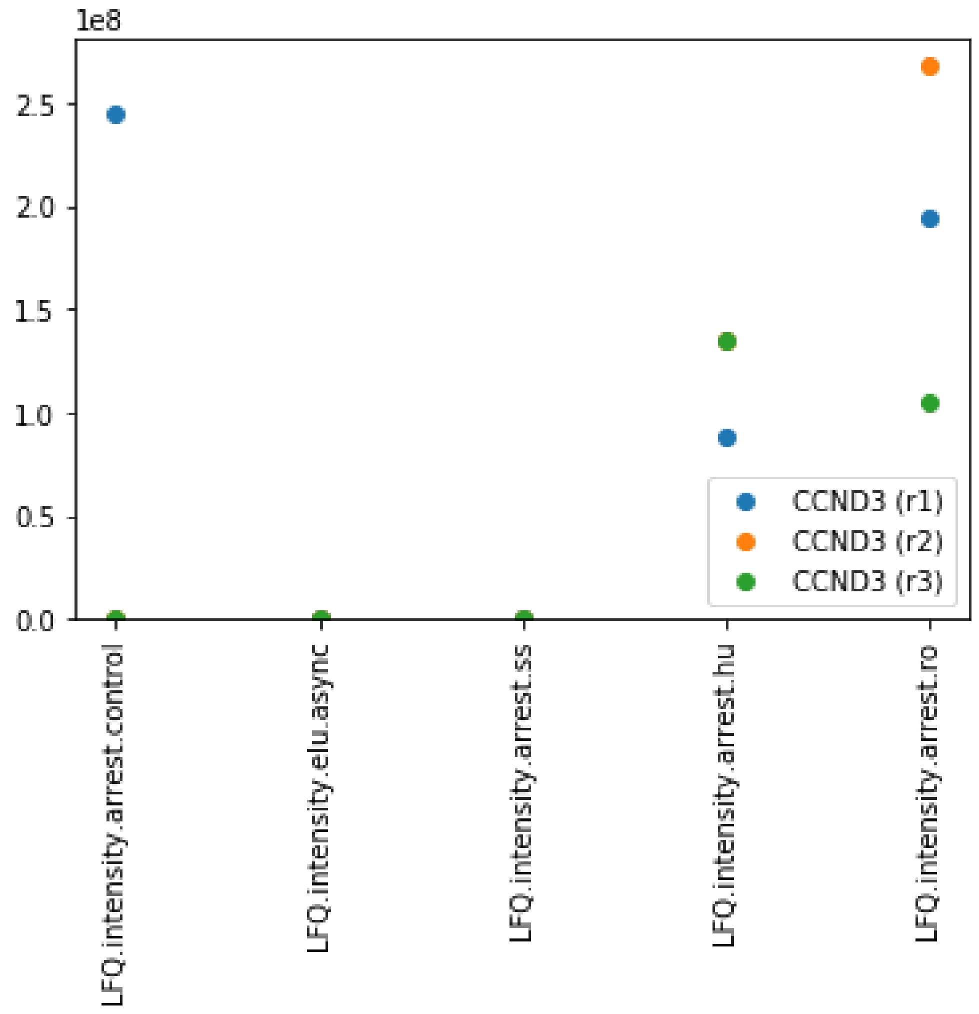

1e8
4
3
2
1
0
CDC16 (r1)
CDC16 (r2)
CDC16 (r3)
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro


1e9
3.0
2.5
2.0
1.5
1.0
0.5
0.0
PPP2R2A (r1)
PPP2R2A (r2)
PPP2R2A (r3)
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro

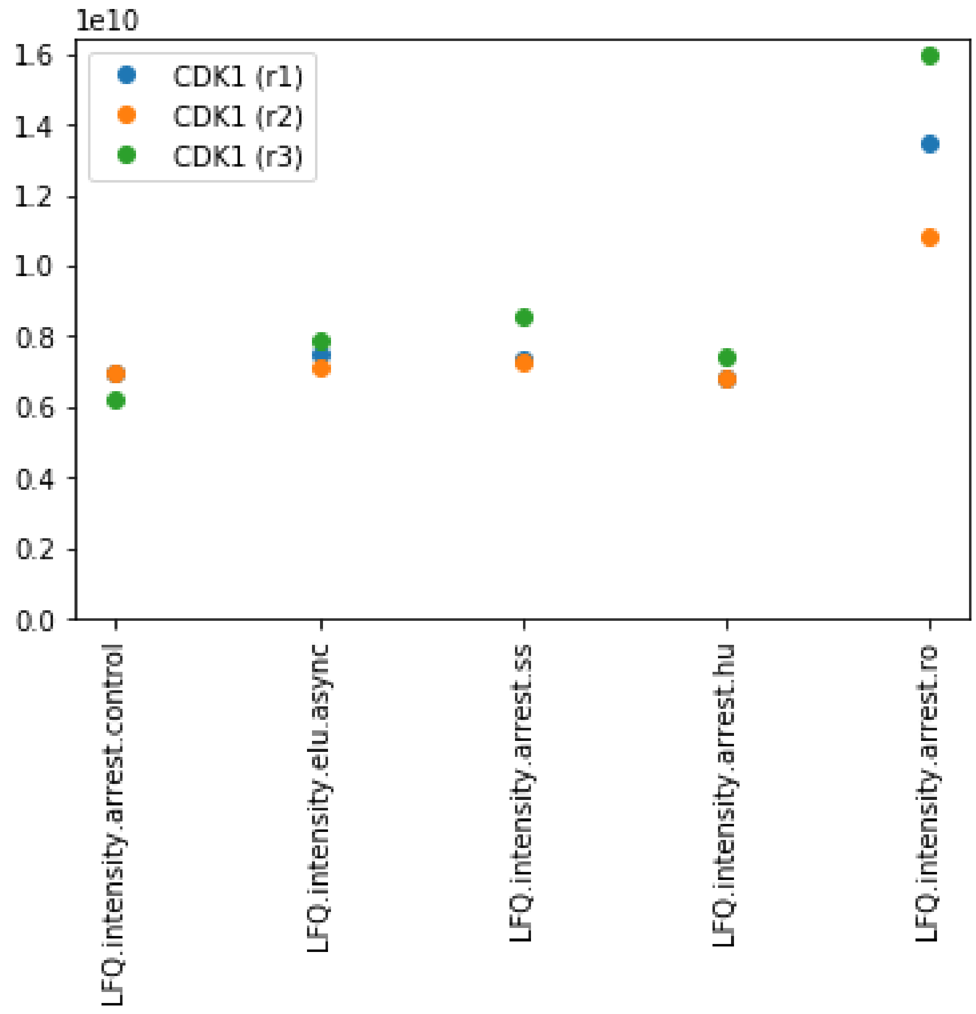


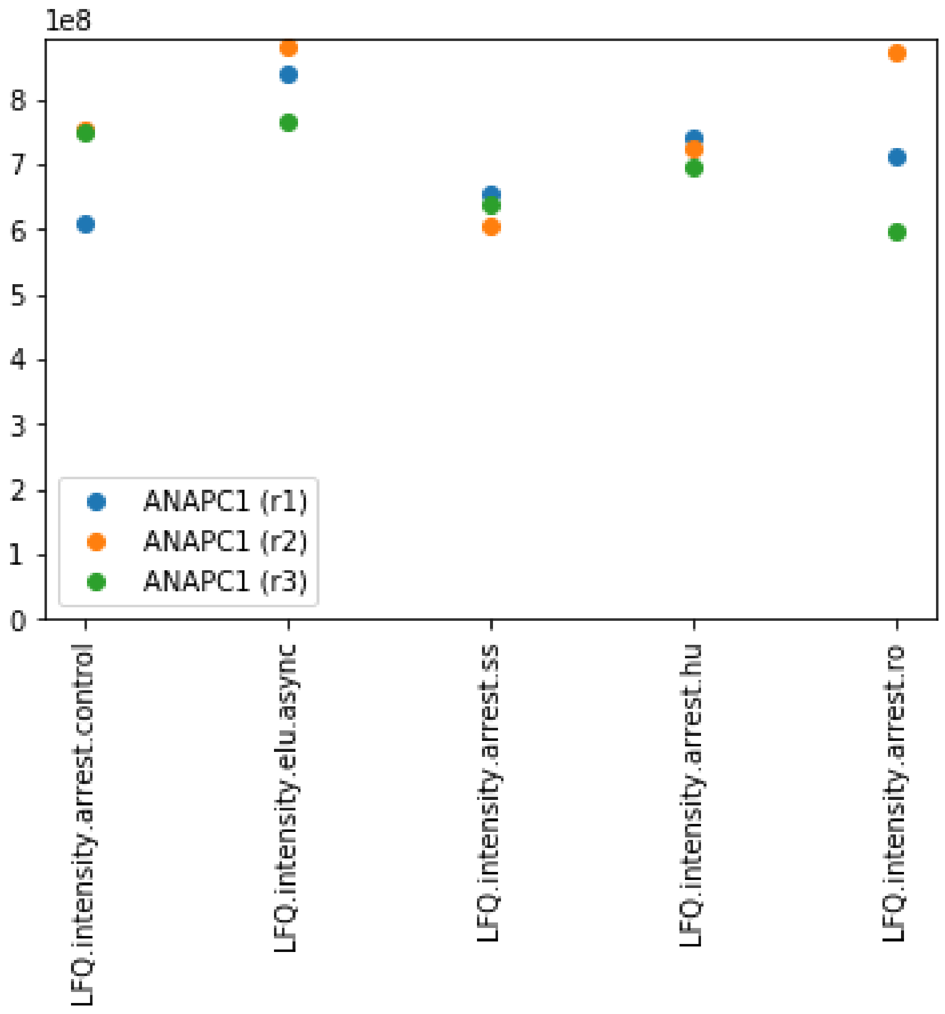
1e9
1.0
0.8
0.6
0.4
0.2
0.0
ANAPC4 (r1)
ANAPC4 (r2)
ANAPC4 (r3)
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro

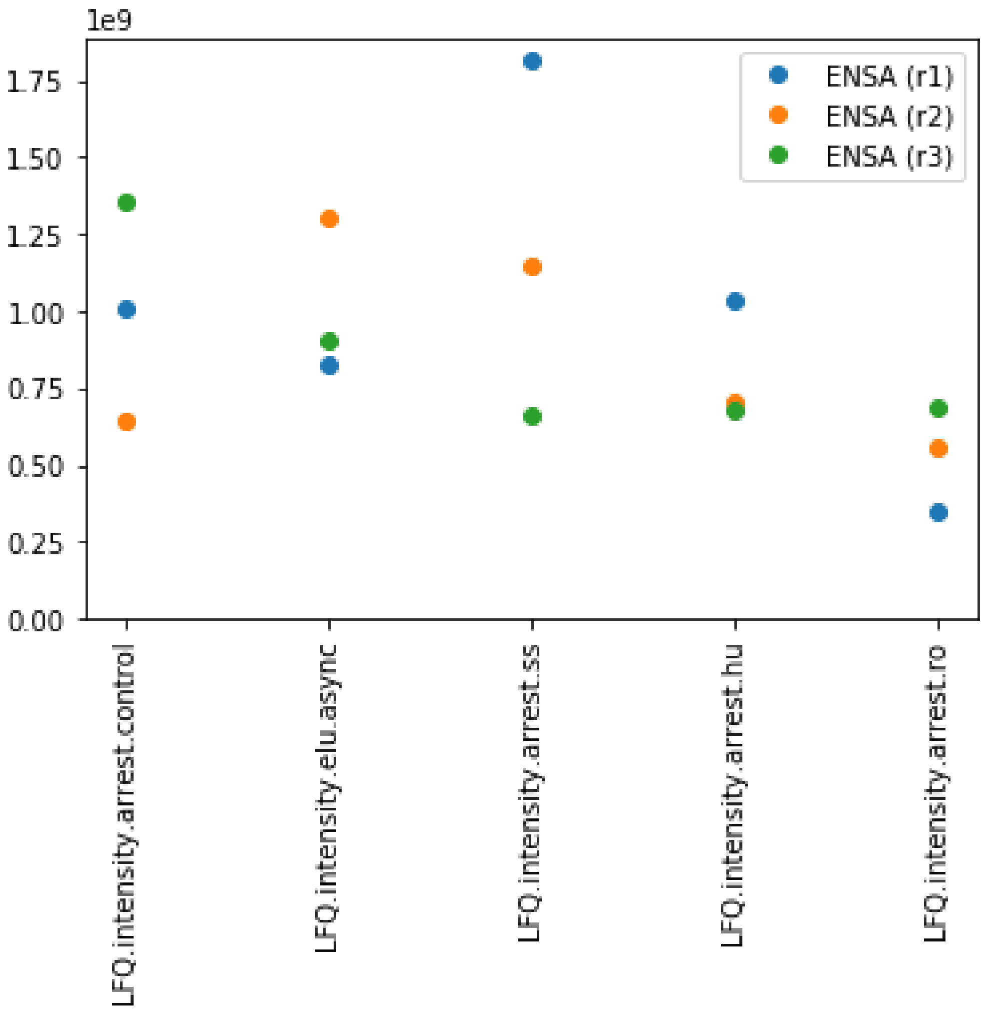

1e8
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
CCNA2 (r1)
CCNA2 (r2)
CCNA2 (r3)
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro

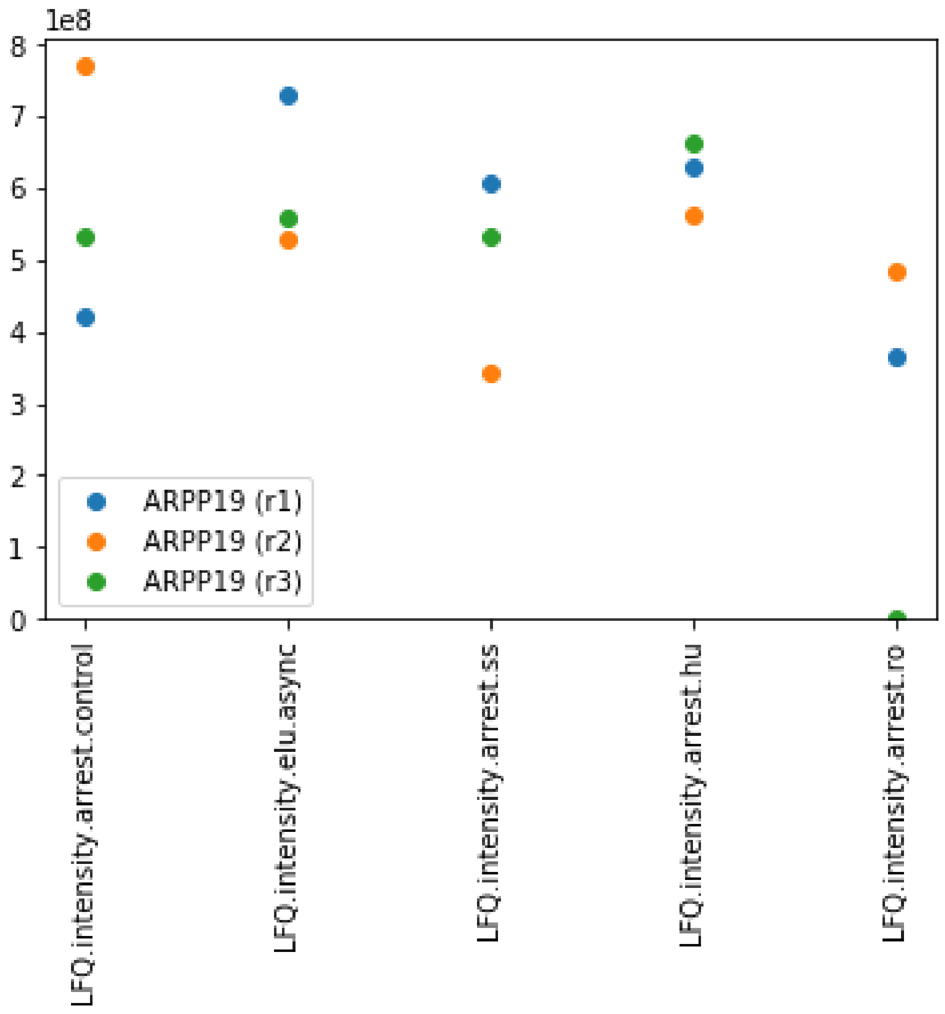


1e8
2.5
2.0
1.5
1.0
0.5
0.0
CCND3 (r1)
CCND3 (r2)
CCND3 (r3)
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro

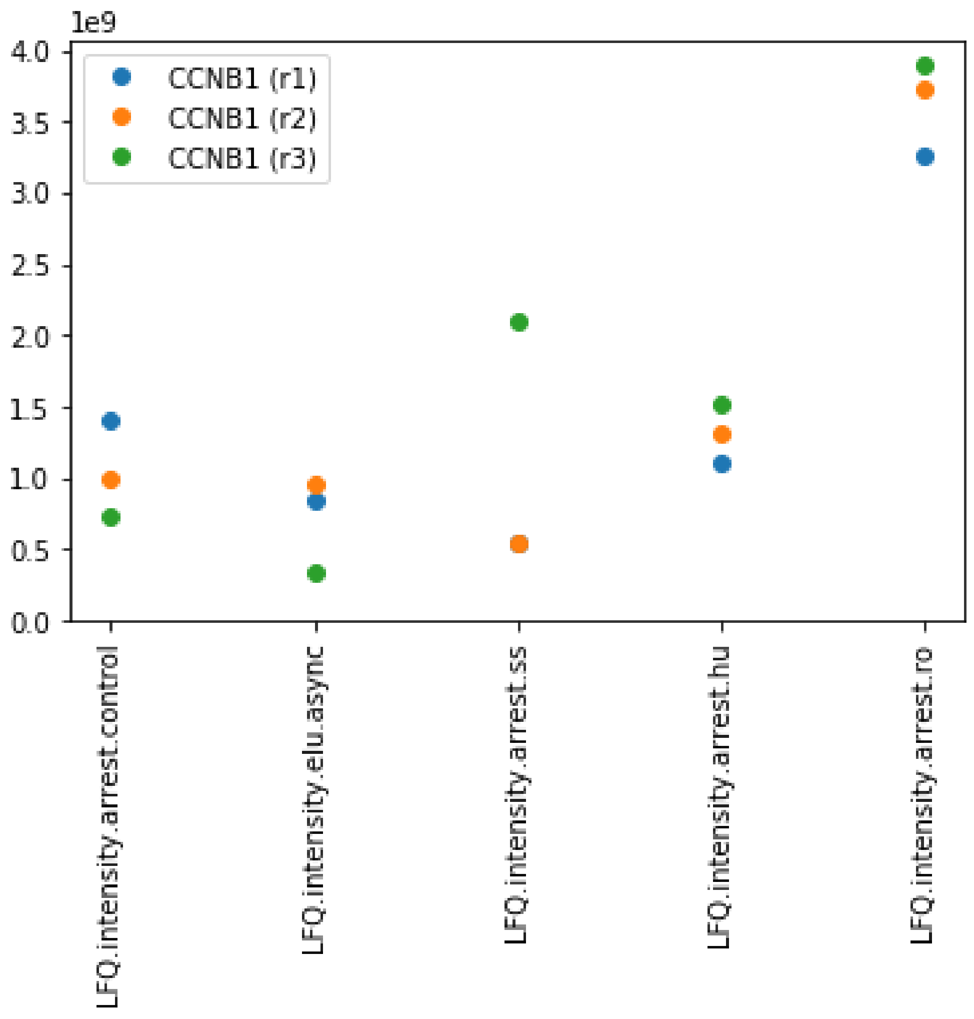

1e10
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
CDK1 (r1)
CDK1 (r2)
CDK1 (r3)
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro


1e8
0
1
2
3
4
5
6
7
8
ANAPC1 (r1)
ANAPC1 (r2)
ANAPC1 (r3)
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro


1e9
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
ENSA (r1)
ENSA (r2)
ENSA (r3)
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro



1e8
8
7
6
5
4
3
2
1
0
ARPP19 (r1)
ARPP19 (r2)
ARPP19 (r3)
LFQ.intensity.arrest.control
LFQ.intensity.elu.async
LFQ.intensity.arrest.ss
LFQ.intensity.arrest.hu
LFQ.intensity.arrest.ro

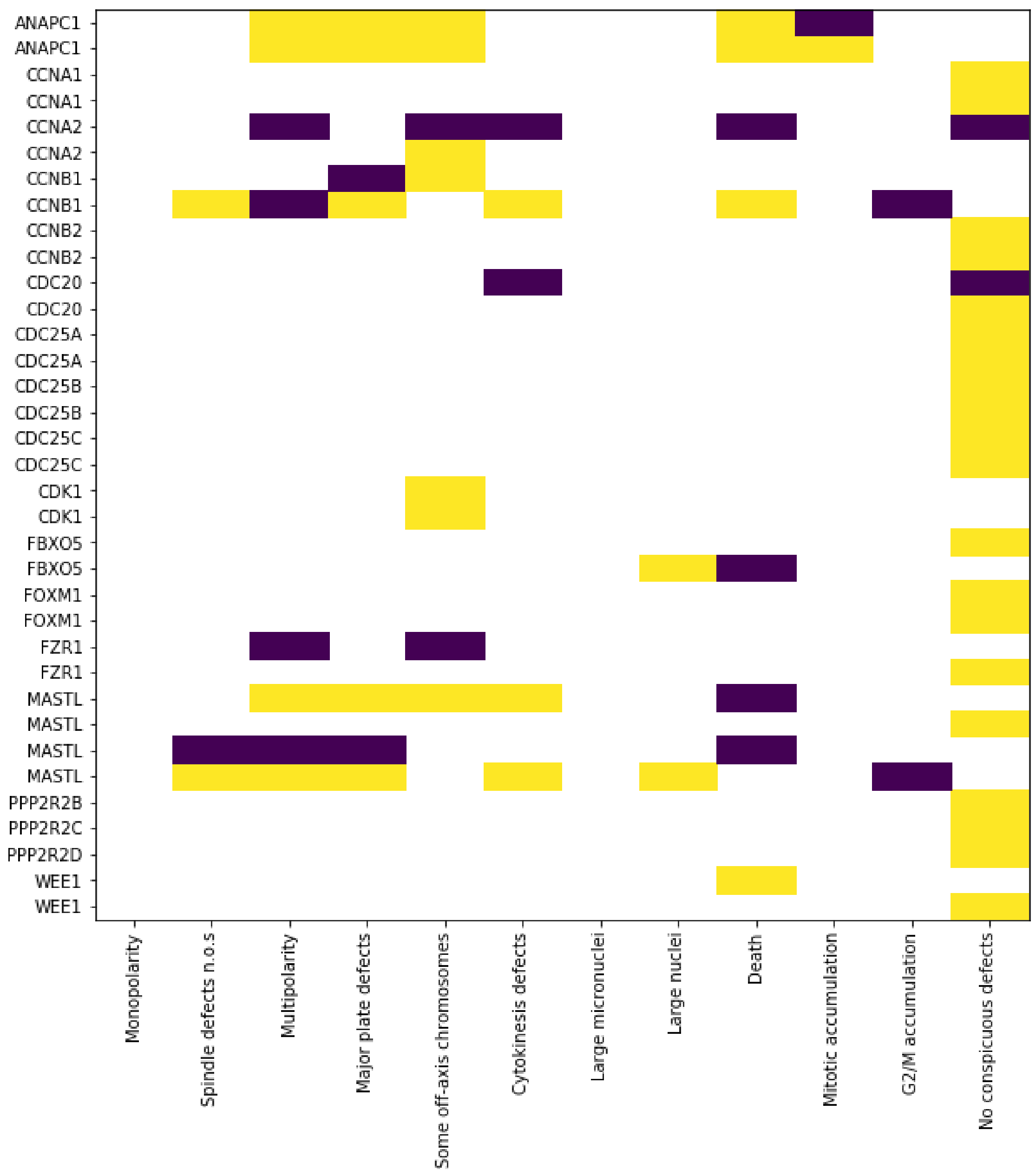


Conclusions: * CDC20, cyclin B1/2 and CDK1 rise from ss over hu to ro arrest. * Again, very noisy data

7 McKinley et Cheeseman - CRISPR screen in HeLa

[ ]:
```
##################
# Cripr/Cas9 KO in HeLa (Cheeseman)
##################

import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
from ast import literal_eval
```

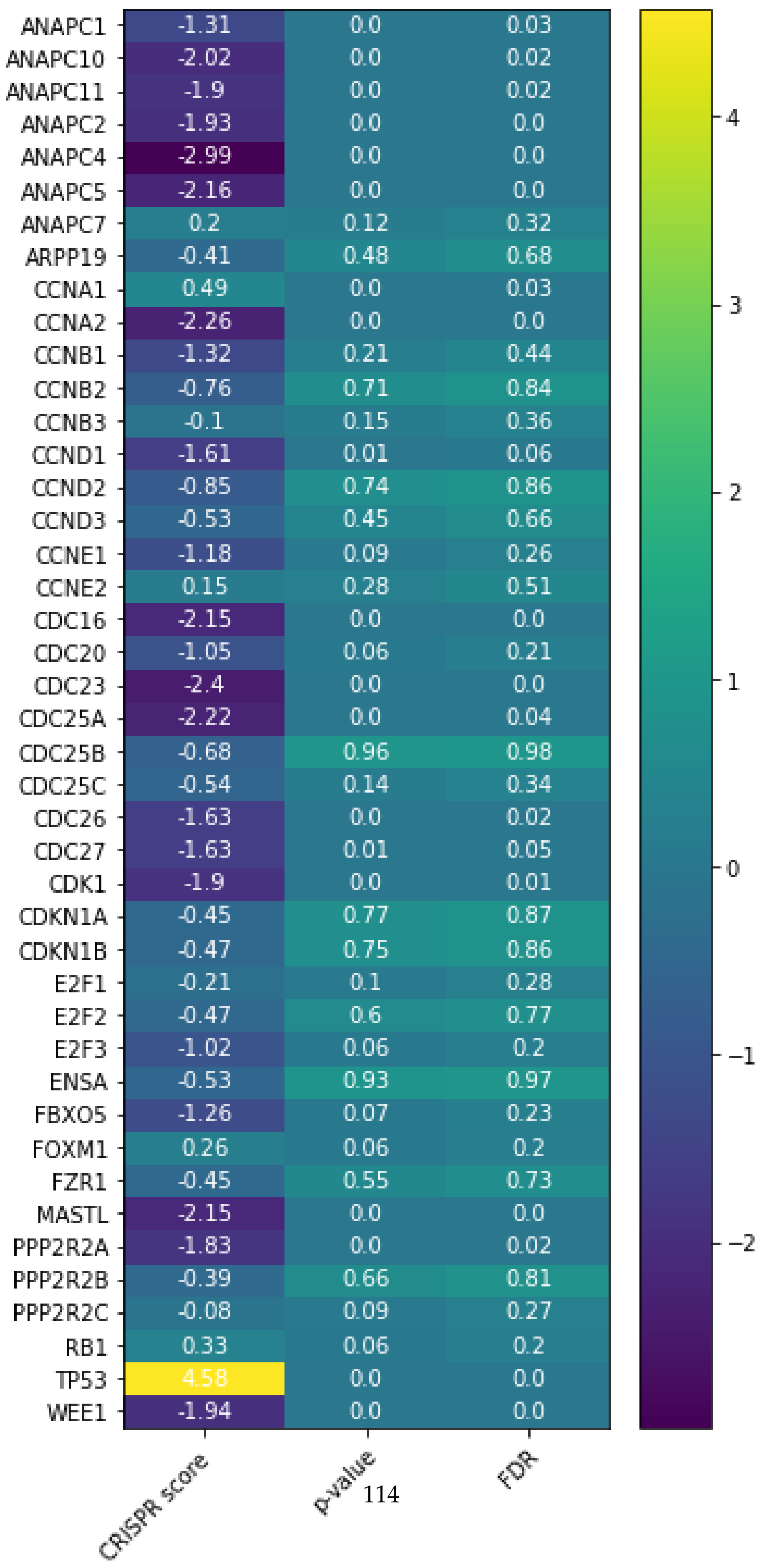
```
# Enter (part of) gene names to plot
display = []

directory = os.path.abspath('')
if not display:
    data_file = os.path.join(directory, 'merged.xlsx')
    df = pd.read_excel(data_file)
    display = list(set(df.loc[:, 'Gene name']))
    # print(df.loc[:, ['Gene name', 'ccModel Paul ID']].drop_duplicates().
 ↪dropna())
    display = [val for val in display if isinstance(val, str)]

# Fetch data of interest and prepare for plotting
data_file = os.path.join(directory, '1-s2.0-S1534580717300394-mmc5.xlsx')
df = pd.read_excel(data_file)

genes_of_interest = []
for string in display:
    for item in list(df.loc[:, 'Gene Target']):
        if string == str(item):
            genes_of_interest.append(item)

rows_of_interest = [True if item in genes_of_interest else False for item in␣
 ↪list(df['Gene Target'])]
cols_of_interest = [True if isinstance(item, np.float64) else False for item in␣
 ↪list(df.iloc[0])]
df_of_interest = df.loc[rows_of_interest, cols_of_interest]
df_of_interest.index = list(df.iloc[rows_of_interest, 0])
df_of_interest = df_of_interest.sort_index(ascending=False)

# Plot
plt.figure(figsize=(10,10))
plt.pcolor(df_of_interest)
plt.yticks(np.arange(0.5, len(df_of_interest.index), 1), df_of_interest.index)
plt.xticks(np.arange(0.5, len(df_of_interest.columns), 1), df_of_interest.
 ↪columns)
for tick in plt.gca().get_xticklabels():
    tick.set_rotation(90)
plt.show()

# They just induced KO with > 1 different gRNAs per gene and looked at the␣
 ↪phenotype 4 days later. They did not check if
# the KO worked and on how many chromosomes it worked. The frequency of␣
 ↪phenotype observation was classified into < 5%, 5-30%
# and > 30% of the population.
```



Conclusions: * The phenotypes do not add up to 100% of the population * CCNB1 and MASTL have several working gRNAs that differ in their effects. * HeLa may not be viable if ANAPC1, CCNA2 or FBXO5 are knocked out. * Data is not reliable.

8 Yilmaz et al. - CRISPR screen in H1 hESC

[ ]:
```
##################
# Cripr/Cas9 KO in H1 hESC (Yilamaz et al.)
##################

import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
from ast import literal_eval

# Enter (part of) gene names to plot
display = []

directory = os.path.abspath('')
if not display:
    data_file = os.path.join(directory, 'merged.xlsx')
    df = pd.read_excel(data_file)
    display = list(set(df.loc[:, 'Gene name']))
    display = [val for val in display if isinstance(val, str)]

# Fetch data of interest and prepare for plotting
data_file = os.path.join(directory, '41556_2018_88_MOESM3_ESM.xlsx')
df = pd.read_excel(data_file)

genes_of_interest = []
for string in display:
    for item in list(df.loc[:, 'Gene Symbol']):
        # print('The string is {}'.format(string))
        # print('The item is {}'.format(item))
        if string == str(item):
            genes_of_interest.append(item)

rows_of_interest = [True if item in genes_of_interest else False for item in␣
 ↪list(df['Gene Symbol'])]
cols_of_interest = [False, False, True, True, True]
df_of_interest = df.loc[rows_of_interest, cols_of_interest]
df_of_interest.index = list(df.iloc[rows_of_interest, 1])
df_of_interest = df_of_interest.sort_index(ascending=False)

# Plot
plt.figure(figsize=(5, 12))
plt.pcolormesh(df_of_interest)
plt.colorbar()
plt.yticks(np.arange(0.5, len(df_of_interest.index), 1), df_of_interest.index)
plt.xticks(np.arange(0.5, len(df_of_interest.columns), 1), df_of_interest.
 ↪columns, rotation=45, ha="right",
         rotation_mode="anchor")
# Loop over data dimensions and create text annotations.
for i in range(len(df_of_interest.index)):
    for j in range(len(df_of_interest.columns)):
        text = plt.text(j+0.5, i+0.5, round(df_of_interest.iloc[i, j], 2),
                       ha="center", va="center", color="w")
plt.show()

# Crispr score: the CRISPR scores are the average of the log2 ratios of the␣
 ↪abundance of all sgRNAs for each gene between
# final (day 23) and initial (day 1) populations post transfection. Note that␣
 ↪this does measure how much more competitive
# a given mutation is as compared to the rest of the cell population (i.e. a␣
 ↪population with approx. 70% WT cells and 30%
# cells carrying (largely growth promoting) mutations). In that sense the Cripr␣
 ↪score measures how much more growth promoting
# a given mutation is compared to the average mutation.
# Unfortunately, I do not know if the average mutation is growth promoting or␣
 ↪growth limiting (think of edge cases, such as all
# mutations are lethal, except of one, which is only slightly growth limiting;␣
 ↪although, the p-value closest to 1 is at a
# Crispr score of -0.68, what does that tell us? I think that the proportion of␣
 ↪WT cells has increased).
# Also note that the strength of the effect depends on the efficiency of each␣
 ↪of the
# 10 gRNAs in knocking out a gene (little effect can mean the gene was hard to␣
 ↪knock out).

# p-value: determined via a Kolmogorov-Smirnov test
```



	CRISPR score	p-value	FDR
ANAPC1	-1.31	0.0	0.03
ANAPC10	-2.02	0.0	0.02
ANAPC11	-1.9	0.0	0.02
ANAPC2	-1.93	0.0	0.0
ANAPC4	-2.99	0.0	0.0
ANAPC5	-2.16	0.0	0.0
ANAPC7	0.2	0.12	0.32
ARPP19	-0.41	0.48	0.68
CCNA1	0.49	0.0	0.03
CCNA2	-2.26	0.0	0.0
CCNB1	-1.32	0.21	0.44
CCNB2	-0.76	0.71	0.84
CCNB3	-0.1	0.15	0.36
CCND1	-1.61	0.01	0.06
CCND2	-0.85	0.74	0.86
CCND3	-0.53	0.45	0.66
CCNE1	-1.18	0.09	0.26
CCNE2	0.15	0.28	0.51
CDC16	-2.15	0.0	0.0
CDC20	-1.05	0.06	0.21
CDC23	-2.4	0.0	0.0
CDC25A	-2.22	0.0	0.04
CDC25B	-0.68	0.96	0.98
CDC25C	-0.54	0.14	0.34
CDC26	-1.63	0.0	0.02
CDC27	-1.63	0.01	0.05
CDK1	-1.9	0.0	0.01
CDKN1A	-0.45	0.77	0.87
CDKN1B	-0.47	0.75	0.86
E2F1	-0.21	0.1	0.28
E2F2	-0.47	0.6	0.77
E2F3	-1.02	0.06	0.2
ENSA	-0.53	0.93	0.97
FBXO5	-1.26	0.07	0.23
FOXM1	0.26	0.06	0.2
FZR1	-0.45	0.55	0.73
MASTL	-2.15	0.0	0.0
PPP2R2A	-1.83	0.0	0.02
PPP2R2B	-0.39	0.66	0.81
PPP2R2C	-0.08	0.09	0.27
RB1	0.33	0.06	0.2
TP53	4.58	0.0	0.0
WEE1	-1.94	0.0	0.0

Conclusions: * APC ko seems lethal no matter which subunit you hit. * CCNA2 ko seems lethal, whereas CCNA1 ko seems growth promoting (note that CCNA1 is expressed in testis and brain, as well as in several leukemic cell lines, and is thought to primarily function in the control of meiosis)! * CDC25A and WEE1 ko seems lethal, B not, C maybe. * CDK1 ko seems lethal. * MASTL and PP2R2A (maybe C, but not B) ko seems lethal * TP53 ko is growth promoting.

- The fold change is a result from average KO efficiency of the 10 gRNAs and KO effect.

9 Neumann et al. - siRNA screen

[ ]:
```
##################
# siRNA KD in HeLa (Neumann et al.)
##################

import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
import requests
from ast import literal_eval
import copy
from IPython.display import display as disp
from IPython.display import HTML

# Enter reproducibility threshold
reproducibility_thresh = 0.5 # siRNAs where < 1 phenotype is reproducible (i.e.␣
 ↪occurs in > reproducibility_thresh
# of the replicates) will be discarded in the grsa_consistent table.
display = [] # e.g. ANAPC1 is ENSG00000153107

# Create Gene name to Ensembl ID conversion table
directory = os.path.abspath('')
if not display:
    data_file = os.path.join(directory, 'merged.xlsx')
    df = pd.read_excel(data_file)
    translation_table = df.loc[:, ['ENSEMBL ID', 'Gene name']].drop_duplicates()
    display = [val for val in translation_table.loc[:, 'ENSEMBL ID'] if␣
 ↪isinstance(val, str)]

# Create genes_to_rna_frame
ENDPOINT = 'https://www.mitocheck.org/cgi-bin/mtc'
genes_to_rna_dict = {}
for gene in display:
    response = requests.get(ENDPOINT,
                            params={'action': 'get_data',
                        'gene': gene,
                        'data': 'dsRNAs',
                        'format': 'json'})
    response.raise_for_status()
    try:
        json_response = response.json()
    except:
        json_response = literal_eval(response.text)
    rna_list = []
    for rna in json_response:
        rna_id = rna['id']
        rna_catalog_number = rna['catalog_number']
        rna_list.append((rna_id, rna_catalog_number))
    genes_to_rna_dict[gene] = rna_list

genes_to_rna_frame = pd.DataFrame.from_dict(genes_to_rna_dict, orient='index')
genes_to_rna_frame = genes_to_rna_frame.reset_index().melt(id_vars='index').
 ↪sort_values(by=['index'])
genes_to_rna_frame = genes_to_rna_frame.rename({'index': 'Genes', 'value':␣
 ↪'siRNAs'}, axis=1)
genes_to_rna_frame = genes_to_rna_frame.loc[genes_to_rna_frame.loc[:, 'siRNAs'].
 ↪notnull(), :]
genes_to_rna_frame[['siRNA_id', 'siRNA_catalog_number']] = pd.
 ↪DataFrame(genes_to_rna_frame['siRNAs'].\
                                                                            ␣
 ↪tolist(), index=genes_to_rna_frame.index)
genes_to_rna_frame = genes_to_rna_frame.drop(['variable', 'siRNAs'], axis=1)
```

[ ]:
```
# Get KD strength for each siRNA if known
data_file = os.path.join(directory, '2008-09-09955C-SupplementaryTable1.xls')
ko_strength = pd.read_excel(data_file, header=4)
ko_strength.head(20)

# Create rna_attributes_frame showing phenotypes for each siRNA
phenotypes_of_interest = ['metaphase delayed', 'cell death', 'prometaphase␣
 ↪delayed', 'mitosis delayed',
                          'proliferating cells', 'decreased duration of mitotic␣
 ↪prophase',
                          'increased duration of mitotic prophase'] # Todo:␣
 ↪figure out what these phenotypes mean

rna_attributes_frame = pd.DataFrame(columns=phenotypes_of_interest,␣
 ↪index=genes_to_rna_frame['siRNA_id'])
for siRNA_id in genes_to_rna_frame['siRNA_id']:
    response = requests.get(ENDPOINT,
                        params={'action': 'get_data',
                        'dsRNA': siRNA_id,
                        'data': 'phenotypes',
                        'format': 'json'})
    response.raise_for_status()
    try:
        json_response = response.json()
    except:
        json_response = literal_eval(str(response.content))
    for phenotype in json_response:
        term = phenotype['CMPO']['term']
        if term in phenotypes_of_interest:
            pos, total = phenotype['reproducibility'].split('/')
            phenotype['reproducibility']
            rna_attributes_frame.loc[siRNA_id, term] =␣
 ↪phenotype['reproducibility']
```

[ ]:
```
# Create grsa table that shows genes, siRNAs their KD strength and other␣
 ↪attributes and write to excel
grs = genes_to_rna_frame.merge(ko_strength.astype(
        {'Ambion siRNA ID': 'str'}), how='left',␣
 ↪left_on='siRNA_catalog_number', right_on='Ambion siRNA ID').\
        drop(['siRNA_catalog_number', 'Ambion siRNA ID'], axis=1)
grsa = grs.merge(rna_attributes_frame, how='outer', on='siRNA_id')
grsa = grsa.merge(translation_table, how='left', left_on='Genes',␣
 ↪right_on='ENSEMBL ID').\
        set_index('Gene name').drop('ENSEMBL ID', axis=1).sort_values(by=['Gene␣
 ↪name'])

with open('grsa.xlsx', 'wb') as file:
    grsa.to_excel(file)
print('-- Genes, siRNAs, KD strength and pheonotypes (first 50 rows) --')
disp(grsa.drop(['Genes', 'Target ENSEMBL gene(s)'], axis=1).head(50))

# Filter grsa table for working siRNAs and only report phenotypes that are not␣
 ↪inconsistent with other working siRNAs
# Note that only the phenotypes of interest are used for this analysis here.␣
 ↪Would I include all measured phenotypes
# (as would be the cleaner way to do), I would get more working siRNAs and thus␣
 ↪more chances of inconsistencies.
def thresholder(x, reproducibility_thresh):
    if isinstance(x, float):
        if x > reproducibility_thresh:
            return True
        else:
            return False
    else:
        return x
```

```
rna_attributes_thesholded = copy.deepcopy(rna_attributes_frame)
rna_attributes_thesholded.loc[:, 'metaphase delayed':] =␣
 ↪rna_attributes_thesholded.loc[:, 'metaphase delayed':].\
        replace(np.nan, 0).applymap(lambda x: eval(str(x)))
reproducible_rnas = rna_attributes_thesholded.loc[:, 'metaphase delayed':].
 ↪to_numpy()
reproducible_rnas = (reproducible_rnas > reproducibility_thresh).any(axis=1)
rna_attributes_thesholded = rna_attributes_thesholded.loc[reproducible_rnas, :].
 ↪\
        applymap(lambda x: thresholder(x, reproducibility_thresh))

grsa_consistent = grs.merge(rna_attributes_thesholded, how='right',␣
 ↪left_on='siRNA_id', right_index=True)
grsa_consistent = grsa_consistent.groupby('Genes').all().loc[:, 'metaphase␣
 ↪delayed':] # Todo: try replacing all() with sum()
genes_without_pheno = grsa_consistent.loc[:, 'metaphase delayed':].to_numpy()
genes_without_pheno = genes_without_pheno.any(axis=1)
grsa_consistent = grsa_consistent.merge(translation_table, how='left',␣
 ↪left_index=True, right_on='ENSEMBL ID').\
        set_index('Gene name').loc[genes_without_pheno, :].
 ↪sort_values(by=['Gene name'])

with open('grsa_consistent.xlsx', 'wb') as file:
    grsa_consistent.to_excel(file)
print('-- All genes with consistent phenotypes --')
disp(grsa_consistent)

# They used several siRNAs per gene and took videos of transfected and controll␣
 ↪cell populations. They monitored the
# percentage of cells that show a given phenotype (e.g. large nuclei,␣
 ↪chromosomes in metaphase plate) over time. If the
# fraction of transfected cells in a given phenotype at any time was different␣
 ↪(manually define percentage threshold)
# than control, I think.
```

```
-- Genes, siRNAs, KD strength and pheonotypes (first 50 rows) --

              siRNA_id  % mean remaining mRNA  metaphase delayed cell death  \
Gene name
ANAPC1     DSR00061831                    NaN               NaN        NaN
ANAPC1     DSR00042680                    NaN               NaN        NaN
ANAPC1     DSR00047613                    NaN               NaN        3/3
ANAPC1     DSR00061527                    NaN               NaN        NaN
ANAPC1     DSR00018342                    NaN               NaN        NaN
ANAPC1     DSR00018343                    NaN               NaN        NaN
ANAPC1     DSR00066570                    NaN               NaN        NaN
ANAPC1     DSR00066678                    NaN               NaN        NaN
```

```
ANAPC1     DSR00046663                     NaN              NaN        NaN
ANAPC1     DSR00046001                     NaN              NaN        NaN
ANAPC10    DSR00034296                     NaN              NaN        NaN
ANAPC10    DSR00034284                     NaN              NaN        1/3
ANAPC10    DSR00055476                     NaN              NaN        NaN
ANAPC10    DSR00055477                     NaN              NaN        NaN
ANAPC11    DSR00055812                     NaN              NaN        NaN
ANAPC11    DSR00048727                     NaN              NaN        NaN
ANAPC11    DSR00046621                     NaN              3/3        NaN
ANAPC11    DSR00042638                     NaN              2/3        NaN
ANAPC11    DSR00048716                     NaN              1/5        NaN
ANAPC11    DSR00055811                     NaN              NaN        NaN
ANAPC11    DSR00047571                     NaN              2/4        NaN
ANAPC2     DSR00055746                     NaN              NaN        NaN
ANAPC2     DSR00055745                     NaN              NaN        NaN
ANAPC2     DSR00031855                     NaN              3/3        NaN
ANAPC2     DSR00031867                     NaN              2/3        NaN
ANAPC4     DSR00025164                     NaN              3/4        1/4
ANAPC4     DSR00055752                     NaN              NaN        NaN
ANAPC4     DSR00055751                     NaN              NaN        NaN
ANAPC4     DSR00025165                     NaN              2/3        NaN
ANAPC5     DSR00055806                     NaN              NaN        NaN
ANAPC5     DSR00016762                     NaN              2/3        1/3
ANAPC5     DSR00060030                     NaN              NaN        NaN
ANAPC5     DSR00016761                     NaN              3/3        NaN
ANAPC5     DSR00055805                     NaN              NaN        NaN
ANAPC7     DSR00060768                     NaN              NaN        NaN
ANAPC7     DSR00061118                     NaN              NaN        NaN
ANAPC7     DSR00025218                     NaN              NaN        NaN
ANAPC7     DSR00025219                     NaN              NaN        NaN
ARPP19     DSR00049319                     NaN              NaN        NaN
ARPP19     DSR00040362                     NaN              NaN        NaN
ARPP19     DSR00049800                     NaN              NaN        NaN
ARPP19     DSR00049797                     NaN              NaN        3/3
ARPP19     DSR00049791                     NaN              NaN        NaN
ARPP19     DSR00017651                     NaN              NaN        NaN
ARPP19     DSR00017652                     NaN              NaN        NaN
CCNA1      DSR00065257                     NaN              NaN        NaN
CCNA1      DSR00065702                     NaN              NaN        NaN
CCNA1      DSR00013189                     NaN              NaN        NaN
CCNA1      DSR00013190                     NaN              NaN        NaN
CCNA1      DSR00013188                     NaN              NaN        NaN

         prometaphase delayed mitosis delayed proliferating cells  \
Gene name
ANAPC1                      NaN             NaN                NaN
ANAPC1                      NaN             NaN                NaN
ANAPC1                      NaN             NaN                NaN
```

ANAPC1	NaN	NaN	NaN
ANAPC1	NaN	NaN	1/4
ANAPC1	NaN	NaN	NaN
ANAPC1	NaN	NaN	NaN
ANAPC1	NaN	NaN	NaN
ANAPC1	NaN	NaN	NaN
ANAPC1	NaN	NaN	NaN
ANAPC10	NaN	NaN	1/3
ANAPC10	NaN	NaN	NaN
ANAPC10	NaN	NaN	NaN
ANAPC10	NaN	NaN	NaN
ANAPC11	NaN	2/4	NaN
ANAPC11	NaN	NaN	NaN
ANAPC11	NaN	NaN	1/3
ANAPC11	NaN	NaN	NaN
ANAPC11	NaN	NaN	1/5
ANAPC11	NaN	NaN	NaN
ANAPC11	NaN	1/4	NaN
ANAPC2	NaN	3/4	NaN
ANAPC2	NaN	2/4	NaN
ANAPC2	NaN	NaN	NaN
ANAPC2	NaN	NaN	NaN
ANAPC4	NaN	2/4	NaN
ANAPC4	NaN	2/4	NaN
ANAPC4	NaN	NaN	NaN
ANAPC4	NaN	NaN	NaN
ANAPC5	NaN	3/4	NaN
ANAPC5	NaN	1/3	NaN
ANAPC5	NaN	NaN	NaN
ANAPC5	NaN	NaN	NaN
ANAPC5	NaN	NaN	NaN
ANAPC7	NaN	NaN	NaN
ANAPC7	NaN	NaN	NaN
ANAPC7	NaN	NaN	NaN
ANAPC7	NaN	NaN	1/4
ARPP19	NaN	NaN	NaN
ARPP19	NaN	NaN	NaN
ARPP19	NaN	1/3	NaN
ARPP19	3/3	3/3	NaN
ARPP19	NaN	NaN	NaN
ARPP19	NaN	NaN	NaN
ARPP19	NaN	NaN	NaN
CCNA1	NaN	NaN	NaN
CCNA1	NaN	NaN	NaN
CCNA1	NaN	NaN	NaN
CCNA1	NaN	NaN	NaN
CCNA1	NaN	NaN	NaN

```
          decreased duration of mitotic prophase  \
Gene name
ANAPC1                                       NaN
ANAPC1                                       NaN
ANAPC1                                       NaN
ANAPC1                                       NaN
ANAPC1                                       NaN
ANAPC1                                       NaN
ANAPC1                                       NaN
ANAPC1                                       NaN
ANAPC1                                       NaN
ANAPC1                                       NaN
ANAPC10                                      NaN
ANAPC10                                      NaN
ANAPC10                                      NaN
ANAPC10                                      NaN
ANAPC11                                      NaN
ANAPC11                                      NaN
ANAPC11                                      NaN
ANAPC11                                      NaN
ANAPC11                                      NaN
ANAPC11                                      NaN
ANAPC11                                      NaN
ANAPC2                                       NaN
ANAPC2                                       NaN
ANAPC2                                       NaN
ANAPC2                                       NaN
ANAPC4                                       NaN
ANAPC4                                       NaN
ANAPC4                                       NaN
ANAPC4                                       NaN
ANAPC5                                       NaN
ANAPC5                                       NaN
ANAPC5                                       NaN
ANAPC5                                       NaN
ANAPC5                                       NaN
ANAPC7                                       NaN
ANAPC7                                       NaN
ANAPC7                                       NaN
ANAPC7                                       NaN
ARPP19                                       NaN
ARPP19                                       NaN
ARPP19                                       NaN
ARPP19                                       NaN
ARPP19                                       NaN
ARPP19                                       NaN
ARPP19                                       NaN
CCNA1                                        NaN
```

```
CCNA1                                     NaN
CCNA1                                     NaN
CCNA1                                     NaN
CCNA1                                     NaN

          increased duration of mitotic prophase
Gene name
ANAPC1                                    NaN
ANAPC1                                    NaN
ANAPC1                                    NaN
ANAPC1                                    NaN
ANAPC1                                    NaN
ANAPC1                                    NaN
ANAPC1                                    NaN
ANAPC1                                    NaN
ANAPC1                                    NaN
ANAPC1                                    NaN
ANAPC10                                   NaN
ANAPC10                                   NaN
ANAPC10                                   NaN
ANAPC10                                   NaN
ANAPC11                                   NaN
ANAPC11                                   NaN
ANAPC11                                   NaN
ANAPC11                                   NaN
ANAPC11                                   NaN
ANAPC11                                   NaN
ANAPC11                                   NaN
ANAPC2                                    NaN
ANAPC2                                    NaN
ANAPC2                                    NaN
ANAPC2                                    NaN
ANAPC4                                    NaN
ANAPC4                                    NaN
ANAPC4                                    NaN
ANAPC4                                    NaN
ANAPC5                                    NaN
ANAPC5                                    NaN
ANAPC5                                    NaN
ANAPC5                                    NaN
ANAPC5                                    NaN
ANAPC7                                    NaN
ANAPC7                                    NaN
ANAPC7                                    NaN
ANAPC7                                    NaN
ARPP19                                    NaN
ARPP19                                    NaN
ARPP19                                    NaN
```

```
ARPP19                                          NaN
ARPP19                                          NaN
ARPP19                                          NaN
ARPP19                                          NaN
CCNA1                                           NaN
CCNA1                                           NaN
CCNA1                                           NaN
CCNA1                                           NaN
CCNA1                                           NaN


-- All genes with consistent phenotypes --

           metaphase delayed  cell death  prometaphase delayed  \
Gene name
ANAPC1                 False        True                 False
ANAPC11                 True       False                 False
ANAPC4                  True       False                 False
ARPP19                 False        True                  True
CDC16                   True       False                 False
CDC23                   True       False                 False
E2F2                    True       False                 False
WEE1                   False       False                 False

           mitosis delayed  proliferating cells  \
Gene name
ANAPC1               False                False
ANAPC11              False                False
ANAPC4               False                False
ARPP19                True                False
CDC16                False                False
CDC23                False                False
E2F2                 False                False
WEE1                  True                False

           decreased duration of mitotic prophase  \
Gene name
ANAPC1                                      False
ANAPC11                                     False
ANAPC4                                      False
ARPP19                                      False
CDC16                                       False
CDC23                                       False
E2F2                                        False
WEE1                                        False

           increased duration of mitotic prophase       ENSEMBL ID
Gene name
```

```
ANAPC1                                        False   ENSG00000153107
ANAPC11                                       False   ENSG00000141552
ANAPC4                                        False   ENSG00000053900
ARPP19                                        False   ENSG00000128989
CDC16                                         False   ENSG00000130177
CDC23                                         False   ENSG00000094880
E2F2                                          False   ENSG00000007968
WEE1                                          False   ENSG00000166483
```

Conclusion: * Most siRNAs do either not produce reproducible phenotypes or produce reproducible phenotypes that are inconsistent with other siRNAs targetting the same gene. * This data set may be more useful for assessing siRNAs specificity than for parameter fitting. * The phenotypic patterns do not seem to be well defined (e.g. what is the difference between 'increased duration of mitotic prophase' and 'metaphase delayed'. * The necessity of APC is confirmd * Interestingly E2F2 seems to effect metaphase * Unexpectedly WEE1 kinase knockdown appears to cause 'delayed mitosis'. * Better not trust this data set.

10 Rule based RP model description

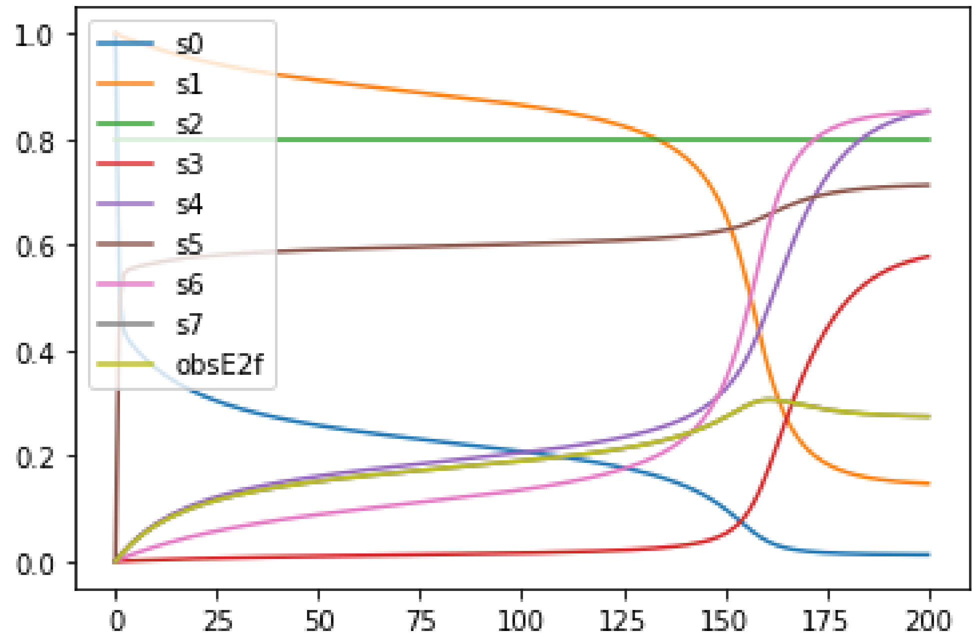
[ ]:
```python
from pysb import *
from pysb.integrate import Solver
import numpy as np
import pysb
import os
import copy

# pysb.pathfinder.set_path('BioNetGen', '/usr/local/sbin')

Model('RP')

# Define Proteins
Monomer('Rb', ['E2f', 'p'], {'p': ['u', 'p']})
Monomer('E2f', ['Rb', 'Px'])
Monomer('Px', ['E2f'])
Monomer('Ce')
Monomer('Cd')

# Define kinetc parameters
Parameter('kDpRb', 1)
Parameter('kPhRb', 1.5)
Parameter('kSyCe1', 0.01)
Parameter('kSyCe2', 0.1)
Parameter('kDeCe', 0.11)
Parameter('kSyE2f1', 0.01)
Parameter('kSyE2f2', 0.1)
Parameter('kDeE2f', 0.11)
Parameter('kDiE2fRb', 0.1)
Parameter('kAsE2fRb', 100)
Parameter('kAsEPx', 50)
Parameter('kDiEPx', 5)

# Define initial conditions
Parameter('Rb_0', 1)
Initial(Rb(E2f=None, p='u'), Rb_0)
Parameter('Px_0', 1)
Initial(Px(E2f=None), Px_0)
Parameter('Cd_0', 0.8)
Initial(Cd(), Cd_0)

# Define reaction rules
Rule('SyE2f1', None >> E2f(Rb=None, Px=None), kSyE2f1)
Rule('AsE2f_Px', E2f(Rb=None, Px=None) + Px(E2f=None) >>
        E2f(Rb=None, Px=1) % Px(E2f=1) +
        E2f(Rb=None, Px=None), kAsEPx)
Rule('DiE2f_Px', E2f(Rb=None, Px=1) % Px(E2f=1) >>
        Px(E2f=None), kDiEPx)
Rule('SyE2f2', E2f(Rb=None, Px=1) % Px(E2f=1) >>
                                E2f(Rb=None, Px=1) % Px(E2f=1) +
                                E2f(Rb=None, Px=None), kSyE2f2)
Rule('DeE2f', E2f(Rb=WILD, Px=None) >> None, kDeE2f, delete_molecules=True) #␣
 ↪Todo: check if Rb, E2f_gene and
        # Ce_gene are set free or also degraded.
        # If delete_molecules is set True, only the E2f is degraded.

Rule('SyCe1', None >> Ce(), kSyE2f1)
Rule('SyCe2', E2f(Rb=None, Px=1) % Px(E2f=1) >>
                                E2f(Rb=None, Px=1) % Px(E2f=1) + Ce(), kSyCe2)
Rule('DeCe', Ce() >> None, kDeCe)

Rule('AsE2fRb', E2f(Rb=None, Px=None) + Rb(E2f=None, p='u') |
        E2f(Rb=1, Px=None) % Rb(E2f=1, p='u'), kAsE2fRb, kDiE2fRb)

Rule('PhRbCd1', Rb(E2f=None, p='u') + Cd() >> Rb(E2f=None, p='p') + Cd(),␣
 ↪kPhRb, delete_molecules=True) # Todo: Write this as a macro for both, Ce and␣
 ↪Cd
Rule('PhRbCe1', Rb(E2f=None, p='u') + Ce() >> Rb(E2f=None, p='p') + Ce(),␣
 ↪kPhRb, delete_molecules=True) # Todo: Check if the last two lines give the␣
 ↪correct ODEs
Rule('PhRbCd2', E2f(Rb=1, Px=None) % Rb(E2f=1, p='u') + Cd() >> Rb(E2f=None,␣
 ↪p='p') + E2f(Rb=None, Px=None) + Cd(), kPhRb, delete_molecules=True) # Todo:␣
 ↪Write this as a macro for both, Ce and Cd
```

```
Rule('PhRbCe2', E2f(Rb=1, Px=None) % Rb(E2f=1, p='u') + Ce() >> Rb(E2f=None,␣
↪p='p') + E2f(Rb=None, Px=None) + Ce(), kPhRb, delete_molecules=True) # Todo:␣
↪Check if the last two lines give the correct ODEs

Rule('DpRb', Rb(E2f=None, p='p') >> Rb(E2f=None, p='u'), kDpRb)

# Define observables
Observable('obsE2f', E2f(Rb=ANY, Px=None))

# Run simulation
t = np.linspace(0, 200, 201)
solver = Solver(RP, t)
solver.run()

# Export model
from pysb.export import export
python_model = export(RP, 'python')
output_dir = os.path.abspath('')
with open(os.path.join(output_dir, 'python_model.py'), 'w') as file:
        file.writelines(python_model)


with open(os.path.join(output_dir,'test_ODEs.txt'), 'w') as file:
        i = 0
        for species, ode in zip(RP.species, RP.odes):
            file.writelines('s{} - {}: {}\n'.format(i, species, ode))
            i+=1

# Plot resulst
from pysb.simulator import ScipyOdeSimulator
import matplotlib.pyplot as plt
# import pylab as pl
t = np.linspace(0, 200, 201)
x = ScipyOdeSimulator(RP).run(tspan=t)
plt.plot(t, x.dataframe)
labels = copy.copy(x.dataframe.columns.values)
for i in range(len(labels)):
        labels[i] = labels[i].lstrip("_")
plt.legend(labels, loc='upper left')
```

```
2019-08-22 15:17:18.114 - pysb.simulator.scipyode - WARNING - [RP] This system
of ODEs will be evaluated in pure Python. This may be slow for large models. We
recommend installing a package for compiling the ODEs to C code: 'weave'
(recommended for Python 2) or 'cython' (recommended for Python 3). This warning
can be suppressed by specifying compiler='python'.
2019-08-22 15:17:18.188 - pysb.simulator.scipyode - WARNING - [RP] This system
of ODEs will be evaluated in pure Python. This may be slow for large models. We
```

```
recommend installing a package for compiling the ODEs to C code: 'weave'
(recommended for Python 2) or 'cython' (recommended for Python 3). This warning
can be suppressed by specifying compiler='python'.
```

[ ]:
```
<matplotlib.legend.Legend at 0x21999537d68>
```


s0
s1
s2
s3
s4
s5
s6
s7
obsE2f