Validation tests

Background

I've been developing a plate-based analog of traditional P450 titrations. I think I'm mostly there but have a few last questions to answer and also want to put this experiment to bed with a big test to end all tests.

Loose ends:

- **Which plate type is best??** I have a lot of diffent plate types that I was sent by various suppliers. Ideally, I can get this thing working well in the cheapest plate type.
- **How sensitive can I make it??** At the moment, I'm having trouble picking up a signal from palmitic acid, which might be to do with plate type, or maybe it can be helped by tinkering with the buffer. We'll see here.

I'm aiming to do this experiment with a fluid handling robot to save the experiment from the walking accident waiting to happen that is me. I'll do this with wild type P450 BM3 heme domain. For my substrates I'm using the usual suspects:

- **Arachadionic acid** - Natural substrate, This has been giving me a good signal so far and doesn't normally have dramas.
- **Lauric acid** - but gives a weaker signal than Arachadionic acid
- **Plamitic acid** - this one is a tricky one. It gives a substrate shift in titration experiments, but a weak one, which I haven't been abe to pick up in my plate assays so far. It would be good to have the assay sensitive enough to detect binders of substrates like this.
- **4-Phenylimidazole** - Inhibitor, should give a different type of shift, which I'll have to write something to accomodate for this. Also give a fairly weak signal.

The Plan

Here are the plates I have and want to test:

Make	Plate type	Product Number	Qty
Thermo	Nunclon Delta Surface	?	20
Brand	?	781620	2
Brand	Lipograde	781860	4
Nunc	Maxisorp	464718	1
Corning	Cellbind	3770BC	17
Corning	Cellbind	3640	14

Buffers

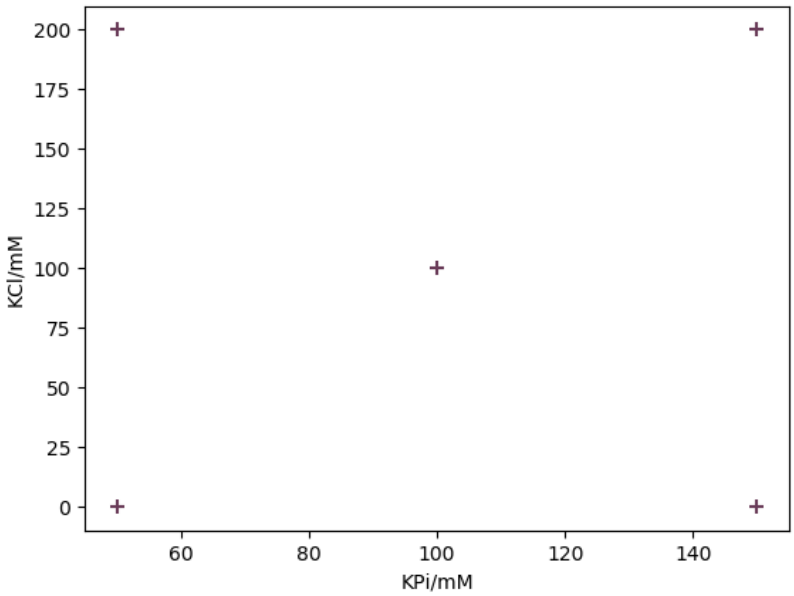
I also want to test some buffer conditions, but not too many. The current buffer I'm working with is 100 mM Kpi @ pH7, which apparently is where BM3 is happy on a normal day. Things are a bit different in a plate because of the contact area. I'll dig up some literature or something but Hazel recommended me some ranges to work in:

Buffer component	Low/mM	Mid/mM	High/mM

Buffer component	Low/mM	Mid/mM	High/mM
KPi	50	100	150
KCl	0	100	200

I want to do a combinatoria design with these ones, and I also wanted the midpoint to check for nonlinearities because I was interested. I'll do it like this:

```
Out[12]:
   KPi  KCl
0   50    0
1   50  200
2  100  100
3  150    0
4  150  200
```



Here's another way of looking at it:

Layouts

- **4 substrates** times **5 Buffer conditons** plus a column for zeroing the whole thing. That's 21/24, so I'll use the last 3 columns for repeats of the center point.

```
In [2]: pd.read_csv('PlateLayout1.csv')
Out[2]:
    Column          Substrate  Kpi /MM  Kcl/mM
0        1               DMSO      100       0
1        2  Arachadionic Acid       50       0
2        3  Arachadionic Acid       50     200
3        4  Arachadionic Acid      100     100
4        5  Arachadionic Acid      100     100
5        6  Arachadionic Acid      150       0
6        7  Arachadionic Acid      150     200
7        8        Lauric Acid       50       0
8        9        Lauric Acid       50     200
9       10        Lauric Acid      100     100
10      11        Lauric Acid      100     100
11      12        Lauric Acid      150       0
12      13        Lauric Acid      150     200
13      14      Palmitic Acid       50       0
14      15      Palmitic Acid       50     200
15      16      Palmitic Acid      100     100
16      17      Palmitic Acid      100     100
17      18      Palmitic Acid      150       0
18      19      Palmitic Acid      150     200
19      20  4-Phenylimidazole       50       0
20      21  4-Phenylimidazole       50     200
21      22  4-Phenylimidazole      100     100
```

```
22      23  4-Phenylimidazole      150       0
23      24  4-Phenylimidazole      150     200
```

Each compound has each of 5 buffer conditions per plate plus a repeat of the 100:100 center point, except for 4-Phenylimidazole because I ran out of space. 🤷

I think I'll make a single master plate by hand and pipette that into the 6 or so plates by robot. I'll try to get my hands on a few more 384 well plates from the building, I'd really like to find the cheapest viable option.

Master plate calculations

```
In [8]: nplates = 6
In [9]: vol_of_each_compound_conc = nplates *6*2*25 # 6 reps/plate for everything except 4phenylimidazole, 2 w
In [4]: masterplate = np.zeros((8,12)) #96 wells
In [12]: masterplate[:,:5]+=1*(vol_of_each_compound_conc+100) #plus a bit of dead volume
In [13]: masterplate
Out[13]:
array([[1900., 1900., 1900., 1900., 1900.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.],
       [1900., 1900., 1900., 1900., 1900.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.],
       [1900., 1900., 1900., 1900., 1900.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.],
       [1900., 1900., 1900., 1900., 1900.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.],
       [1900., 1900., 1900., 1900., 1900.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.],
       [1900., 1900., 1900., 1900., 1900.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.],
       [1900., 1900., 1900., 1900., 1900.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.],
       [1900., 1900., 1900., 1900., 1900.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.]])
# I don't think this will work, the deepest plate I know about does 2000 μM, but the wells will have to accomo
In [14]: masterplate = np.zeros((8,12))
In [15]: masterplate[:,:9]+=1*((vol_of_each_compound_conc)/2+100)
In [16]: masterplate
Out[16]:
array([[1000., 1000., 1000., 1000., 1000., 1000., 1000., 1000., 1000.,
           0.,    0.,    0.],
       [1000., 1000., 1000., 1000., 1000., 1000., 1000., 1000., 1000.,
           0.,    0.,    0.],
       [1000., 1000., 1000., 1000., 1000., 1000., 1000., 1000., 1000.,
           0.,    0.,    0.],
       [1000., 1000., 1000., 1000., 1000., 1000., 1000., 1000., 1000.,
           0.,    0.,    0.],
       [1000., 1000., 1000., 1000., 1000., 1000., 1000., 1000., 1000.,
           0.,    0.,    0.],
       [1000., 1000., 1000., 1000., 1000., 1000., 1000., 1000., 1000.,
           0.,    0.,    0.],
       [1000., 1000., 1000., 1000., 1000., 1000., 1000., 1000., 1000.,
           0.,    0.,    0.],
       [1000., 1000., 1000., 1000., 1000., 1000., 1000., 1000., 1000.,
           0.,    0.,    0.]])
# Looks fine, by the end, the plate should be loke this:
In [17]: masterplate[-1,:9]+=1000 # last row ends up double volume
In [20]: masterplate_df = pd.DataFrame(masterplate,columns = ['DMSO','Arachadionic      Acid','Arachadionic Ac
    ...: 'Lauric Acid','Lauric Acid','Palmitic Acid','Palmitic Acid','4-Phenylimidazole','4-Phenylimidazole','
In [23]: masterplate_df.to_csv('masterplate_df.csv')
```

masterplate_df.csv

Cool, that should do it.

Buffer pipetting calculations

I might as well do this now

```
In [45]: bufferconcs = pd.read_csv('PlateLayout1.csv')

In [46]: bufferconcs=bufferconcs.groupby(['Kpi /MM','Kcl/mM']).size().reset_index().rename(columns={0:'count'}
...: print(bufferconcs)
Out[46]:
   Kpi /MM  Kcl/mM  count
0       50       0      4
1       50     200      4
2      100       0      1
3      100     100      7
4      150       0      4
5      150     200      4


In [50]: bufferconcs['count']*=(6+25+8) # 6 reps * 8 wells per col with protein in * 25 ul\
    ...: print(bufferconcs) # I cheated and renamed the col by hand
    ...:
   Kpi /MM  Kcl/mM  Vol Prot/µl
0       50       0   6084
1       50     200   6084
2      100       0   1521
3      100     100  10647
4      150       0   6084
5      150     200   6084

In [55]: bufferconcs.columns=['Kpi /MM',  'Kcl/mM',  'Volume/ul']
In [56]: bufferconcs['Volume/ul']+=500 # seem like a reasonable safety margin?

In [57]: bufferconcs
Out[57]:
   Kpi /MM  Kcl/mM  Volume/ul
0       50       0       6584
1       50     200       6584
2      100       0       2021
3      100     100      11147
4      150       0       6584
5      150     200       6584
```