Selwyn Test

February 1, 2022

Written by G. Gygli.
Contact gudrun.gygli@kit.edu in case of questions.
Made available without any warranty under a CC-By license. This script uses Python 3

1 A Selwyn Test

This script aims to help you to understand a Selwyn test for experiments describing a single-substrate, single-enzyme catalyzed reactions.

It allows you to simulate time course data that:

- passes the Selwyn test
- or
- fails the Selwyn test

To be able to perform a Selwyn test with this notebook, we need to

1. Import required packages
2. Define some parameters
3. Simulate the data
4. Perform the Selwyn test

If you perform a Selwyn test for your real experimental data, you can read in data instead of simulating it in step 3, and will enter your experimental conditions in step 2.

1.1 1. Import required packages

As always, first we need to import some packages and define some parameters:

```python
[1]: # Here, we import all the python packages we need to run this script. # in the unlikely case they are not installed on your computer, this might help:
   ➔ # https://jakevdp.github.io/blog/2017/12/05/
   ➔installing-python-packages-from-jupyter/
   ➔imported 04.10.21
   ➔import pandas as pd
   ➔from scipy.special import lambertw
   ➔from scipy.optimize import curve_fit
```
import numpy as np
import scipy
import warnings
from scipy.optimize import OptimizeWarning
import matplotlib.pyplot as plt

1.2 2. Define some parameters

1.2.1 2.1 Enzyme properties

[2]:
Km = 1000.0 # initial value 1000
Vmax = 2.0 # initial value 2

1.2.2 2.2 Experimental conditions of our simulated experiment

[3]:
E = [10, 20, 30, 40, 50, 60] # initial values 10, 20, 30, 40, 50, 60, ENZYME
concentrations at the beginning of the experiment
S = 1000.0 # initial value 1000, substrate concentration at the beginning of the experiment
# S should be chosen to ensure the enzyme reaction runs at Vmax in these experiments.
# the S chosen here as initial value is very excessive

1.2.3 2.3 Define things for the notebook

[4]:
# we will simulate data for 6 different enzyme concentrations
# when modifying things here, keep in mind that the length of df and E have to match!
# one could also use a dictionary here ;)
# because we are going to simulate data that passes and data that fails the Selwyn test, we need two dataframes

df_selwyn_pass = pd.DataFrame(columns=['time(s)','c_E1','c_E2','c_E3','c_E4','c_E5','c_E6'])
df_selwyn_fail = pd.DataFrame(columns=['time(s)','c_E1','c_E2','c_E3','c_E4','c_E5','c_E6'])

1.3 3. Simulate the data

1.3.1 3.1 Generate time course data that passes the Selwyn test

Here, you can simulate time course data for enzyme time course data that passes the Selwyn test, meaning that there is NO INACTIVATION of the enzyme in this experiment.

[5]:
#define the function to simulate data that passes the Selwyn Test
def Selwyn(t,E,S):

y = \frac{E(S/K_m) - (V_{max}/K_m) \cdot t}{S - np.exp((-V_{max} \cdot t)/E) \cdot (1 - (S/(V_{max} \cdot E)))}

return y

def selwyn_pass["time (s)"] = list(range(0, 50))  # initial values: 0,50 -> this will create a list with values ranging from 0 to 50
# now, we use the function we defined above to simulate data using time and the given substrate concentration
# for each of the different enzyme concentrations
df_selwyn_pass["c_E1"] = df_selwyn_pass["time (s)"].apply(lambda x:\n    Selwyn(x+1,E[0],S))
df_selwyn_pass["c_E2"] = df_selwyn_pass["time (s)"].apply(lambda x:\n    Selwyn(x+1,E[1],S))
df_selwyn_pass["c_E3"] = df_selwyn_pass["time (s)"].apply(lambda x:\n    Selwyn(x+1,E[2],S))
df_selwyn_pass["c_E4"] = df_selwyn_pass["time (s)"].apply(lambda x:\n    Selwyn(x+1,E[3],S))
df_selwyn_pass["c_E5"] = df_selwyn_pass["time (s)"].apply(lambda x:\n    Selwyn(x+1,E[4],S))
df_selwyn_pass["c_E6"] = df_selwyn_pass["time (s)"].apply(lambda x:\n    Selwyn(x+1,E[5],S))

# here, we plot the data we simulated:
fig = plt.figure(figsize=[5,3], dpi=500)
plt.scatter(x = df_selwyn_pass["time (s)"], y = df_selwyn_pass["c_E1"], label = $E_0$)  # E[0]
plt.scatter(x = df_selwyn_pass["time (s)"], y = df_selwyn_pass["c_E2"], label = $E_1$)  # E[1]
plt.scatter(x = df_selwyn_pass["time (s)"], y = df_selwyn_pass["c_E3"], label = $E_2$)  # E[2]
plt.scatter(x = df_selwyn_pass["time (s)"], y = df_selwyn_pass["c_E4"], label = $E_3$)  # E[3]
plt.scatter(x = df_selwyn_pass["time (s)"], y = df_selwyn_pass["c_E5"], label = $E_4$)  # E[4]
plt.scatter(x = df_selwyn_pass["time (s)"], y = df_selwyn_pass["c_E6"], label = $E_5$)  # E[5]

# and add some meaningful labels
plt.title("Simulated time course data that will pass the Selwyn test", fontsize=14)
plt.legend(loc='lower right', title="$E_0$ (\u03BC M)")
plt.ylabel('[P] (\u03BC M)')
plt.xlabel('time (s)')

[5]: Text(0.5, 0, 'time (s)')
1.3.2 3.2 Generate time course data that fails the Selwyn test

Here, you can simulate time course data for enzyme time course data that fails the Selwyn test, meaning that there is **INACTIVATION** of the enzyme in this experiment.

For this, we need to use slightly different function to simulate our data than before.

```python
[6]: def Selwyn_fail(t,E,S,inactivation):
    # y = E*(S/Km)-(Vmax/Km)*t
    # accounting for enzyme inactivation (which will increase with time) during the experiment
    E=E-(t*(E/inactivation))
    y = S - np.exp((-Vmax*t*E)/Km)*(1 - (S/(Vmax*E+t))))
    return y

inactivation=80 # initial value: 80
# if this value is chosen too small, the simulated enzyme inactivation will become HUGE
# and it will lead to E=0 or (unrealistic) negative enzyme concentration
# in practice, this means a division by 0 and the script will crash

df_selwyn_fail["time (s)"]=list(range(0, 50))
df_selwyn_fail["c_E1"] = df_selwyn_fail["time (s)"].apply(lambda x:
    Selwyn_fail(x+1,E[0],S,inactivation))
df_selwyn_fail["c_E2"] = df_selwyn_fail["time (s)"].apply(lambda x:
    Selwyn_fail(x+1,E[1],S,inactivation))
```
df_selwyn_fail["c_E3"] = df_selwyn_fail["time (s)"].apply(lambda x:
    Selwyn_fail(x+1,E[2],S,inactivation))
df_selwyn_fail["c_E4"] = df_selwyn_fail["time (s)"].apply(lambda x:
    Selwyn_fail(x+1,E[3],S,inactivation))
df_selwyn_fail["c_E5"] = df_selwyn_fail["time (s)"].apply(lambda x:
    Selwyn_fail(x+1,E[4],S,inactivation))
df_selwyn_fail["c_E6"] = df_selwyn_fail["time (s)"].apply(lambda x:
    Selwyn_fail(x+1,E[5],S,inactivation))

fig = plt.figure(figsize=[5,3], dpi=500)
plt.scatter(x = df_selwyn_fail["time (s)"], y = df_selwyn_fail["c_E1"], label =
    E[0])
plt.scatter(x = df_selwyn_fail["time (s)"], y = df_selwyn_fail["c_E2"], label =
    E[1])
plt.scatter(x = df_selwyn_fail["time (s)"], y = df_selwyn_fail["c_E3"], label =
    E[2])
plt.scatter(x = df_selwyn_fail["time (s)"], y = df_selwyn_fail["c_E4"], label =
    E[3])
plt.scatter(x = df_selwyn_fail["time (s)"], y = df_selwyn_fail["c_E5"], label =
    E[4])
plt.scatter(x = df_selwyn_fail["time (s)"], y = df_selwyn_fail["c_E6"], label =
    E[5])

plt.title("Simulated time course experiments that will fail the Selwyn test",
    fontsize=14)
plt.legend(loc='lower right', title="E_0 (\mu M)")
plt.ylabel('[P] (\mu M)')
plt.xlabel('time (s)')

[6]: Text(0.5, 0, 'time (s)')
1.4 4. Perform the Selwyn test

Now we perform a Selwyn test with the simulated data:

We check whether the time course data are dependent on the enzyme concentration by multiplying time with the respective enzyme concentration used to record the time course $(E[N]*x)$. 

1.4.1 4.1 Selwyn test is passed

If there is no dependence on enzyme concentration, the curves overlap completely.

$\Rightarrow$ the Selwyn test is passed and we know now that the enzyme is not inactivated in the experiment.

```python
fig = plt.figure(figsize=[5,3], dpi=500)
plt.scatter(x = df_selwyn_pass["time (s)"].apply(lambda x: E[0]*x), y =)
plt.scatter(x = df_selwyn_pass["time (s)"].apply(lambda x: E[1]*x), y =)
plt.scatter(x = df_selwyn_pass["c_E1"], label = E[0])
plt.scatter(x = df_selwyn_pass["time (s)"].apply(lambda x: E[2]*x), y =)
plt.scatter(x = df_selwyn_pass["c_E2"], label = E[1])
plt.scatter(x = df_selwyn_pass["time (s)"].apply(lambda x: E[3]*x), y =)
plt.scatter(x = df_selwyn_pass["c_E3"], label = E[2])
plt.scatter(x = df_selwyn_pass["time (s)"].apply(lambda x: E[4]*x), y =)
plt.scatter(x = df_selwyn_pass["c_E4"], label = E[3])
plt.scatter(x = df_selwyn_pass["time (s)"].apply(lambda x: E[5]*x), y =)
plt.scatter(x = df_selwyn_pass["c_E5"], label = E[4])
plt.scatter(x = df_selwyn_pass["time (s)"].apply(lambda x: E[6]*x), y =)
plt.scatter(x = df_selwyn_pass["c_E6"], label = E[5])

plt.title("A PASSED Selwyn test", fontsize=14)
plt.legend(loc='lower right', title="$E_0$ (\textmu M)"
plt.ylabel('[P] (\textmu M)
plt.xlabel('$E_0*t$ (\textmu M*s)')
```

[7]: Text(0.5, 0, '$E_0*t$ (M*s)')
1.4.2 4.2 Selwyn test fails:

We check whether the time course data are dependent on the enzyme concentration by multiplying time with the respective enzyme concentration used to record the time course \((E[N]*x)\).

If there is a dependence on enzyme concentration, the curves do not overlap completely.

-> the Selwyn test is failed and we know now that the enzyme is inactivated in the experiment

8: fig = plt.figure(figsize=[5,3], dpi=500)

plt.scatter(x = df_selwyn_fail["time (s)"].apply(lambda x: E[0]*x), y =)
    df_selwyn_fail["c_E1"], label = E[0])
plt.scatter(x = df_selwyn_fail["time (s)"].apply(lambda x: E[1]*x), y =)
    df_selwyn_fail["c_E2"], label = E[1])
plt.scatter(x = df_selwyn_fail["time (s)"].apply(lambda x: E[2]*x), y =)
    df_selwyn_fail["c_E3"], label = E[2])
plt.scatter(x = df_selwyn_fail["time (s)"].apply(lambda x: E[3]*x), y =)
    df_selwyn_fail["c_E4"], label = E[3])
plt.scatter(x = df_selwyn_fail["time (s)"].apply(lambda x: E[4]*x), y =)
    df_selwyn_fail["c_E5"], label = E[4])
plt.scatter(x = df_selwyn_fail["time (s)"].apply(lambda x: E[5]*x), y =)
    df_selwyn_fail["c_E6"], label = E[5])

plt.title("A FAILED Selwyn test", fontsize=14)
plt.legend(loc='lower right', title="$E_0$ (\u03BC M)")
plt.ylabel(r'$[P]$ ($\mu$M)')
plt.xlabel(r'$[E_0]*t$ ($\mu$M*s)')

[8]: Text(0.5, 0, r'$[E_0]*t$ ($\mu$M*s)')