PROYECTO FINAL DE GENÓMICA COMPUTACIONAL

Parametrización y Automatización de Análisis de Datos de Cinética Enzimática

Mateo Calderón Vargas

Introducción

La catálisis enzimática, es un fenómeno que suele producir un aumento muy grande en la rapidez de reacción, del orden de 10^6 o 10^8 (Chang, 2008). Durante los experimentos para el estudio de dichas biomacromoléculas, se requiere la obtención de un gran número de datos con el fin de que el estudio tenga la mayor precisión posible. No obstante, aunque el tener más datos represente una ventana más amplia al fenómeno que estamos estudiando, también implica un riesgo mayor al momento de manipular mayor cantidad de datos, en especial en estudios de catálisis enzimática, donde es necesario realizar operaciones con los datos recolectados durante el experimento.

El estudio de inhibidores enzimáticos, es un blanco importante en la investigación farmacológica ya que los mecanismos de infección de muchos microorganismos patógenos es por medio de enzimas, al igual que enfermedades propias de la célula eucariota. En estos casos, vemos aún más clara la importancia de realizar un procesamiento de datos y análisis preciso sobre catálisis enzimática.

Para poder operar con dichos datos de manera precisa y rápida, es necesaria la implementación de un algoritmo, capaz de ordenar, operar y graficar con la menor intervención posible del usuario, ya que esto aumenta la posibilidad de error. La realización de este proyecto, presentaría gran utilidad, tanto para fines didáticos como aplicados, por lo tanto el uso de este código iría más allá que para fines de este curso.

La implemetación de dicho código, resultaría útil para aquella persona que presente conocimientos básicos o casi nulos de programación ya que el formato en el que se guardan los datos de cinética enzimática, debe ser consistente, por lo que sería fácil el análisis de cualquier archivo .xlsx con datos de este tipo. Además, las operaciones que se realizan durante este análisis son mayoritáreamente cálculo de valores inversos, regresiones lineales y graficaciones, lo cual abre la posibilidad al alumn@ de poner en práctica la programación modular para hacer más eficiente el código.

Objetivos

- Desarrollar un código automatizado de análisis cualitativo y cuantitativo de datos de inhibidores enzimáticos.

- Hacer cada tipo de análisis a través de un módulo, con el fin de practicar programación modular y hacer el código más eficiente.

Metodología

Este código está compuesto por dos componentes importantes. El primero es el análisis cualitativo, el cual se aprecia de mejor manera en el notebook de Jupyter que se encuntra en el repositorio de GitHub creado para este proyecto. El segundo son los scripts llamado script_1.jl y Linear_regression, los cuales cotienen el código del análisis cuantitaitvo.

Primero comenzamos definiendo una función que le permita al código interactúar con el usuario, para especificar cual es el archivo a analizar.
Esta función es equivalente a la función "input( )" de python, y la definimos como se muestra a continuación.

```
function input(prompt::String = "")
   print(prompt * " ")
   chomp(readline())
end
```

Con eso sale una opción en la terminal donde se le alimenta el archivo , la hoja de trabajo y el rango de los datos.
Posteriormente, definimos las siguientes variables.

```
file_name = input("Nombre del archivo")
data_sheet = input("Hoja de trabajo")
data_range=input("Rango de celdas")
data_no_header=input("Rango de celdas sin nombre de columna")
```

Con el input de los datos que se alimente a las varibales anteriores es que el programa lee la hoja de datos de excel.

```
data_header = XLSX.readdata(string(file_name), string(data_sheet), string(data_range)
```

El comando anterior con los datos del archivo a analizar se ve de la siguiente forma

```
dataset = XLSX.readdata(“Archivo.xlsx”, “Hoja_de_trabajo”, “Rango_de_datos”)
```

La función XLSX proviene de la paquetería ExcelFiles.jl ( ).

Para poder analizar los datos de velocidad inicial propios de cada inhibidor y la enzima control, se transfomaron los datos a una matríz, con el comando:

```
m = convert(Matrix{Floa64}, dataset)
```

donde: convert es la función que genera la matríz, {Float64}, el tipo de datos por la que estará formada y dataset son los datos que convertirá a matríz.

Después se calculó el inverso de cada elemento de la matríz, con el fin de poder generar los gráficos de Lineweaver-Burk y un análisis de regresión lineal con el cual se pueda calcular los parámetros enzimáticos, como km y velocidad máxima.

```
mat_inv = 1 ./ m
```

Con el inverso de los elementos de la matriz, se convirtió cada columna a un vector asignado a una variable, llamada con el nombre de la columna.

```
 sustrato = mat_inv[1:10]
```

```
control = mat_[11:20]

I = mat_inv[21:30]
.
.
.
```

En el notebook, se graficaron los datos de cada inhibidor vs el cotrol en gráficas independietes con las cunciones plot y scatter para obtenenr una línea contínua y los puntos de medición.

```
plot(sustrato, control, label = "control")
scatter!(sustrato, control, label = false)
plot!(sustrato, I, label = "Inhibidor_I")
scatter!(sustrato, I, label = false, xlabel="1/Sustrato [mM]", ylabel="1/Velocidad inicial [mM]/ms")
```

Por otro lado, la graficación de estos datos en el código de script_1.jl se implementó de forma diferente. Una de las ventajas de Julia es que permite generar un arreglo de figuras, cual ahorra código y espacio.
Se graficaron los datos de de cada inhibidor vs el experimento control en gráficas separadas asigandas a una variable, dándo un total de cuatro gráficas debido al número de inhibidores. Con estas cuatro gráficas se generó un arreglo conocido como "layout", donde es posible juntar las cuatro gráficas en una sola figura, lo cual ahorra código y espacio.

```
p1 = plot(plot(sustrato, control, label = false),
      scatter!(sustrato, control, label = false),
      scatter!(sustrato, X, label = false))



p2 = plot(plot(sustrato, control, label = false),
            scatter!(sustrato, control, label = false),
            scatter!(sustrato, I, label = false))

.
.
```

El comando para generar el "layout", fue:

```
fig = plot(p1, p2, p3, p4, layout=(2,2), fmt = :png, margin=-7mm, link = :x)
```

Al asignar a una variable la función de plot, permite guardar la imagen a través de una serie de condicionales, donde primero se abre una ventana de interacción con el usuario del código mediante la función input que definimos al principio, preguntando si se desea guardar la imagen. El código fue el siguiente.

```
save = input("Deseas guardar la figura")

if save == "si"
    png("layout.png")
else
```

```
    print("OK")
end
```

El análisis de regresión lineal se hizo en otro script llamado "Linear_regression.jl", el cual es llamado con la función include("linear_regression"). Este análisis se hizo mediante la paquetería “LinearRegression”, la cual permite guardar los coeficientes de la ecuación que cálcula. Dichos coeficientes son guardados en un vector de dos entradas, con el cual es posible operar. El comando con el que se realizó dicho análisis fue el siguiente:

```
lr_control = linregress(sustrato, control)

lr_I = lineregress(sustrato, I)
.
.
.
```

Para generar el arreglo numérico de los coeficientes de cada regresión lineal, asignamos las siguientes variables a la función coef( ), con el argumento de la variable a la que se le asignó la regresión lineal.

```
array_control = coef(lr_control)

array_I = coef(lr_I)
.
.
.
```

Con dichos arreglos se generó otro arreglo que los contenía, llamado "total_array".

```
total_array = [array_I, array_X, array_Y, array_Z]
```

Por último, a través de un ciclo "for" sabes cuales de los arreglos pertenecen a un inhibidor de tipo competitivo o acompetitivo.

```
for i in total_array
    if abs(i[1] - array_control[1]) < 0.001
        print("$i[1], Es inhibidor acompetitivo")
    end
    if abs(i[2] - array_control[2]) < 0.001
        print("$i[1], Es inhibidor competitivo")
    end
end
```

Posteriormente se definieron las ecuaciones de cada recta regresión lineal para poder obtener el parámetro enzimático de Km.

```
function eq_compet(x)
    y = 0.0099*x .+ 0.0099
    return y
end
```

Resultados

El primer resultado que se obtiene al correr el "notebook" es la gráfica de los datos crudos, la cual nos puede dar una idea de que realmente estamos trantando con inhibidores enzimáticos y se ve el efecto en el cambio de catálisis enzimática.

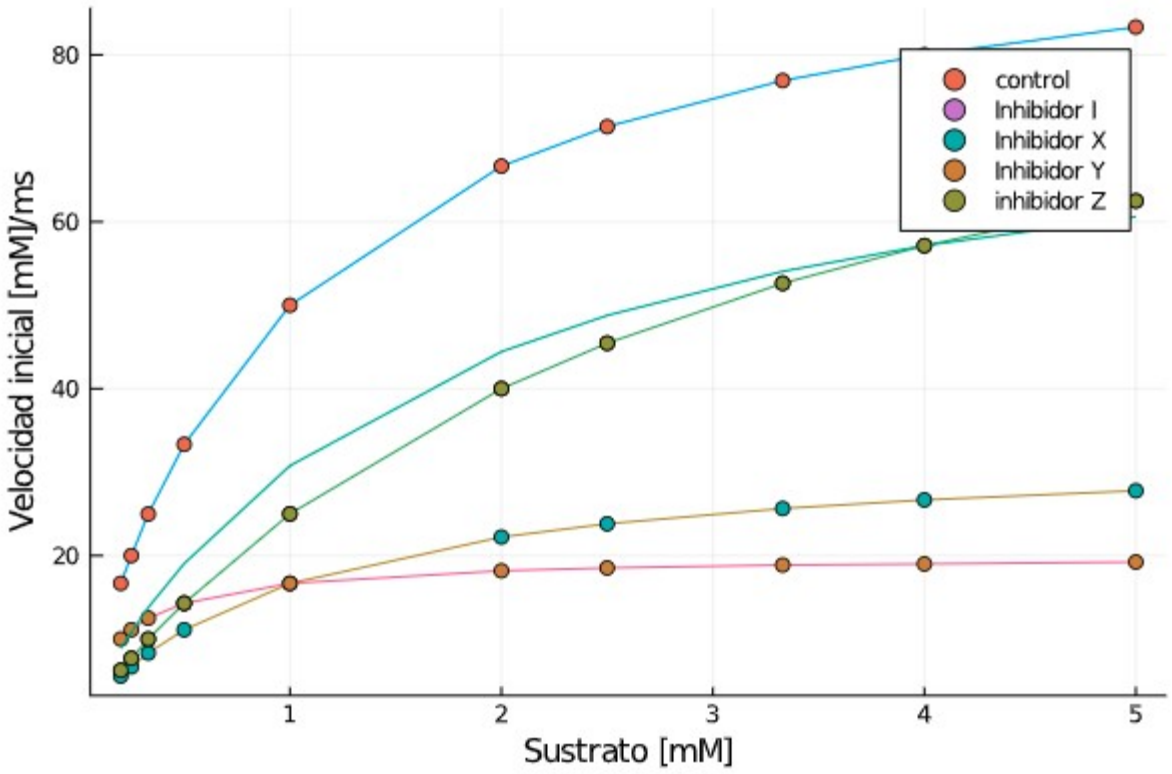

Figura 1. Gráfica de datos crudos de Velodidad inicial vs sustrato, junto con los datos diferentes inhibidores que se utilizaron para la implementación del código.

No obstante, la graficación de estos datos no está implementada en el script de análisis cuantitativo, ya que a partir de dicha gráfica, únicamente se obtiene una inferencia cualitativa.

A continuación se muestran las siguientes gráficos de "Lineweaver-Burk", a partir de los cuales ya es posible hacer una inferencia sobre que tipo de inhibidor que se está estudiando, pero no de los parámetros enzimáticos propios de cada inhibidor.

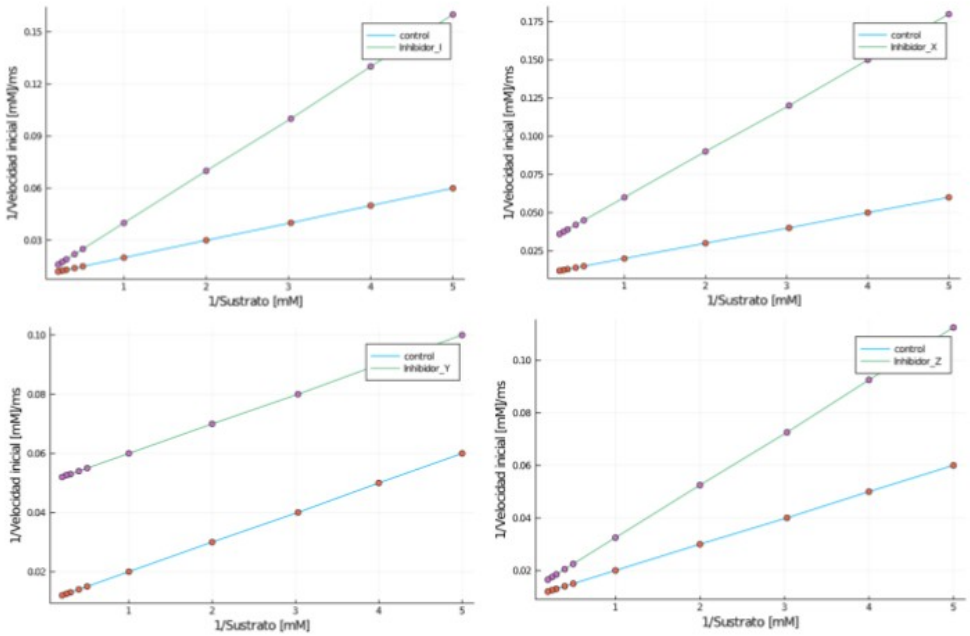

Figura 2. Graficos de "Lineweaver-Burk".

El arreglo de gráficas que se obtiene con el script_1.jl, se guarda como una sola imagen, a diferencia del notebook.

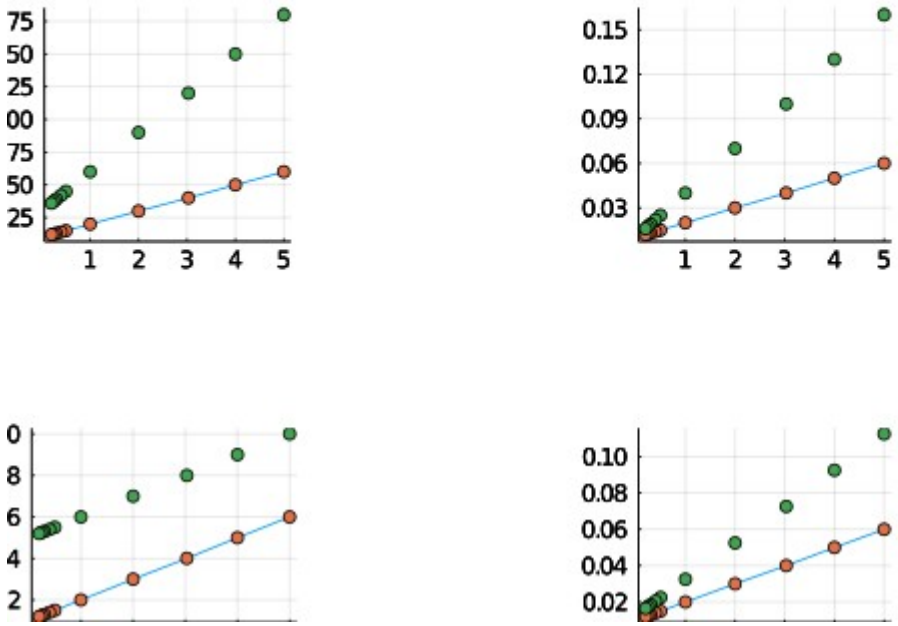

Figura 3. Arreglo de gráficas guaradades como una sola figura, lograda mediante la opción de layout en Julia.

Aquí es importante aclarar que el código para mostrar las etiquetas de los ejes y leyendas, no se logró implemetar satisfactoriamente ya que se requiere de mayor experiencia y conocimiento para programar dichos arreglos. No obstante, las inferencias pueden complementarse con el código del notebook.

La sección de código en la que se encuentra la comparación de coeficientes de regresión lineal, si es capaz de identificar que coeficientes son iguales, pero únicamente las indica como entradas del arreglo. No fue posible hacer que el algoritmo arrojara el nombre del inhibidor como se encuentra en el inicio de cada columa de la hoja de excel.

Por último, el cálculo del parámetro enzimático Km, tampoco se logró implementar de la forma esperada, ya que fue necesario la definición de las ecuaciones de regresión lineal para cada inhibidor y ver en que punto del dominio, la función valía cero. De acuerdo a Lineweaver y Burk, dicho valor corresponde al inverso de la Km.
Dado que esto fue necesario hacerlo a mano, nos aleja del objetivo de automatizar el análisis.

Discusión
El análisis de datos de cinética enzimática puede ser un buen ejemplo para adentrarse en el área de ciencia de datos y programación modular, ya que son datos que presentan un formato fácil de estandarizar y se utilizan funciones relativamente faciles de parametrizar. No obstante, la parte de automatización es lo que resulta de más reto. El script generado en este proyecto, cumple con un análisis cuantitativo y cualitativo de inhibidores enzimáticos. Sin embargo, hay secciones en las cuales es importante mejorar el código.
Una de ellas es el hecho de tener que definir "a mano" la asiganción del rango de columnas de la matriz inversa a una variable. e. g. `sustrato = mat_inv[1:10]`. El problema en este caso, no es tanto la asignación del rango de datos, ya que es posible implementar un código que reconozca únicamente las columnas de la matríz y las convierta a vectores. Sino el hecho de tener que asignar dicho vector a una variable para llevar a cabo las demás operaciones. Este código deber ser capaz de reconocer columna por columna del archivo de excel y ser capaz de asignar cada una a una varibale diferente.

El resto de aspectos mejorables en el código son de la misma naturaleza, por ejemplo, el código para generar los gráficos, al ser bastante repetitivo puede implementarse a través de un ciclo "for", si se tienen los conocimientos necesarios tanto de la lógica de un lenguage de programación, como la sintaxis del lenguage con el que se esté programando.

Por otro lado, la implemetación de este código de análisis de datos de cinética enzimática, reduce sustancialmente la probabilidad de cometer errores en la manipulación de los datos al momento de generar las gráficas o calcular la regresión lineal, además de realizar el análisis a una mayor velocidad que si se hiciera manualmente.

<u>Futuras Implementaciones en el código</u>
Este primer script, está diseñado para analizar datos que ya han sido procesados y están reportados en velocidad inicial [M]/t vs sustrato [M], no obstante muchas veces los datos crudos de cinética enzimática provienen de mediciones a través de espectrofotometría. Esto implica un paso extra en el procesamiento de los datos para convertirlos a velocidad inicial y trabajar con ellos. La implementación de este código está contemplada, ya que es parte esencial en cualquier análisis de datos de cinética enzimática.
Otra rama de código con la posibilidad de ser implementada, es un análisis estadístico, con el cual se pueda saber que enzima tiene mayor eficiencia catalítica y saber si cierta enzima sigue una cinética de Michaelis Menten.

Conclusiones
Se logró el objetivo de aprender y mejorar en la escritura de código, al igual que generar una herramienta para realizar un análisis cualitativo y cuantitativo. La primera implementación de este

script para el análisis de datos de cinética enzimática, deja bases sólidas para utilizar este código, ya sea con fines pedagógicos o de aplicación.
Por otro lado, la implementación de este código realizado en Julia, ayudó a maximizar los resultados deseados debido a la eficiencia en la sintaxis del lenguage.

Referencias

Chang R. (2008). Fisicoquímica tercera edición. Mc.Graw-Hill Interamericana. Paseo de la Reforma 1015, Torre A. Delegación Álvaro Obregon C. P. 01376, Ciudad de México. (p: 285)

Bezanson, Jeff, Edelman, Alan, Karpinsky, Stefan, Shah, Viral (2017). Julia, a first approach to numerical compuing. SIAM. 59(1):65-98