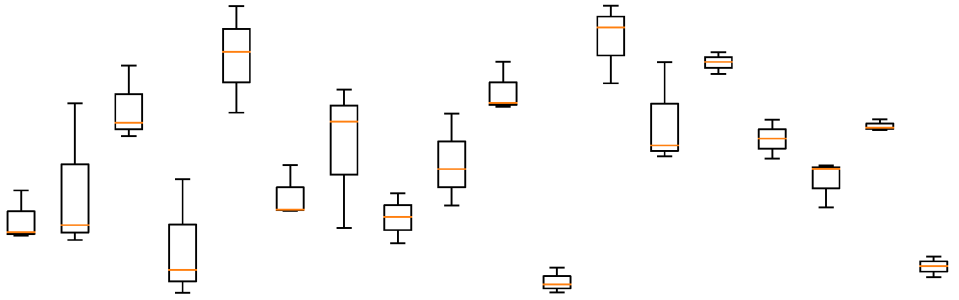
MANUAL DE USUARIO

Análisis de datos obtenidos del espectrómetro de emisión atómica con plasma de nitrógeno sostenido por microondas (MP-AES) utilizando Phyton.



González, Alan [1]; Wrobel, Katarzyna [1]; García, Elisa [2]; Gaytán, Luis [1]; Hernández, Jorge [3]; Lara, Deny [2]; Meza, Haydee [2]; Hernández, María[1]

[1]Dpto. de Química; [2]Dpto. de Ing. Química; [3]Dpto. de Biología Experimental.

Universidad de Guanajuato, Campus Gto, División de Ciencias Naturales y Exactas.

UNIVERSIDAD DE GUANAJUATO

XXVI Verano de la Ciencia Universidad de Guanajuato

UG

VERANOS DE LA CIENCIA

El manual tiene como objetivo apoyar a personas que tengan el interés de trabajar con Python, sin importar si se tiene o no, previo conocimiento del tema de Análisis de datos en Python. Es por ello por lo que el objetivo del manual no solo es ayudar al manejo y modificación del programa, también conlleva ser guía del cómo obtener Python desde cero.

Para un buen uso de los programas realizados, se decidió realizar un Manual de Usuario que permita el entendimiento de cómo manejar y modificar el código realizado en Python. Este manual está dirigido a profesores y estudiantes dedicados al análisis de datos con un enfoque quimiométrico que requieran un programa que facilite de manera automática el análisis de datos.

UNIVERSIDAD DE GUANAJUATO

Índice general

Capítulo 1

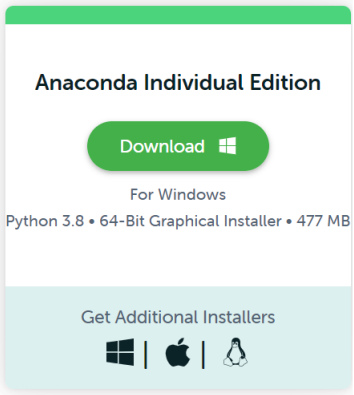
SUITE ANACONDA

Python es un lenguaje sencillo, rápido e ideal para trabajar y experimentar con datos obtenidos para **Análisis de datos**. Con el paso de los años, este se ha erigido como el principal recurso de la programación para el desarrollo de herramientas que permitan el análisis, tratado y procesamiento de los datos. Por eso mismo, en el mundo del *Big Data* tiene cada vez mayor peso para las investigaciones, aprender **Python** se torna una prioridad para aquellos que buscan adentrarse en el mundo del *data analytics*.

1.1. Descargar Anaconda

Para el uso del código es necesario tener el software *Anaconda*, el cual nos facilitará la tarea de instalar el ambiente e incluirá *JupyterLab*, siendo una aplicación que nos ofrecerá crear visualizaciones de datos.Anaconda
Anaconda es una multiplataforma que se puede utilizar para Windows, Linux y Mac. Para este caso se establecerá el procedimiento para la descarga e instalación en la plataforma de Windows.

Nos dirigimos a la Home de Anaconda e iremos a la sección de Download (`https://www.anaconda.com/products/individual`).
Procedemos a elegir la plataforma -Windows, Linux o Mac-.



Para este caso se eligió la versión de **Python** 3.6 con instalador gráfico a 64 bits.

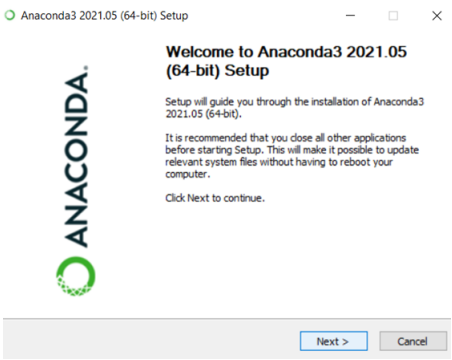
Con esto guardaremos en nuestro disco duro 477 MB (según sistema operativo) y obtendremos un archivo con el nombre similar a Anaconda3-2021.05-Windows-x86_ 64.exe.

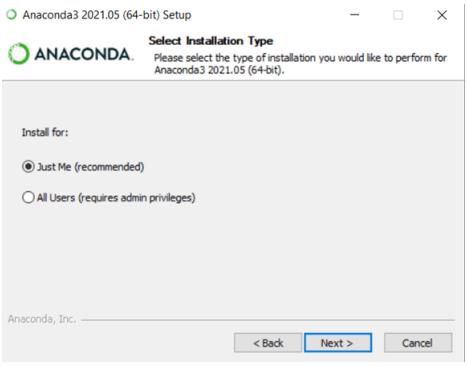
1.2. Instalar Anaconda

Para el siguiente paso se instalará la aplicación en nuestro sistema, es importante señalar que se deberán tener permisos de administrador si se instala para otros usuarios.
Ejecutamos el archivo anteriormente descargado dando doble clic. Dicha acción abrirá un wizard de instalación. Siguiendo los pasos, podemos seleccionar instalación sólo para nuestro usuario y/u otros, y seleccionar la ruta en disco donde instalaremos.

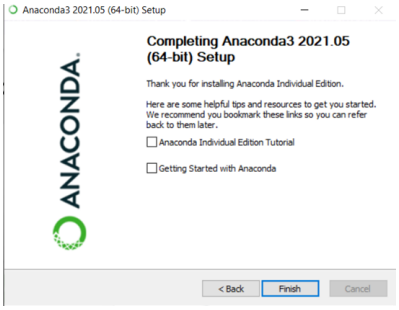
¿Qué es un Wizard?

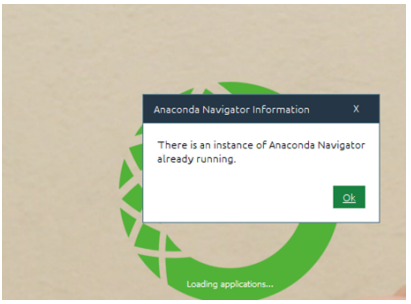
Un **wizard** es una pequeña aplicación que nos guiará paso a paso por todo el proceso para la instalación y personalización de una aplicación.





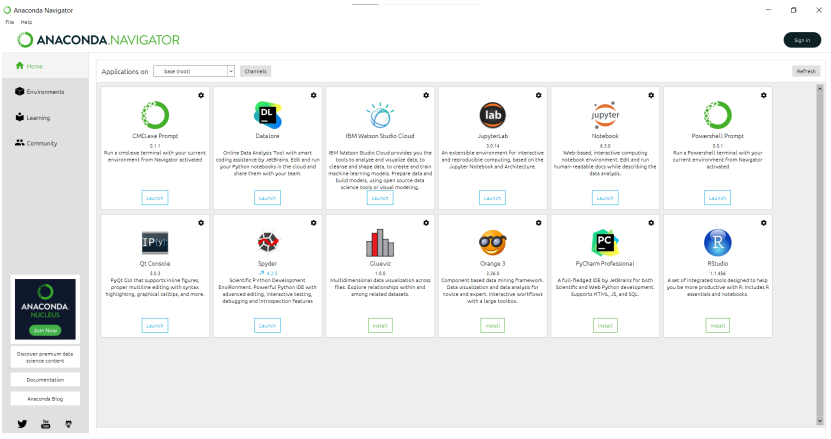
Al terminar la instalación le damos en *"finish"* para que comience a correr el programa. En caso de que esto no suceda, se busca la aplicación en el sistema y dando doble clic se comienza a correr.



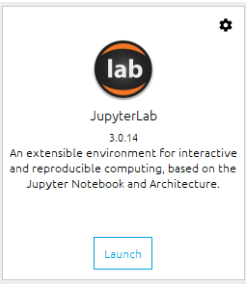


1.3. Iniciar Anaconda

Anaconda viene con una suite de herramientas gráficas llamada "*Anaconda Navigator*". Al iniciar la aplicación visualizaremos un navegador como la siguiente.



En la sección de Home podremos encontrar distintos canales, de los cuales para este caso iniciaremos *JupyterLab*.



Capítulo 2

JUPYTER LAB

JupyterLab es un entorno de desarrollo interactivo basado en la web para **JupyterNotebooks**, código y datos. JupyterLab es flexible, configure y organiza la interfaz de usuario para soportar una amplia gama de flujos de trabajo en ciencia de datos, computación científica y aprendizaje automático.

JupyterLab es extremadamente potente y flexible, ya que permite usar una increíble cantidad de fuentes desde texto, imágenes, hojas de cálculo, HTML, PDF, LaTeXy mucho más. Paralelamente permite trabajar con código escrito en cantidad de lenguajes populares como **Python**, R o Scala, editando y ejecutando el código sin salirnos de la propia aplicación. Además es capaz de usar diversos tipos de consolas de comandos y almacenar scripts para su ejecución.

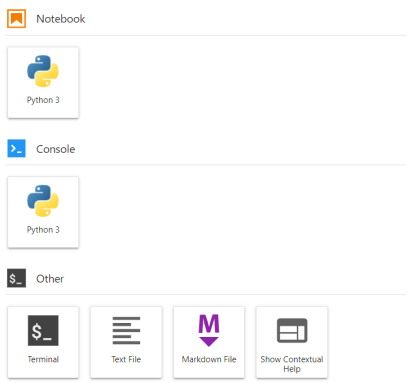
Los cuadernos de JupyterLab gracias a su formato flexible se pueden modificar con facilidad para evaluar sobre la marcha nuevos resultados. Asimismo, resulta sencillo producir nuevas versiones de un mismo cuaderno, convertirlo a otros formatos y compartirlo con otras personas, favoreciendo esta característica el trabajo colaborativo.

2.1. Abrir la aplicación web

Una vez desplegado el servidor con *JupyterLab* instalado, podremos acceder a la aplicación web. Posteriormente nos encontraremos con la interfaz de la aplicación, que tiene un aspecto como el que podemos ver en la siguiente imagen.

JypiterLab consta de una barra con el menú típico de opciones en la parte superior. Debajo de este menú la ventana se divide en dos grandes áreas:

- En el área de la izquierda: en "Files" se pueden gestionar los archivos y carpetas de la aplicación; en *"Running"* hacer un seguimiento de las sesiones y si es necesario detener núcleos (kernels), en *"Commands"* encontrar y ejecutar cualquier comando de *JupyterLab* y en *"Tabs"* seleccionar un área de trabajo entre las disponibles.
- En el área de la derecha llamada *"Launcher"* se muestran, por defecto, varios accesos directos para editar cuadernos, iniciar sesiones en las distintas consolas y editar de textos.



Desde el Launcher, en la parte principal de la pantalla, podemos crear fácilmente un nuevo Notebook, pulsando el botón **Python** 3» que hay en la sección «Notebook». Un notebook Jupyter se distribuye a lo largo de diversas celdas, donde podemos generar el contenido con toda la gama de elementos soportados por Jupyter.

2.1.1 Modificar un cuaderno

Cuando sea necesario abrir el cuaderno hacer doble clic sobre su nombre o seleccionar la opción **"Open"** del menú contextual. En ambos casos primero se abrirá el cuaderno y con posterioridad se ejecutarán todas las celdas. Después, para editar el contenido de alguna celda: hacer doble clic sobre una celda con texto o clic sobre una celda con código y reescribir su contenido.

Capítulo 3

MÓDULOS & LIBRERÍAS

Al momento de programar es recurrente utilizar -módulos, librerías, paquetes, entre otros- para facilitar el desarrollo de un programa sin tener que repetir códigos anteriormente desarrollados.

El lenguaje **Python** presenta una sintaxis y semántica referente a la biblioteca estándar. Existen muchos componentes opcionales que se incluyen en las distribuciones de **Python** y esto le permite al sistema contar con la portabilidad de los programas.

Las librerías **Python** son amplias, con gran cantidad de producciones en contenidos. Consta de módulos que permiten el acceso de funcionalidades del sistema como entrada y salida de archivos, soluciones estandarizadas a problemas de programación, etc. Los módulos en **Python** se encargan de alentar y reforzar la portabilidad de programas que separan especificaciones de la plataforma y conseguir **APIs neutrales**. Así podemos reconocer una librería como un conjunto de módulos cuya agrupación tiene una finalidad específica y que también puede ser invocada, tal como un módulo. Pero no es un módulo, sino un conjunto de ellos con una estructura determinada para lograr una finalidad.

¿Qué son los APIs?

Un **Aplplication Programming Interface** es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones.

3.1. Importar Módulos

Python tiene una gran colección de módulos que son llamados *Módulos de biblioteca estándar*, existen más de 200 módulos para tareas de programación comunes. Estos se pueden encontrar en la mayoría de las instalaciones en distintas plataformas. Dichos módulos se pueden importar de dos formas, utilizando la sentencia **import** y/o usando la sentencia **from**; con los mencionados comandos se encontrarán, compilarán y ejecutarán los módulos.
La principal diferencia es que la sentencia import trae todo el módulo, mientras que la instrucción from obtiene atributos específicos de los módulos.

Para el uso de este código se utilizaron las siguientes bibliotecas.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
from IPython.display import HTML
import statistics as stats
```

3.1.1 Pandas

Pandas es una herramienta de manipulación y análisis de datos de código abierto rápida, potente, flexible y fácil de usar, construida sobre el lenguaje de programación **Python**.

3.1.2 Matplotlib

Matplotlib es una biblioteca de **Python** orientada a la creación de visualización dinámicas o estáticas de datos. Al comenzar con esta biblioteca, tenemos que importar la misma y siempre importar además la de NumPy. (**?**)

3.1.3 NumPy

NumPy es una librería de **Python** especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos. Incorpora una nueva clase de objetos llamados arrays que permite representar colecciones de datos de un mismo tipo en varias dimensiones, y funciones muy eficientes para su manipulación.

¿Qué son los arrays?

Un **array** es una estructura de datos de un mismo tipo organizada en forma de tabla o cuadrícula de distintas dimensiones.

3.1.4 Scikit-Learn

Scikit-Learn cuenta con algoritmos de clasificación, regresión, clustering y reducción de dimensionalidad. Además, presenta la compatibilidad con otras librerías de **Python** como NumPy, SciPy y matplotlib.
La gran variedad de algoritmos y utilidades de Scikit-learn la convierten en la herramienta básica para empezar a programar y estructurar los sistemas de análisis datos y modelado estadístico. Los algoritmos de Scikit-Learn se combinan y depuran con otras estructuras de datos y aplicaciones externas como Pandas.

3.1.5 IPython

Es un intérprete de comandos para el lenguaje de programación **Python**, donde se ha potenciado la componente interactiva (de ahí la letra I en su nombre). Se ha desarrollado una interfaz, denominada Notebook, basada en un entorno computacional web que se visualiza en un navegador, y que permite la inclusión de cualquier elemento accesible a una pagina web, ademas de permitir la eje-

cución de código escrito en el lenguaje de programación **Python**. Esta interfaz se ejecuta separadamente del núcleo de ejecución de computación.

3.1.6 Statistics

El módulo statistics implementa muchas fórmulas estadísticas comunes para cálculos eficientes usando varios tipos numéricos de **Python** (int, float, Decimal, y Fracción). Ofrece funciones para el cálculo de valores estadísticos en el campo de los números reales. Algunas de las funciones que ofrece son:

- **statistics.mean()**: devuelve la media aritmética de los datos
- **statistics.median()**: devuelve la mediana de los datos
- **statistics.mode()**: devuelve la moda de los datos
- **statistics.stdev()**: devuelve la desviación estándar de la muestra de la población representada por los datos
- **statistics.variance()**: devuelve la varianza de la muestra de la población representada por los datos
- **statistics.pstdev()**: devuelve la desviación estándar de la población representada por los datos
- **statistics.pvariance()**: devuelve la varianza de la población representada por los datos

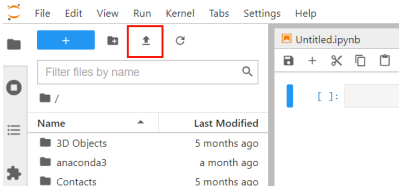
Capítulo 4

IMPORTAR DATOS

4.1. Excel a Python

Para importar datos desde un archivo en Excel a **Python** es un procedimiento muy simple. Para ello es necesario tener la tabla de datos en un archivo Excel con extensión **csv** con ayuda de pandas, por lo que ya se debe tener importada la librería de pandas.

Teniendo el archivo procederemos a guardarlo en nuestras carpetas de JupyterLab, dando clic en el icono de *"Upload Files"* como se observa en la imagen de la derecha. Seleccionamos el archivo y aplicamos *"abrir"*.



Nota. En caso que el icono *"Upload Files"* no sea visible, teclear `Ctrl+Shift+F`.

4.1.1 Cargar archivo

Como ultimo paso, utilizando una instrucción podemos leer el archivo Excel en **Python**.

```
df=pd.read_csv("ArchivoExcel")
```

- **df** es el nombre de una variable donde vamos a almacenar el archivo Excel, puede ser cualquier nombre que deseemos. Aunque es usual usar **df** por el término *dataframe*.

- pd.**read_csv**(**"ArchivoExcel"**) es una función de lectura, el nombre del archivo debe colocarse entre apóstrofos. En este caso se utiliza solamente el nombre del archivo porque lo hemos colocado en la misma ubicación donde tenemos instalado el programa **Python** que estamos ejecutando. Más específicamente el término **pd**, hace referencia al nombre con el cual importamos la librería de pandas. Mientras que **read_csv** es necesario para definir el formato del archivo.

4.1.2 Revisar Datos

En caso necesario de que el autor busque revisar los datos importados se utiliza la siguiente instrucción. El número en color verde en paréntesis es el número de filas que se visualizarán.

```
df . head (5)
```

	Label	Type	Date Time	Element	Element Label	Flags	Unadjusted Data	Concentration	Unit	Intensity	...	Replicates
0	Blank	BLK	03/04/2021 10:18	Ca 396.847	Ca	NaN	0	0	ppm	0.0	...	3
1	Blank	BLK	03/04/2021 10:18	Mg 285.213	Mg	NaN	0	0	ppm	0.0	...	3
2	Blank	BLK	03/04/2021 10:18	Na 589.592	Na	NaN	0	0	ppm	0.0	...	3
3	Blank	#	03/04/2021 10:18	Na 589.592	Na	NaN	0	0	ppm	0.0	...	3
4	Blank	BLK	03/04/2021 10:18	K 769.897	K	NaN	0	0	ppm	0.0	...	3

4.2. Cargar datos específicos

Es necesario solicitar qué **Elemento** en específico será utilizado para los distintos tratamientos que maneja el código. Es por ello que se introduce la instrucción de **input** para solicitar datos externos al código.

```
Element_a=input("Ingrese un elemento: ")
```

4.2.1 Menú de opciones

Por comodidad se agregaron las siguientes instrucciones a el código para que se imprima un menú de opciones con los elementos, esto para que al momento de solicitar un elemento en específico sea más sencillo el elegir.

```
from IPython.display import HTML
s = '<script type="text/Javascript»'
s ± 'var win = window.open(, "Title", "toolbar=no, location=no,
    directories=no, status=no, menubar=no, scrollbars=yes, resizable=yes,
    width=780, height=200, top= -(screen.height-400) + ", left=
    -(screen.width-840));'
s ± 'win.document.body.innerHTML = \" +
    df.to_html().replace("\n",'\\') + '\' ; '
s ± '</script>'
HTML(s)
```

	Label	Type	Date Time	Element	Element Label	Flags	Unadjusted Data	Concentration	Unit	Intensity	Concentration SD	Concentration %RSD	Intensity SD	Intensity %RSD	Replicates	Concentration Replicate 1	Concentration Replicate 2	Concentration Replicate 3	Intensity Replicate 1
0	Blank	BLK	03/04/2021 10:18	Ca 396.847	Ca	NaN	0	0	ppm	0.00	-	-	11.56	- 100.00	3	0	0	0	7.39
1	Blank	BLK	03/04/2021 10:18	Mg 285.213	Mg	NaN	0	0	ppm	0.00	-	-	34.16	- 100.00	3	0	0	0	-39.43
2	Blank	BLK	03/04/2021 10:18	Na 589.592	Na	NaN	0	0	ppm	0.00	-	-	8.76	- 100.00	3	0	0	0	6.48
3	Blank	#	03/04/2021 10:18	Na 589.592	Na	NaN	0	0	ppm	0.00	-	-	9.26	- 100.00	3	0	0	0	-10.58
4	Blank	BLK	03/04/2021 10:18	K 769.897	K	NaN	0	0	ppm	0.00	-	-	14.14	- 100.00	3	0	0	0	-14.75
5	Standard 1	STD	03/04/2021 10:20	Ca 396.847	Ca	NaN	0.1	0.1	ppm	5032.56	-	-	43.15	0.86	3	0.1	0.1	0.1	4985.59
6	Standard 1	STD	03/04/2021 10:20	Mg 285.213	Mg	NaN	0.1	0.1	ppm	8619.80	-	-	31.91	0.37	3	0.1	0.1	0.1	8589.77
7	Standard 1	STD	03/04/2021 10:20	Na 589.592	Na	NaN	0.1	0.1	ppm	10367.00	-	-	81.49	0.79	3	0.1	0.1	0.1	10458.09
8	Standard 1	#	03/04/2021 10:20	Na 589.592	Na	NaN	0.1	0.1	ppm	10374.57	-	-	37.66	0.36	3	0.1	0.1	0.1	10410.07
9	Standard 1	STD	03/04/2021 10:20	K 769.897	K	NaN	0.1	0.1	ppm	1727.59	-	-	10.77	0.62	3	0.1	0.1	0.1	1715.19
10	Standard 2	STD	03/04/2021 10:21	Ca 396.847	Ca	NaN	0.26	0.26	ppm	10677.54	-	-	64.49	0.6	3	0.26	0.26	0.26	10751.06
11	Standard 2	STD	03/04/2021 10:21	Mg 285.213	Mg	NaN	0.26	0.26	ppm	29614.40	-	-	95.19	0.32	3	0.26	0.26	0.26	29689.67
12	Standard 2	STD	03/04/2021 10:21	Na 589.592	Na	NaN	0.26	0.26	ppm	32059.20	-	-	414.27	1.29	3	0.26	0.26	0.26	32514.98
13	Standard 2	#	03/04/2021 10:21	Na 589.592	Na	NaN	0.26	0.26	ppm	32056.01	-	-	175.68	0.55	3	0.26	0.26	0.26	31853.34
14	Standard 2	STD	03/04/2021 10:21	K 769.897	K	NaN	0.26	0.26	ppm	5186.99	-	-	20.32	0.39	3	0.26	0.26	0.26	5208.59
15	Standard 3	STD	03/04/2021 10:23	Ca 396.847	Ca	NaN	0.5	0.5	ppm	19855.91	-	-	63.26	0.32	3	0.5	0.5	0.5	19813.00
16	Standard 3	STD	03/04/2021 10:23	Mg 285.213	Mg	NaN	0.5	0.5	ppm	56451.89	-	-	165.09	0.29	3	0.5	0.5	0.5	56366.46
17	Standard 3	STD	03/04/2021 10:23	Na 589.592	Na	NaN	0.5	0.5	ppm	61150.52	-	-	130.73	0.21	3	0.5	0.5	0.5	61044.82

Capítulo 5

CURVAS DE CALIBRACIÓN

Generamos un *dataframe* con los datos que nos interesa, es decir generamos etiquetas condicionales de los datos solicitados, bajo la siguiente instrucción.

```
CurvCalCadf = df.loc[(df['Type'] == 'STD') & (df['Element'] ==
                Element_a), 'Label','Intensity','Intensity SD', 'Concentration']]
CurvCalCadf.head(4)          # Revisar los datos
```

	Label	Intensity	Intensity SD	Concentration
5	Standard 1	5032.56	43.15	0.1
10	Standard 2	10677.54	64.49	0.26
15	Standard 3	19855.91	63.26	0.5
20	Standard 4	41508.45	502.17	1

5.1. Regresión lineal

El análisis de la regresión lineal se utiliza para predecir el valor de una variable según el valor de otra. La variable que desea predecir se denomina variable dependiente. La variable que está utilizando para predecir el valor de la otra

variable se denomina variable independiente.
Para obtener la curva de calibración es necesario reservar los datos de Concentración e Intensidad, por lo que se toman de las columnas y se transforman a una matriz con la librería NumPy anteriormente importada. Es importante transformar los valores tanto del eje X de *strings* a *floats*, además de reacomodar los datos especificando los ejes (X para las concentraciones y Y para las intensidades).

```
XC=((np.array(CurvCalCadf['Concentration'])) # Transforma la columna
                                              una matriz en NumPy
X_C=np.float_(XC)                    #Transforma X en floats y no strings
Y=np.array(CurvCalCadf['Intensity']) #Transforma la columna a una
                                      matriz en NumPy
X_C=X_C.reshape(-1,1) # Reacomoda los datos, X para las concentraciones,
                        Y para las intensidades
Y_I=Y.reshape(-1,1)
regr = linear_model.LinearRegression()
regr.LinearRegression(X_C,Y_I) # Hace la regresión lineal
```

Además, es necesario obtener los valores predichos para el eje y, junto a las instrucciones para que en la curva se impriman los valores de pendiente e intercepción.

```
y_I_pred = regr.predict(X_C)          # Obtiene los valores predichos para y
m_C=regr.coef_                                  # obtinene la pendiente
b_C=regr.intercept_                             # obtiene el intercepto
print('Coefficients_C: \n', regr.coef_)                      # Imprime
print('Intercept_C: \n', regr.intercept_)
print('Coefficient of determination_C: %.4f'
        % r2_score(Y_I,y_I_pred)) # Obtiene la R2
```

5.2. Cálculo de las concentraciones

5.2.1 Muestras

Para el cálculo de las concentraciones es forzoso importar del Archivo del Excel las intensidades de las muestras.

```
# Genera un dataframe con los datos que nos interesa
CurvCalCadf = df.loc[(df['Type'] == 'Sample') & (df['Element'] ==
                Element_a),'Label','Intensity','Intensity SD']]
CurvCalCadf.head(29)                    # Revisar los datos
bC =b_C                                 # intercepción
mC =m_C                                 # pendiente
M_C =1 /mC
YM_I =np.array(CurvCalCadf['Intensity'])  # Transforma la columna a una
                                        matriz en numpy
```

5.2.2 Matriz

Las siguientes instrucciones son para crear una matriz. El número en color verde entre paréntesis expresa el valor de los elementos de la matriz y el 29 el número de columnas. Para corroborar se agrego un **print** de la matriz formada.

```
matriz= []
for i in range(1):
        matriz.append([1]*29)
print (matriz)

b1_C=matriz             # matriz de 1's
b_c= bC*b1_C            # matriz producto de multiplicar el valor de la
                          intercepción por la matriz de 1's creada
yb_C=YM_I - b_c
XM_C=M_C*yb_C           # matriz de las concentraciones calculadas
```

5.3. Gráfica

Al imprimir una gráfica se usará la librería Matplotlib que está especializada en la creación de gráficos en dos dimensiones. Por lo que lo mandamos a llamar con el alías que lo importamos **plt**.

```
plt.subplot(1,1,1)
plt.scatter(X_C, Y_I, color='Black', label='calibración')
plt.scatter(XM_C, YM_I, color='Blue', label='muestras', marker='^ ')
plt.plot (X_C, y_I_pred, color='Green', linewidth=1,
          label='regresión lineal')

plt.title(f'Curva de calibración Element_a')
plt.xlabel('[μg/L]')
plt.ylabel('Intensidad')
plt.legend(loc='upper left')
plt.savefig('Curva de calibración.pdf',format='pdf')

#Mostrar el gráfico
plt.show()
```

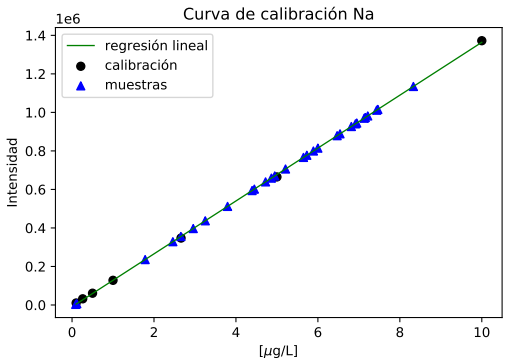


Figura 5.1: Gráfica de ejemplo. Curva de calibración Na 589.592.

Capítulo 6

MEDIA & DESVIACIÓN ESTÁNDAR

Importante señalar que muchos datos necesarios para calcular la **Media & Desviación Estándar** son obtenidos en el capitulo 5 *Curvas de Calibración*. Por lo que generamos un *dataframe* con los datos que nos interesa, es decir generamos etiquetas condicionales de los datos solicitados, bajo la siguiente instrucción.

```
CurvCalCadf = df.loc[(df['Type'] == 'STD') & (df['Element'] ==
              Element_a), 'Label','Intensity','Intensity SD', 'Concentration']]
CurvCalCadf.head(4)          # Revisar los datos
```

	Label	Intensity	Intensity SD	Concentration
9	Standard 1	1727.59	10.77	0.1
14	Standard 2	5186.99	20.32	0.26
19	Standard 3	9882.19	80.75	0.5
24	Standard 4	20587.86	102.38	1
29	Standard 5	55516.46	648.86	2.66

6.1. Matriz

Con la siguiente instrucción mandamos a llamar los datos de las muestras para hacer una matriz.

```
Muestras_adf=df.loc[(df['Type']=='Sample')&(df['Element']==Element_a),
                    ['Label','Intensity','Intensity SD']]
```

```
R1_df=df.loc[((df['Label']=='1.1R')|(df['Label']=='1.2R')|(df['Label']=='1.3R'))
        &(df['Type']=='Sample')&(df['Element']==Element_a),['Label',
        'Intensity','Intensity SD']]

R2_df=df.loc[((df['Label']=='2.1R')|(df['Label']=='2.2R')|(df['Label']=='2.3R'))
        &(df['Type']=='Sample')&(df['Element']==Element_a),['Label',
        'Intensity','Intensity SD']]

R3_df=df.loc[((df['Label']=='3.1R')|(df['Label']=='3.2R')|(df['Label']=='3.3R'))
        &(df['Type']=='Sample')&(df['Element']==Element_a),['Label',
        'Intensity','Intensity SD']]

R4_df=df.loc[((df['Label']=='4.1R')|(df['Label']=='4.2R')|(df['Label']=='4.3R'))
        &(df['Type']=='Sample')&(df['Element']==Element_a),['Label',
        'Intensity','Intensity SD']]

R5_df=df.loc[((df['Label']=='5.1R')|(df['Label']=='5.2R')|(df['Label']=='5.3R'))
        &(df['Type']=='Sample')&(df['Element']==Element_a),['Label',
        'Intensity','Intensity SD']]

R6_df=df.loc[((df['Label']=='6.1R')|(df['Label']=='6.2R')|(df['Label']=='6.3R'))
        &(df['Type']=='Sample')&(df['Element']==Element_a),['Label',
        'Intensity','Intensity SD']]

R7_df=df.loc[((df['Label']=='7.1R')|(df['Label']=='7.2R')|(df['Label']=='7.3R'))
        &(df['Type']=='Sample')&(df['Element']==Element_a),['Label',
```

```
        'Intensity','Intensity SD']]

R8_df=df.loc[((df['Label']=='8.1R')|(df['Label']=='8.2R')|(df['Label']=='8.3R'))
        &(df['Type']=='Sample')&(df['Element']==Element_a),['Label',
        'Intensity','Intensity SD']]

R9_df=df.loc[((df['Label']=='9.1R')|(df['Label']=='9.2R')|(df['Label']=='9.3R'))
        &(df['Type']=='Sample')&(df['Element']==Element_a),['Label',
        'Intensity','Intensity SD']]

R10_df=df.loc[((df['Label']=='10.1R')|(df['Label']=='10.2R')|(df['Label']==
        '10.3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),
        ['Label','Intensity','Intensity SD']]
```

Posterior a tener los valores de la matriz, convertimos la columna de la matriz en un arreglo de datos.

```
Z_1=np.array(R1_df['Intensity'])
Z_2=np.array(R2_df['Intensity'])
Z_3=np.array(R3_df['Intensity'])
Z_4=np.array(R4_df['Intensity'])
Z_5=np.array(R5_df['Intensity'])
Z_6=np.array(R6_df['Intensity'])
Z_7=np.array(R7_df['Intensity'])
Z_8=np.array(R8_df['Intensity'])
Z_9=np.array(R9_df['Intensity'])
Z_10=np.array(R10_df['Intensity'])
```

6.2. Calcular concentración

Al tener los datos de interés, se utilizan instrucciones que determinen los ejes x y ejes y para la gráfica. En este caso será la concentración para el eje de las x,

mientras que para el eje de las y lo contendrá la intensidad. Por consiguiente se tomarán las mismas instrucciones vista en *Regresión lineal.*

```
# Eje x de la curva
X=np.array(CurvCal_adf['Concentration'])
X_A=(np.float_(X)).reshape(-1,1)
# Eje y de la curva
Y_A=(np.array(CurvCal_adf['Intensity'])).reshape(-1,1)
# Crear la regresión lineal
regr = linear_model.LinearRegression()
# Correr el modelo utilizando los conjuntos creados
regr.fit(X_A,Y_A)
# Hacer predicciones de Y
Y_a_pred=regr.predict(X_A)
# Ecuación de la recta y=mx+b
m_a=regr.coef_
b_a=regr.intercept_
```

```
R1_df['Concentración']=((Z_1-b_a)/m_a).reshape(-1,1)
R2_df['Concentración']=((Z_2-b_a)/m_a).reshape(-1,1)
R3_df['Concentración']=((Z_3-b_a)/m_a).reshape(-1,1)
R4_df['Concentración']=((Z_4-b_a)/m_a).reshape(-1,1)
R5_df['Concentración']=((Z_5-b_a)/m_a).reshape(-1,1)
R6_df['Concentración']=((Z_6-b_a)/m_a).reshape(-1,1)
R7_df['Concentración']=((Z_7-b_a)/m_a).reshape(-1,1)
R8_df['Concentración']=((Z_8-b_a)/m_a).reshape(-1,1)
R9_df['Concentración']=((Z_9-b_a)/m_a).reshape(-1,1)
R10_df['Concentración']=((Z_10-b_a)/m_a).reshape(-1,1)
```

```
R1=np.array(R1_df['Concentración'])
R2=np.array(R2_df['Concentración'])
R3=np.array(R3_df['Concentración'])
R4=np.array(R4_df['Concentración'])
```

```
R5=np.array(R5_df['Concentración'])
R6=np.array(R6_df['Concentración'])
R7=np.array(R7_df['Concentración'])
R8=np.array(R8_df['Concentración'])
R9=np.array(R9_df['Concentración'])
R10=np.array(R10_df['Concentración'])
```

6.3. Estadísticas

Recordando que en en el capitulo 3 *MÓDULOS & LIBRERÍAS*, importamos **statistics**, del cual nos apoyaremos para calcular la media y desviación estándar dentro de la matriz anteriormente obtenida.

```
m1=stats.mean(R1)
st1=stats.pstdev(R1)
m2=stats.mean(R2)
st2=stats.pstdev(R2)
m3=stats.mean(R3)
st3=stats.pstdev(R3)
m4=stats.mean(R4)
st4=stats.pstdev(R4)
m5=stats.mean(R5)
st5=stats.pstdev(R5)
m6=stats.mean(R6)
st6=stats.pstdev(R6)
m7=stats.mean(R7)
st7=stats.pstdev(R7)
m8=stats.mean(R8)
st8=stats.pstdev(R8)
m9=stats.mean(R9)
st9=stats.pstdev(R9)
m10=stats.mean(R10)                    #m = media
st10=stats.pstdev(R10)                 #s = desviación estándar
```

6.4. Gráficas

Utilizando la librería Matplotlib para el desarrollo de gráficas de barras, damos las primeras instrucciones para almacenar los datos obtenidos, tales como la **media**, **desviación estándar**, además de guardar las **R´s** anteriormente calculadas. Subsiguiente a eso transferimos esos datos a un *dataframe*, que con ayuda de pandas (ps), se contendrá en el alías *"Barra"*.
Por último le damos valores gráficos visuales a nuestra gráfica de barras y aplicamos un **savefig** para guardar la gráfica obtenida en nuestro dispositivo.

6.4.1 Histograma

```
plt.figure()
Media=[m1,m2,m3,m5,m6,m7,m8,m9,m10]
Stdev=[st1,st2,st3,st5,st6,st7,st8,st9,st10]
Replica=['R1','R2','R3','R5','R6','R7','R8','R9','R10']

Barra=(pd.DataFrame('Replica':Replica,
                    'Concentración':Media,
                    'Desviación estándar':Stdev)).set_index('Replica')

width=0.65
fig, ax=plt.subplots()
ax.bar(Replica, Media,width,yerr=Stdev, label='Concentración',
       color='purple')
ax.set_ylabel('μg/g de muestra')
ax.set_title(f'Element_a')
ax.legend()
plt.savefig("Bar.pdf") #El texto en color rojo es el nombre con el cual se
                       quiere guardar el archivo, importante colocar el formato.
```

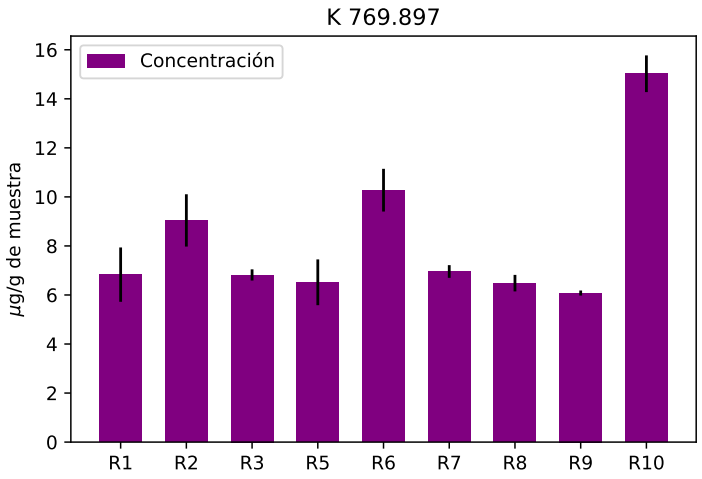


Figura 6.1: Gráfica de ejemplo. Curva de calibración K 769.897.

6.4.2 Diagrama de caja

```
fig1, ax1 = plt.subplots()
plt.ylabel('µg/g de muestra')                    #Etiqueta eje de las Y
plt.xlabel ('Grupos')                            #Etiqueta eje de las Y
ax1.set_title(Element_a)                         #Etiqueta del eje superior

#Datos graficados el término flatten usado como función
ax1.boxplot([R1.flatten(),R2.flatten(),R3.flatten(),R5.flatten(),R6.flatten(),
             R7.flatten(),R8.flatten(),R9.flatten(),R10.flatten()])

#Guardar gráfico
plt.savefig('Curva de calibración.pdf',format='pdf')
plt.show()
```

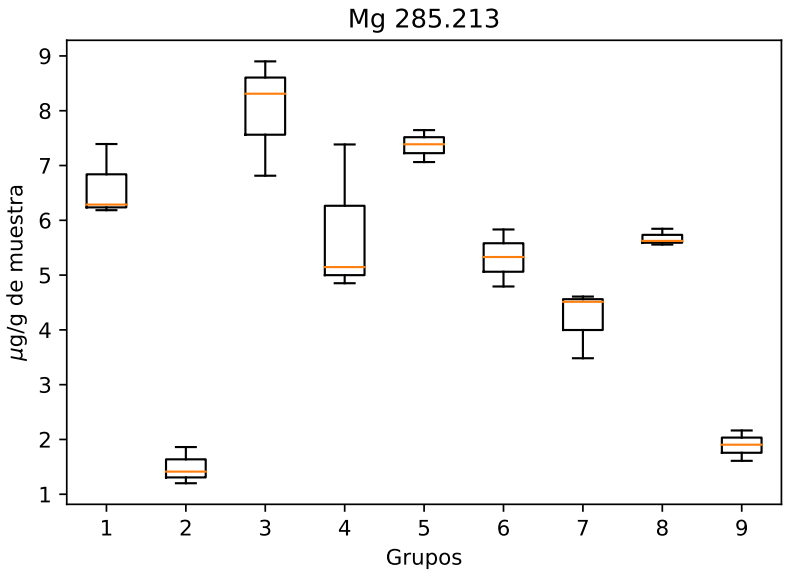


Figura 6.2: Gráfica de ejemplo. Curva de calibración Mg 285.213.

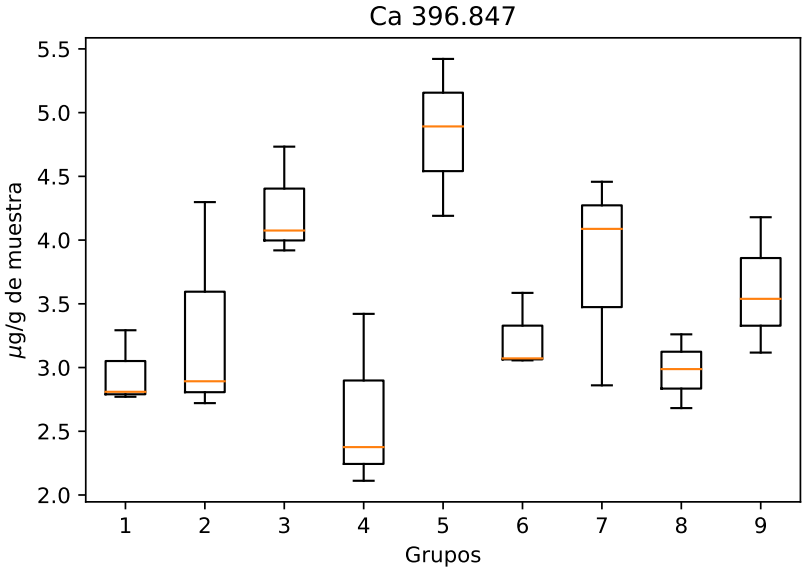


Figura 6.3: Gráfica de ejemplo. Curva de calibración Ca 396.847.

Capítulo 7

ANEXOS

Curva de calibración

July 19, 2021

[1]:
```
#Importar bibliotecas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score

#Leer el archivo
df=pd.read_csv('mayoritarios_marzo.csv')
#df.head(4) #Revisar los datos #muestra las primeras cuatro filas
```

[2]:
```
from IPython.display import HTML
s  = '<script type="text/Javascript">'
s += 'var win = window.open("", "Title", "toolbar=no, location=no,␣
 ↪directories=no, status=no, menubar=no, scrollbars=yes, resizable=yes,␣
 ↪width=780, height=200, top="+(screen.height-400)+", left="+(screen.
 ↪width-840));'
s += 'win.document.body.innerHTML = \'' + df.to_html().replace("\n",'\\') + '\';
 ↪'
s += '</script>'
#HTML(s)
```

[3]:
```
print ("Elementos \n Ca 396.847 \n Mg 285.213 \n Na 589.592 \n K 769.897")
#Solicitar elemento para la curva de calibración
Element_a = input("Ingrese una opción: ")
```

```
Elementos
 Ca 396.847
 Mg 285.213
 Na 589.592
 K 769.897

Ingrese una opción:  Na 589.592
```

[4]:
```
#ELEMENTO: Mg.
CurvCalCadf = df.loc[(df['Type'] == 'STD') & (df['Element'] ==␣
 ↪Element_a),['Label','Intensity','Intensity SD', 'Concentration']] #Genera un␣
 ↪dataframe con los datos que nos interesa.
```

```
CurvCalCadf.head(4) #Revisar los datos
```

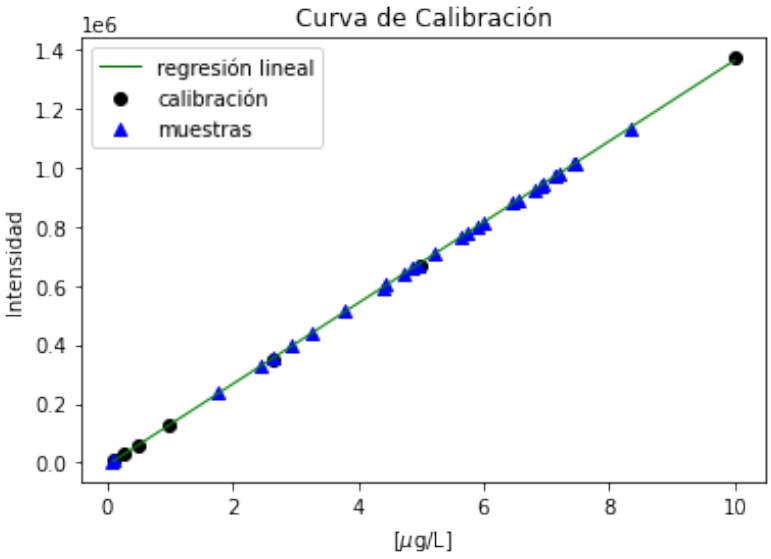
[4]:

	Label	Intensity	Intensity SD	Concentration
7	Standard 1	10367.00	81.49	0.1
12	Standard 2	32059.20	414.27	0.26
17	Standard 3	61150.52	130.73	0.5
22	Standard 4	128094.80	1037.80	1

[5]:
```
XC=((np.array(CurvCalCadf['Concentration']))) #Transforma la columna a una␣
 ↪matriz en NumPy
X_C=np.float_(XC) #Transforma X en floats y no strings
Y=np.array(CurvCalCadf['Intensity']) #Transforma la columna a una matriz en␣
 ↪NumPy
X_C=X_C.reshape(-1,1) #Reacomoda los datos, X para las concentraciones, Y para␣
 ↪las intensidades
Y_I=Y.reshape(-1,1)
regr = linear_model.LinearRegression()
regr.fit(X_C,Y_I) #Hace la regresión lineal

y_I_pred = regr.predict(X_C) #Obtiene los valores predichos para y
m_C=regr.coef_ #obtinene la pendiente
b_C=regr.intercept_ #obtiene el intercepto
print('Coefficients_C: \n', regr.coef_) #Imprime
print('Intercept_C: \n', regr.intercept_)
print('Coefficient of determination_C: %.4f'
      % r2_score(Y_I,y_I_pred)) #Obtiene la R2

#Muestras
CurvCalCadf = df.loc[(df['Type'] == 'Sample') & (df['Element'] ==␣
 ↪Element_a),['Label','Intensity','Intensity SD']] #Genera un dataframe con␣
 ↪los datos que nos interesa.
CurvCalCadf.head(29) #Revisar los datos
bC=b_C #intercepción
mC=m_C #pendiente
M_C=1/mC
#Importar del excel las intesisdades de las muestras
YM_I=np.array(CurvCalCadf['Intensity']) #Transforma la columna a una matriz en␣
 ↪numpy

#Calculo de la concentraciones
#crear la matriz b
matriz= []
for i in range(1): #matriz de una 1
    matriz.append([1]*29) #El número entre parentesis expresa el valor de los␣
 ↪elementos de la matriz y el 29 el número de columnas
```

```
print (matriz)

b1_C=matriz #matriz de 1´s
b_c= bC*b1_C #matriz producto de multiplicar el valor de la intercepción por la␣
 ↪matriz de 1´s creada.
yb_C=YM_I  - b_c
XM_C=M_C*yb_C #matriz de las concentraciones calculadas

#Graficar la curva de calibracion
plt.subplot(1,1,1)
plt.scatter(X_C, Y_I, color='Black', label='calibración') #
plt.scatter(XM_C, YM_I, color='Blue', label='muestras', marker='^')
plt.plot (X_C, y_I_pred, color='Green', linewidth=1, label='regresión lineal')
plt.title('Curva de Calibración')
plt.xlabel('[$\mu$g/L]')
plt.ylabel('Intensidad')
plt.legend()
plt.savefig("Curva de Calibración.jpg")
```

```
Coefficients_C:
 [[137305.76534393]]
Intercept_C:
 [-9138.76135906]
Coefficient of determination_C: 0.9998
[[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1]]
```



[ ]:

[ ]:

Histograma

July 21, 2021

[11]:
```
#Importar datos en formato csv
import pandas as pd
#Leer el archivo de la hoja de datos en formato csv
df=pd.read_csv('mayoritarios_marzo.csv')
#Revisar los datos
#df.head(4)
```

[12]:
```
# Importar el módulo pyplot con el alias plt para haccer las gráficas
import matplotlib.pyplot as plt
#Crear el entorno de matrices
import numpy as np
#Regresión líneal de mínimos cuadrados ordinarios
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

[13]:
```
from IPython.display import HTML
s  = '<script type="text/Javascript">'
s += 'var win = window.open("", "Title", "toolbar=no, location=no,␣
 ↪directories=no, status=no, menubar=no, scrollbars=yes, resizable=yes,␣
 ↪width=780, height=200, top="+(screen.height-400)+", left="+(screen.
 ↪width-840));'
s += 'win.document.body.innerHTML = \'' + df.to_html().replace("\n",'\\') + '\';
 ↪'
s += '</script>'
#HTML(s)
```

[14]:
```
print ("Elementos \n Ca 396.847 \n Mg 285.213 \n Na 589.592 \n K 769.897")
#Solicitar elemento para la curva de calibración
Element_a = input("Ingrese una opción: ")
```

```
Elementos
 Ca 396.847
 Mg 285.213
 Na 589.592
 K 769.897

Ingrese una opción:  Na 589.592
```

[22]:
```
#Condicional de los datos, etiquetas
CurvCal_adf=df.
 ↪loc[(df['Type']=='STD')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD','Concentration']]
CurvCal_adf.head(4)
```

[22]:

	Label	Intensity	Intensity SD	Concentration
7	Standard 1	10367.00	81.49	0.1
12	Standard 2	32059.20	414.27	0.26
17	Standard 3	61150.52	130.73	0.5
22	Standard 4	128094.80	1037.80	1

[23]:
```
#Hacer una matriz de los datos de la muestra simple
Muestras_adf=df.
 ↪loc[(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]

R1_df=df.loc[((df['Label']=='1.1R')|(df['Label']=='1.2R')|(df['Label']=='1.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
R2_df=df.loc[((df['Label']=='2.1R')|(df['Label']=='2.2R')|(df['Label']=='2.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
R3_df=df.loc[((df['Label']=='3.1R')|(df['Label']=='3.2R')|(df['Label']=='3.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
#R4_df=df.loc[((df['Label']=='4.1R')|(df['Label']=='4.2R')|(df['Label']=='4.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
R5_df=df.loc[((df['Label']=='5.1R')|(df['Label']=='5.2R')|(df['Label']=='5.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
R6_df=df.loc[((df['Label']=='6.1R')|(df['Label']=='6.2R')|(df['Label']=='6.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
R7_df=df.loc[((df['Label']=='7.1R')|(df['Label']=='7.2R')|(df['Label']=='7.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
R8_df=df.loc[((df['Label']=='8.1R')|(df['Label']=='8.2R')|(df['Label']=='8.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
R9_df=df.loc[((df['Label']=='9.1R')|(df['Label']=='9.2R')|(df['Label']=='9.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
```

```
R10_df=df.loc[((df['Label']=='10.1R')|(df['Label']=='10.2R')|(df['Label']=='10.
↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
↪SD']]
```

[24]:
```
#Convertir la columna de la matriz en un arreglo de datos
Z_1=np.array(R1_df['Intensity'])
Z_2=np.array(R2_df['Intensity'])
Z_3=np.array(R3_df['Intensity'])
#Z_4=np.array(R4_df['Intensity'])
Z_5=np.array(R5_df['Intensity'])
Z_6=np.array(R6_df['Intensity'])
Z_7=np.array(R7_df['Intensity'])
Z_8=np.array(R8_df['Intensity'])
Z_9=np.array(R9_df['Intensity'])
Z_10=np.array(R10_df['Intensity'])
```

[25]:
```
#Eje x de la curva
X=np.array(CurvCal_adf['Concentration'])
X_A=(np.float_(X)).reshape(-1,1)
#Eje y de la curva
Y_A=(np.array(CurvCal_adf['Intensity'])).reshape(-1,1)
#Crear la regresión lineal
regr = linear_model.LinearRegression()
#Correr el modelo utilizando los conjuntos creados
regr.fit(X_A,Y_A)
#Hacer predicciones de Y
Y_a_pred=regr.predict(X_A)
#Ecuación de la recta y=mx+b
m_a=regr.coef_
b_a=regr.intercept_

#Calcular la concentración
R1_df['Concentración']=((Z_1-b_a)/m_a).reshape(-1,1)
R2_df['Concentración']=((Z_2-b_a)/m_a).reshape(-1,1)
R3_df['Concentración']=((Z_3-b_a)/m_a).reshape(-1,1)
#R4_df['Concentración']=((Z_4-b_a)/m_a).reshape(-1,1)
R5_df['Concentración']=((Z_5-b_a)/m_a).reshape(-1,1)
R6_df['Concentración']=((Z_6-b_a)/m_a).reshape(-1,1)
R7_df['Concentración']=((Z_7-b_a)/m_a).reshape(-1,1)
R8_df['Concentración']=((Z_8-b_a)/m_a).reshape(-1,1)
R9_df['Concentración']=((Z_9-b_a)/m_a).reshape(-1,1)
R10_df['Concentración']=((Z_10-b_a)/m_a).reshape(-1,1)
```

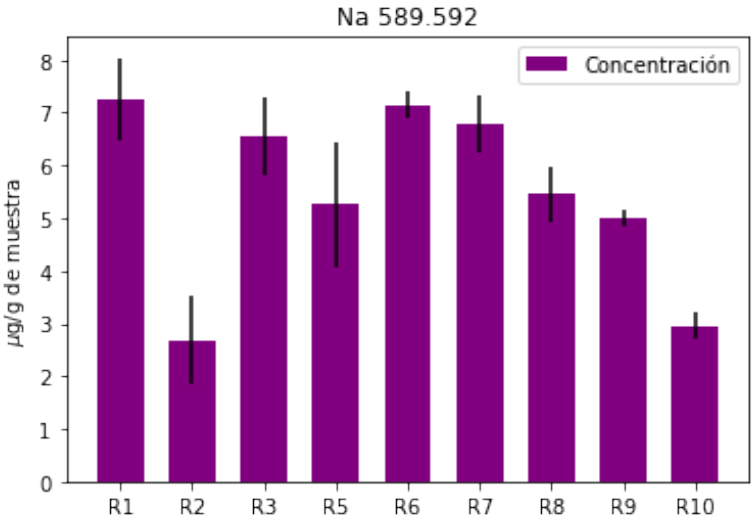
[26]:
```
R1=np.array(R1_df['Concentración'])
R2=np.array(R2_df['Concentración'])
R3=np.array(R3_df['Concentración'])
#R4=np.array(R4_df['Concentración'])
```

```
R5=np.array(R5_df['Concentración'])
R6=np.array(R6_df['Concentración'])
R7=np.array(R7_df['Concentración'])
R8=np.array(R8_df['Concentración'])
R9=np.array(R9_df['Concentración'])
R10=np.array(R10_df['Concentración'])
```

[27]:
```
import statistics as stats
m1=stats.mean(R1)
st1=stats.pstdev(R1)
m2=stats.mean(R2)
st2=stats.pstdev(R2)
m3=stats.mean(R3)
st3=stats.pstdev(R3)
#m4=stats.mean(R4)
#st4=stats.pstdev(R4)
m5=stats.mean(R5)
st5=stats.pstdev(R5)
m6=stats.mean(R6)
st6=stats.pstdev(R6)
m7=stats.mean(R7)
st7=stats.pstdev(R7)
m8=stats.mean(R8)
st8=stats.pstdev(R8)
m9=stats.mean(R9)
st9=stats.pstdev(R9)
m10=stats.mean(R10)
st10=stats.pstdev(R10)
```

[28]:
```
plt.figure()
Media=[m1,m2,m3,m5,m6,m7,m8,m9,m10]
Stdev=[st1,st2,st3,st5,st6,st7,st8,st9,st10]
Replica=['R1','R2','R3','R5','R6','R7','R8','R9','R10']

Barra=(pd.DataFrame({'Replica':Replica,
                     'Concentración':Media,
                     'Desviación estándar':Stdev})).set_index('Replica')

width=0.65
fig, ax=plt.subplots()
ax.bar(Replica, Media,width,yerr=Stdev, label='Concentración', color='purple')
ax.set_ylabel('$\mu$g/g de muestra')
ax.set_title(f'{Element_a}')
ax.legend()
plt.savefig("Bar.pdf")
```

```
<Figure size 432x288 with 0 Axes>
```


Na 589.592
Concentración
µg/g de muestra
0
1
2
3
4
5
6
7
8
R1
R2
R3
R5
R6
R7
R8
R9
R10

[ ]:

BloxPlot

July 21, 2021

[21]:
```
# Importar bibliotecas

#Importar datos en formato csv
import pandas as pd
#Leer el archivo de la hoja de datos en formato csv
df=pd.read_csv('mayoritarios_marzo.csv')
#Revisar los datos
#df.head(4)
```

[22]:
```
# Importar el módulo pyplot con el alias plt para haccer las gráficas
import matplotlib.pyplot as plt
#Crear el entorno de matrices
import numpy as np
#Regresión líneal de mínimos cuadrados ordinarios
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

[23]:
```
from IPython.display import HTML
s  = '<script type="text/Javascript">'
s += 'var win = window.open("", "Title", "toolbar=no, location=no,␣
 ↪directories=no, status=no, menubar=no, scrollbars=yes, resizable=yes,␣
 ↪width=780, height=200, top="+(screen.height-400)+", left="+(screen.
 ↪width-840));'
s += 'win.document.body.innerHTML = \'' + df.to_html().replace("\n",'\\') + '\';
 ↪'
s += '</script>'
#HTML(s)
```

[24]:
```
print ("Elementos \n Ca 396.847 \n Mg 285.213 \n Na 589.592 \n K 769.897")
#Solicitar elemento para la curva de calibración
Element_a = input("Ingrese una opción: ")
```

```
Elementos
 Ca 396.847
 Mg 285.213
 Na 589.592
 K 769.897
```

```
Ingrese una opción:  Na 589.592
```

[30]:
```
#Condicional de los datos, etiquetas
CurvCal_adf=df.
 ↪loc[(df['Type']=='STD')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD','Concentration']]
CurvCal_adf.head(4)
```

[30]:

	Label	Intensity	Intensity SD	Concentration
7	Standard 1	10367.00	81.49	0.1
12	Standard 2	32059.20	414.27	0.26
17	Standard 3	61150.52	130.73	0.5
22	Standard 4	128094.80	1037.80	1

[31]:
```
#Hacer una matriz de los datos de la muestra simple
Muestras_adf=df.
 ↪loc[(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]

R1_df=df.loc[((df['Label']=='1.1R')|(df['Label']=='1.2R')|(df['Label']=='1.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
R2_df=df.loc[((df['Label']=='2.1R')|(df['Label']=='2.2R')|(df['Label']=='2.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
R3_df=df.loc[((df['Label']=='3.1R')|(df['Label']=='3.2R')|(df['Label']=='3.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
#R4_df=df.loc[((df['Label']=='4.1R')|(df['Label']=='4.2R')|(df['Label']=='4.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
R5_df=df.loc[((df['Label']=='5.1R')|(df['Label']=='5.2R')|(df['Label']=='5.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
R6_df=df.loc[((df['Label']=='6.1R')|(df['Label']=='6.2R')|(df['Label']=='6.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
R7_df=df.loc[((df['Label']=='7.1R')|(df['Label']=='7.2R')|(df['Label']=='7.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
R8_df=df.loc[((df['Label']=='8.1R')|(df['Label']=='8.2R')|(df['Label']=='8.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
R9_df=df.loc[((df['Label']=='9.1R')|(df['Label']=='9.2R')|(df['Label']=='9.
 ↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
 ↪SD']]
```

```
R10_df=df.loc[((df['Label']=='10.1R')|(df['Label']=='10.2R')|(df['Label']=='10.
↪3R'))&(df['Type']=='Sample')&(df['Element']==Element_a),['Label','Intensity','Intensity␣
↪SD']]

#Convertir la columna de la matriz en un arreglo de datos
Z_1=np.array(R1_df['Intensity'])
Z_2=np.array(R2_df['Intensity'])
Z_3=np.array(R3_df['Intensity'])
#Z_4=np.array(R4_df['Intensity'])
Z_5=np.array(R5_df['Intensity'])
Z_6=np.array(R6_df['Intensity'])
Z_7=np.array(R7_df['Intensity'])
Z_8=np.array(R8_df['Intensity'])
Z_9=np.array(R9_df['Intensity'])
Z_10=np.array(R10_df['Intensity'])
```

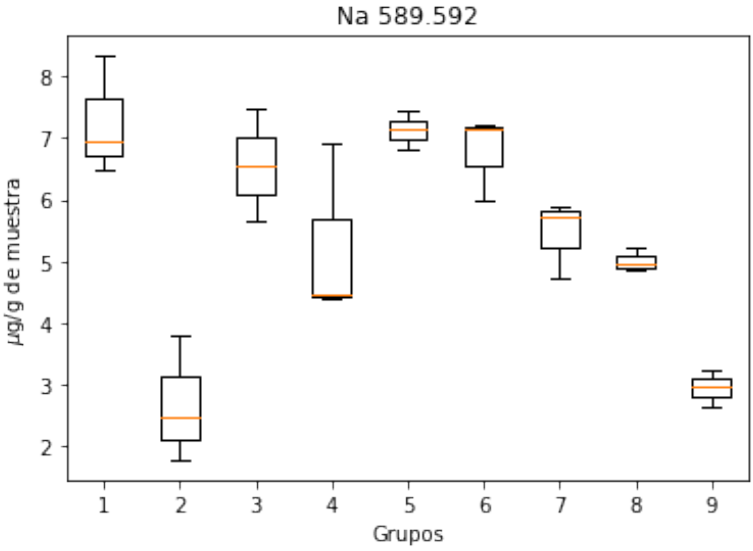
[32]:
```
#Eje x de la curva
X=np.array(CurvCal_adf['Concentration'])
X_A=(np.float_(X)).reshape(-1,1)
#Eje y de la curva
Y_A=(np.array(CurvCal_adf['Intensity'])).reshape(-1,1)
#Crear la regresión lineal
regr = linear_model.LinearRegression()
#Correr el modelo utilizando los conjuntos creados
regr.fit(X_A,Y_A)
#Hacer predicciones de Y
Y_a_pred=regr.predict(X_A)
#Ecuación de la recta y=mx+b
m_a=regr.coef_
b_a=regr.intercept_
```

[33]:
```
#Calcular la concentración
R1_df['Concentración']=((Z_1-b_a)/m_a).reshape(-1,1)
R2_df['Concentración']=((Z_2-b_a)/m_a).reshape(-1,1)
R3_df['Concentración']=((Z_3-b_a)/m_a).reshape(-1,1)
#R4_df['Concentración']=((Z_4-b_a)/m_a).reshape(-1,1)
R5_df['Concentración']=((Z_5-b_a)/m_a).reshape(-1,1)
R6_df['Concentración']=((Z_6-b_a)/m_a).reshape(-1,1)
R7_df['Concentración']=((Z_7-b_a)/m_a).reshape(-1,1)
R8_df['Concentración']=((Z_8-b_a)/m_a).reshape(-1,1)
R9_df['Concentración']=((Z_9-b_a)/m_a).reshape(-1,1)
R10_df['Concentración']=((Z_10-b_a)/m_a).reshape(-1,1)
```

[34]:
```
R1=np.array(R1_df['Concentración'])
R2=np.array(R2_df['Concentración'])
R3=np.array(R3_df['Concentración'])
#R4=np.array(R4_df['Concentración'])
```

```
R5=np.array(R5_df['Concentración'])
R6=np.array(R6_df['Concentración'])
R7=np.array(R7_df['Concentración'])
R8=np.array(R8_df['Concentración'])
R9=np.array(R9_df['Concentración'])
R10=np.array(R10_df['Concentración'])
```

[35]:
```
#Boxplot
fig1, ax1 = plt.subplots()
plt.ylabel('$\mu$g/g de muestra') #Etiqueta eje de las Y
plt.xlabel ('Grupos') #Etiqueta eje de las Y
ax1.set_title(Element_a) #Etiqueta del eje superior
ax1.boxplot([R1.flatten(),R2.flatten(),R3.flatten(),R5.flatten(),R6.
 ↪flatten(),R7.flatten(),R8.flatten(),R9.flatten(),R10.flatten()]) #Datos␣
 ↪graficados #el termino flatten usado como función.
plt.savefig('Curva de calibración.pdf',format='pdf') #Guardar grafico
plt.show()
```



[ ]:

Bibliografía

[Anaconda] : *Anaconda | The World's Most Popular Data Science Platform*. – URL https://www.anaconda.com/. – Zugriffsdatum: 2021-07-21

[noauthor$_n umpy_n odate$] : *NumPy*. – –URL. – Zugriffsdatum: 2021-07-21

noauthor$_p andas_n odate$: *pandas - Python Data Analysis Library*. – –URL. – Zugriffsdatum: 2021-07-21

noauthor$_s cikitlearn_n odate$: *Scikitlearn - Buscar con Google*. – –URL. – Zugriffsdatum: 2021-07-21

noauthor$_s tatistics_n odate$: *statistics — Mathematical statistics functions — Python 3.9.6 docu* –URL. – Zugriffsdatum: 2021-07-21