UNIVERSIDAD DE GUANAJUATO

Manual de prácticas para Gaussian 09. Verano científico 2023

AUTORES:

BALANZAR URZUA IRVING EDUARDO
GUTIERREZ GARCIA INGRID MICHELLE
MALAGÓN ALVAREZ MARICELA
MARTINEZ VEGA JORGE
PACHECO GONZALEZ CARLOS ALDAIR
ROMERO TORRES DAPHNE HANIEL

ASESORES:

DR. CASTELLANOS AGUILA JESUS EDUARDO

DR. SHULIKA OLEKSIY

Índice

Introducción

Este conjunto de experiencias está diseñado con la meta de enfatizar detalles técnicos que a veces son pasados por alto cuando ejecutamos la teoría del funcional de la densidad (DFT). La implementación de interfaz gráfica de usuario, GaussView[1] ciertamente ayudan a hacer los programas de química computacional más accesibles. Como consecuencia, los usuarios usan muchos parámetros por default. Ciertas precauciones deben ser tomadas para hacer que los ajustes por default coincidan con resultados recientes de estudios,[2,3] pero su uso ciego no representa una buena práctica en la química computacional.

Esta libreta de laboratorio se pretende usar principalmente como una guía para principiantes o no expertos en el campo, sin embargo, algunas de las experiencias puedes ser usadas por expertos en un ambiente de investigación también, donde se destacan varios aspectos pasados por alto en la DFT. También mostramos que las bases de datos no son lo único importante para las búsquedas químicas, pero estos pueden ser efectivamente usados como una herramienta educacional también. En efecto, todas estas estructuras son tomadas por el GMTKN55,[4] MGCDB84,[5] o Minnesota 2015[6–8] en sus bases de datos. Los ejercicios son propuestos como acercamiento a la introducción de DFT y sus **NO HACER** representa sugerencias para funcionales o conjunto de bases. También, esta libreta **NO ES** un sustituto de un libro de texto en la fundamentación de la teoría de computación química y/o la DFT. Algunas nociones que son pre-requisitos y que son requeridos son: la naturaleza de muchas aproximaciones del funcional de intercambio-correlación, la definición de un conjunto de bases, la construcción de un mallado para un espacio real de integración, y usos de técnicas de análisis de estabilidad. Una lista sugerida de libros de texto para establecer o revisar estas nociones se da al final de la sección de bibliografía al final de este documento. Dado que tratamos principalmente con cálculos moleculares, no hay experiencias diseñadas para aplicar la DFT a la física del estado sólido o a la ciencia de los materiales. Los comentarios dados al final de cada experiencia se aplican estrictamente a las aplicaciones moleculares del estado fundamental de la DFT, y podría ser peligroso aplicarlos a esos campos. En cambio, incluimos algunos de los artículos de revisión más recientes sobre estos temas en la bibliografía.

Los cálculos se pueden ejecutar en programas de química cuántica comerciales o de código abierto, siempre que el software incluya las aproximaciones de funcionales de intercambio-correlación (*xc*) más comunes. En el desarrollo de este cuaderno de laboratorio, usamos principalmente Gaussian, pero cada experiencia es fácilmente transferible a otros programas de química cuántica. La elección de utilizar una interfaz gráfica de usuario (GUI) o no, se deja al lector. Este cuaderno también está estructurado para que pueda ser utilizado como apoyo a una clase de Química Computacional. Idealmente, cubre un semestre completo (12 a 14 semanas). Los instructores deberán preparar a los estudiantes para que ejecuten cálculos utilizando un software que pueda manejar la teoría funcional de la densidad de Kohn-Sham (KS-DFT). Las experiencias individuales pueden separarse entre sí y distribuirse a los estudiantes semana a semana. Cada experiencia también incluye todas las referencias necesarias, mientras que al final del propio cuaderno también se informa una sección de referencia general con todos los artículos citados.

Todos los materiales en electrónico están disponibles bajo licencia GNU-GPL v3.0 en la siguiente dirección:

https://github.com/peverati/Devil_DFT_Tutorial_IJQC_2020

Consejos generales:

1) Todos los cálculos se realizan en fase gaseosa. No se utilizan modelos de resolución implícitos (o explícitos).
2) Todas las energías de referencia son energías electrónicas y se dan en kcal mol^{-1} (a menos que se indique lo contrario). Los cálculos de termoquímica solo son necesarios en Experiencia 9.
3) Todas las geometrías se dan en una carpeta separada.
4) Cada cálculo se puede realizar en una computadora de escritorio y no debería tomar mucho tiempo, incluso con la base más grande establecida.
5) Siéntase libre de probar más funciones, pero no olvide recopilar y presentar los resultados en consecuencia.

Consejos específicos para estudiantes:

1) Siga siempre las instrucciones dadas por su instructor.
2) Se recomienda leer los artículos sugeridos en la Experiencia 0 antes de realizar las demás experiencias.
3) NO elimine ningún archivo hasta el final de la práctica de laboratorio. Trate de mantenerlos organizados para que pueda acceder a ellos en cualquier momento.
4) Si utiliza una GUI, lea atentamente el manual.

Experiencia 0: ¿Dónde puedo buscar más sobre la DFT y sus funcionales?

El diseño de una experiencia computacional debe comenzar con la identificación del problema químico que se desea estudiar. Luego, debe leer la literatura publicada [5–13] para identificar el método de mejor rendimiento para el problema en cuestión, y solo después de una investigación bibliográfica exhaustiva puede comenzar a enviar cálculos y experimentar con el análisis de los resultados. La siguiente decisión que debe tomar al establecer una buena investigación computacional es la elección de un conjunto de funcionales y bases de intercambio-correlación (xc). Usualmente usamos el término "método" para indicar una combinación específica del conjunto funcional y base xc.

En general, elegir un buen método no es fácil, y es por eso por lo que debe confiar en las sugerencias provenientes de expertos en el campo [1–13]. Los funcionales que utilizará en este cuaderno representan sugerencias con fines educativos, y no estamos fomentando implícitamente su uso en un contexto general de investigación. Como se indicó al comienzo de este párrafo, lo remitimos a otras publicaciones [2–13] para obtener tales sugerencias. Para buenas revisiones de DFT, puede leer las referencias [3, 11, 12 o 13].

Antes de comenzar con las experiencias de este cuaderno, debes conocer las siglas más utilizadas en la literatura de la DFT. Como ejemplo, debe estar familiarizado con la jerga común que se utiliza en las "aproximaciones de los funcionales de la densidad de la escalera de Jacob" presentada por Perdew y Schmidt en 2001 [14]. La Aproximación de la densidad local de spin (LSDA), la aproximación del gradiente generalizado (GGA), GGA (mGGA), y los funcionales híbridos y doble híbridos, no deberían ser un concepto nuevo para usted. Si es así, use la referencia a continuación para revisar estos conceptos antes de pasar a la siguiente experiencia. Verifique su conocimiento usando el siguiente problema.

Problema: intente clasificar los siguientes funcionales como LSDA, GGA, mGGA, GGA híbrido (H-GGA) o mGGA híbrido (H-mGGA). Consulte la literatura [2–13,15] si necesita ayuda para identificar algunos funcionales. Sugerencia: no trataremos con híbridos dobles, por lo que no hay ninguno en esta tabla. Para las referencias apropiadas, vea las siguientes experiencias.

Funcional	B3LYP	HSEH1PBE	THCTHHYB	APFD
Forma				
Funcional	PBEPBE	B3PW91	BVP86	M11
Forma				

Experiencia 1: Aplicando la DFT a la química orgánica– Parte A.

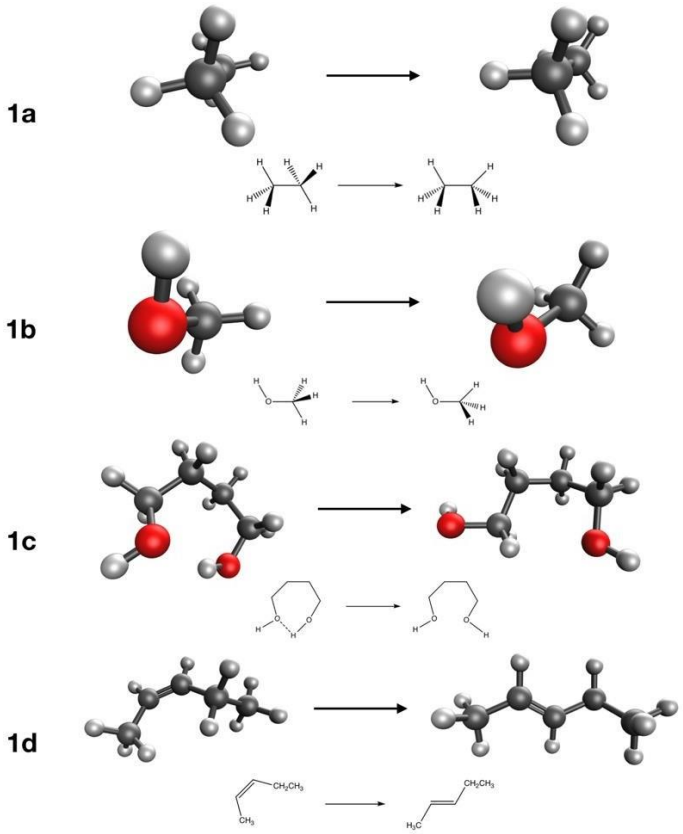


Figura 1 Reacciones estudiadas en la Experiencia 1. Barrera rotacional de etano (1a) y metanol (1b). Cambio conformacional en dos isómeros de butano-1,4-diol (1c). Reacción de isomerización cis-trans del pent-2-eno (1d). Los átomos de carbono son negros, los átomos de oxígeno son rojos y los átomos de hidrógeno son blancos.

En esta primera experiencia, analizarás cuatro reacciones que suelen encontrarse en los libros de texto de introducción a la química orgánica (Figura 1). Para la primera reacción (1a), calculará la diferencia de energía entre las conformaciones escalonadas y eclipsadas del etano. En la segunda reacción (1b), analizará la conformación escalonada y eclipsada en metanol. Para la tercera reacción (1c), observará un cambio de conformación en la molécula de butano-1,4-diol. La última reacción (1d) es la reacción de isomerización de cis-pent-2-eno a trans-pent-2-eno. Las moléculas en 1a-d provienen de los subconjuntos BHROT27 [2], BUT14DIOL [16] y FH51 [17,18] de la base de datos GMTKN55 [2].

Los funcionales que aproximan el termino de xc que sugerimos usar son: BVP86 [19-21], B3LYP [22,23,26–28], B3PW91 [20,22,24,27,29], PBE [30], HSEH1PBE [30], APFD [31,32], M11 [34], THCTHHYB, Hartree-Fock (HF) [39-42] y la teoría de perturbaciones

de segundo orden de Møller-Plesset (MP2) [43]. El conjunto base que debe usar es CC-QZVP [44]. También recomendamos la cuadrícula Ultrafine (poner comando de gaussian) en Gaussian, que corresponde a la cuadrícula de Lebedev (99,590) en Q-Chem.

Las energías de referencia para las reacciones son 2,73 kcal mol^{-1} (1a), 1,01 kcal mol^{-1} (1b), 3,29 kcal mol^{-1} (1c) y –1,15 kcal mol^{-1} (1d). Después de enviar los cálculos con cada uno de los funcionales presentados anteriormente, debe calcular las energías de reacción como la diferencia entre las energías de los productos y los reactivos y luego ponerlas en una tabla. Después de eso, también debe calcular el error absoluto |e| con respecto a los datos de referencia como la diferencia:

$$|e| = |E_{comp} - E_{ref}|$$

Reportalos en la misma tabla donde muestras los resultados.

Problemas:

1) ¿Qué funcional tiene el error más negativo??

---

2) ¿Qué funcional tiene el error más positivo?

---

3) ¿Qué funcional es el mejor ejecutante para cada reacción?

---

4) ¿Qué funcional es el mejor en general?

---

5) ¿Hay funcionales que funcionan mal para estas reacciones?

---

6) Recopile el tiempo computacional total y divídalo por el número de pasos SCF que se necesitan para alcanzar la convergencia para cada funcional. ¿Cómo se desempeñan? Reporte los resultados en la siguiente tabla. Compare B3LYP con los otros funcionales híbridos y el método HF (agregue más páginas si es necesario para informar todos los resultados).

	B3LYP	HSEH1PBE	THCTHHYB	APFD
Tiempo total				
	PBEPBE	B3PW91	BVP86	M11
Tiempo total				

Comentaremos más extensamente los resultados una vez finalizada la Experiencia 3.

Experiencia 2: aplicando la DFT a la química orgánica– Parte B.

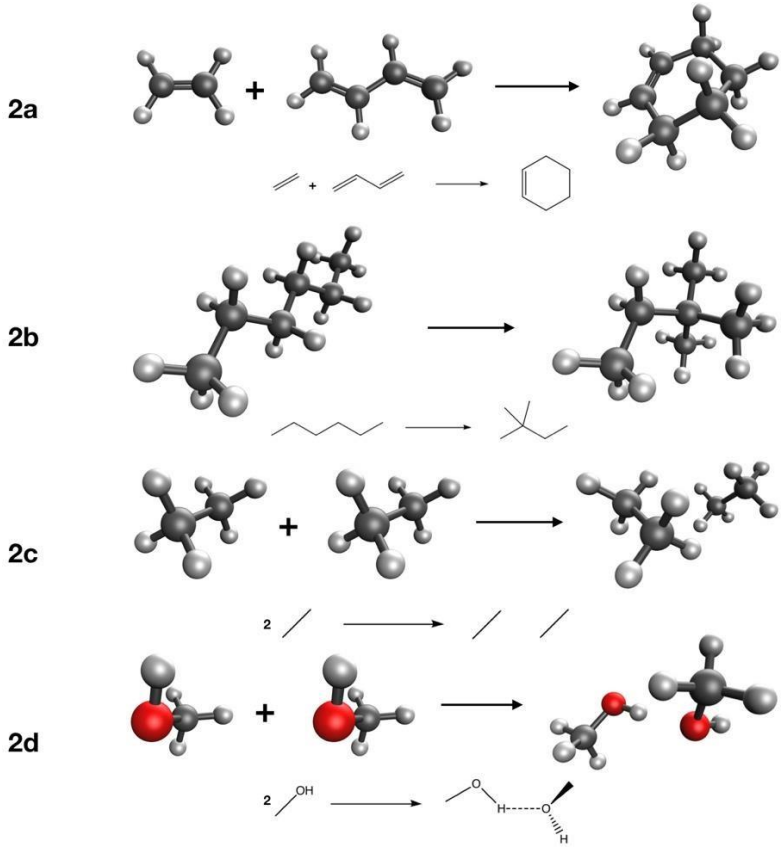


Figura 2 Reacciones estudiadas en la Experiencia 2. Reacción Diels-Alder de eteno y butadieno (2a). Isomerización de n-hexano a neohexano (2b). Formación de los dímeros de eteno (2c) y metanol (2d). Los átomos de carbono son negros, los átomos de oxígeno son rojos y los átomos de hidrógeno son blancos. La línea punteada muestra el enlace de hidrógeno.

En esta segunda experiencia, tratará con cuatro reacciones orgánicas adicionales que se encuentran comúnmente en los libros de texto (Figura 2). La primera (2a) es una reacción de Diels-Alder entre eteno y butadieno y tiene una energía de referencia de -45,40 kcal mol^{-1}. El segundo (2b) es la reacción de isomerización del neohexano (2,2-dimetilbutano) a hexano con una energía de referencia de –2,49 kcal mol^{-1}. El tercero describe la formación del dímero de etano (2c), que se mantiene unido únicamente por interacciones de van der Waals (dispersión), y tiene una energía de referencia de –1,34 kcal mol^{-1}. En el último, analizarás el dímero de metanol (2d), que se mantiene unido por enlaces de hidrógeno, con una energía de referencia de –5,81 kcal mol^{-1}. Estas estructuras provienen de los subconjuntos DARC5 [45,46], y ADIM [49,47], de la base de

datos GMTKN55 [2] y los subconjuntos AlkIsomer11 [47,48] y S66 [49,50] de la base de datos MGCDB84 [3].
De manera similar a la Experiencia 1, utilizará las siguientes funcionales:
BVP86 [19-21], B3LYP [22,23,26–28], B3PW91 [20,22,24,27,29], PBE [30], HSEH1PBE [30], APFD [31,32], M11 [34], THCTHHYB, y los métodos Hartree-Fock (HF) [39-42], y MP2 [43]. El conjunto básico que debe utilizar es def2-QZVP [44]. También recomendamos la cuadrícula Ultrafine en Gaussian, que corresponde a la cuadrícula de Lebedev (99,590) en Q-Chem. Después de enviar los cálculos con cada uno de estos funcionales, debe calcular las energías de reacción calculadas y luego colocarlas en una tabla junto con el error absoluto|e|.

Problemas:

1) ¿Qué funcional tiene el error más negativo?

---

2) ¿Qué funcional tiene el error más positivo?

---

3) ¿Qué funcional es el mejor ejecutante para cada reacción?

---

4) ¿Qué funcional es el mejor en general?

---

5) ¿Hay funcionales que funcionan mal para estas reacciones?

---

6) Recopile el tiempo computacional total y divídalo por el número de pasos SCF que se necesitan para alcanzar la convergencia para cada funcional. ¿Cómo se desempeñan? Reporte los resultados en la siguiente tabla. Compare B3LYP con los otros funcionales híbridos y el método HF (agregue más páginas si es necesario para informar todos los resultados).

	B3LYP	HSEH1PBE	THCTHHYB	APFD
Tiempo total				
	PBEPBE	B3PW91	BVP86	M11

Tiempo total				

7) ¿Los funcionales que incluyen correcciones -D3 funcionan mejor que los correspondientes sin corregir?

Experiencia 3: Casos difíciles para la DFT.

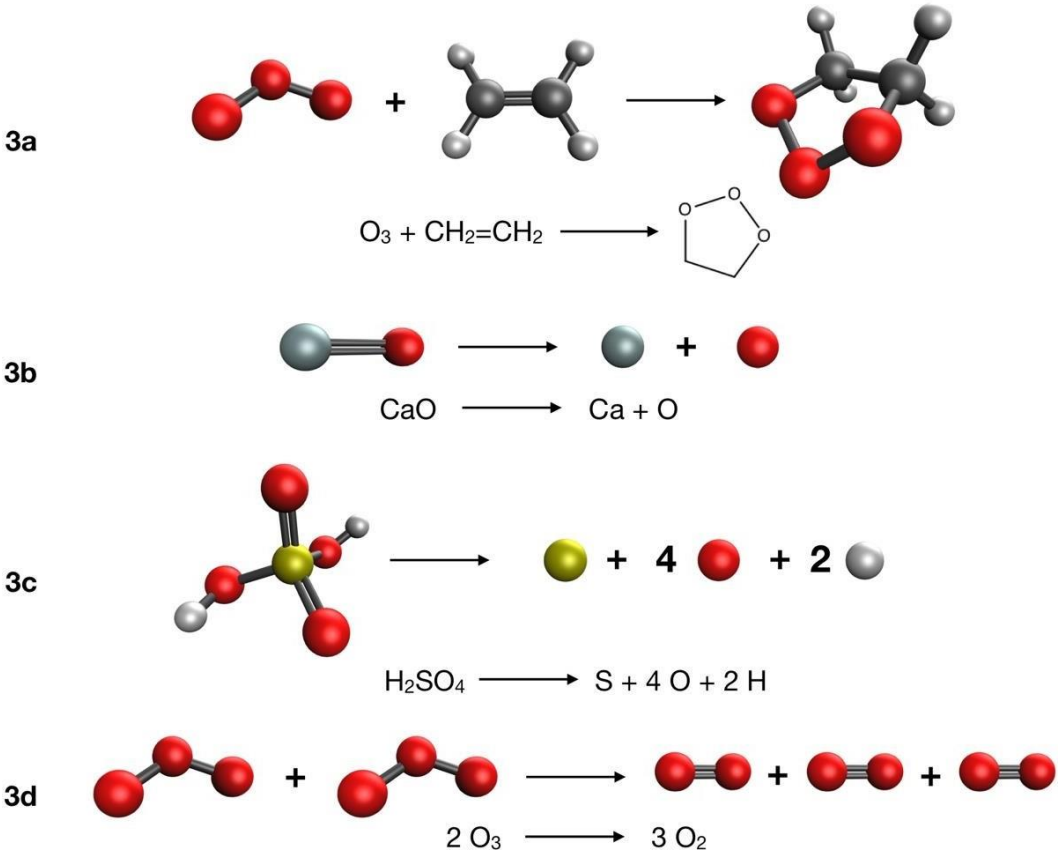


Figura 3 Reacciones estudiadas en la Experiencia 3. La reacción de ozonólisis (3a), la disociación del enlace CaO (3b), la reacción de atomización del ácido sulfúrico (3c) y la conversión de ozono a oxígeno (3d). Los átomos de carbono son negros, los átomos de oxígeno son rojos, los átomos de hidrógeno son blancos, el átomo de azufre es amarillo oscuro y el átomo de calcio es gris.

En la Experiencia 2, encontramos que es posible mejorar el error en las energías de reacción usando la corrección D3 para las interacciones de dispersión. Desafortunadamente, la corrección D3 y todos los demás esquemas diseñados para tener en cuenta la dispersión no son una solución universal cuando se trata de problemas más complicados.

En esta última experiencia sobre funcionales, tratarás con cuatro reacciones. El primero (3a en la figura 3) es la reacción entre eteno y ozono para formar el primer intermedio en una reacción de ozonólisis como se encuentra en muchos libros de texto. El segundo (3b) es la energía de disociación del enlace homolítico de CaO en calcio atómico y oxígeno. El tercero (3c) es la energía de atomización total del ácido sulfúrico. El último (3d) es la energía de reacción de la conversión entre el ozono y el oxígeno molecular. La primera reacción proviene del subconjunto DC13 [2,33,45,51–60] de la base de datos GMTKN55 [2], mientras que la segunda y la tercera provienen de los subconjuntos MR-

MGM-BE4 [61] y DC9 [12,62] de la base de datos Minnesota 2015B [6]. La última reacción se ha obtenido combinando dos puntos de la base de datos W4-17 [63]. Las energías de referencia son –58,7 kcal mol^{-1}, 96,15 kcal mol^{-1}, 602,18 kcal mol^{-1} y –67,60 kcal mol^{-1}, respectivamente.
Debería volver a utilizar los siguientes funcionales: BVP86 [19-21], B3LYP [22,23,26–28], B3PW91 [20,22,24,27,29], PBE [30], HSEH1PBE [30], APFD [31,32], M11 [34], THCTHHYB, Hartree-Fock (HF) [39-42] y MP2 [43]. El conjunto base que debe usar es def2-QZVP [44], como siempre. Nuevamente, también recomendamos la cuadrícula ultrafina en Gaussian, que corresponde a la cuadrícula de Lebedev (99,590) en Q-Chem. Después de enviar los cálculos con cada uno de estos funcionales, debe calcular las energías de reacción y luego colocarlas en una tabla junto con el error absoluto|e|.

Problemas:

1) ¿Qué funcional tiene el error más negativo?

---

2) ¿Qué funcional tiene el error más positivo?

---

3) ¿Qué funcional es el mejor ejecutante para cada reacción?

---

4) ¿Qué funcional es el mejor en general?

---

5) ¿Hay funcionales que funcionan mal para estas reacciones?

---

6) Recopile el tiempo computacional total y divídalo por el número de pasos SCF que se necesitan para alcanzar la convergencia para cada funcional. ¿Cómo se desempeñan? Reporte los resultados en la siguiente tabla. Compare B3LYP con los otros funcionales híbridos y el método HF (agregue más páginas si es necesario para reportar todos los resultados).

	B3LYP	HSEH1PBE	THCTHHYB	APFD
Tiempo total				
	PBEPBE	B3PW91	BVP86	M11

Tiempo total				

6) ¿Los funcionales con corrección D3 funcionan mejor o peor que los respectivos no corregidos?

Problemas generales para las experiencias 1 a 3:

1) ¿Qué funcional o método tiene el mejor desempeño general?

2) ¿Qué funcional o método tiene el peor desempeño general?

3) ¿B3LYP es más rápido que otros funcionales/métodos?

4) ¿Crees que agregar la corrección D3 ayuda? ¿Para cual reacciones?

Consejo: Las reacciones estudiadas en las experiencias anteriores se pueden encontrar en los libros de texto de química orgánica e involucran moléculas orgánicas muy simples. Sin embargo, incluso en estos casos conceptualmente simples, vimos que los resultados son impredecibles y diferentes funcionales se comportan de diferentes maneras. Vale la pena señalar que, si un funcional falla, no es una falla de la DFT en su conjunto. Es un fallo de la aproximación en uso. La DFT es, en principio, exacta, pero tenemos que basarnos en aproximaciones a la teoría exacta. Los resultados que obtenemos dependen de qué tan buenas sean las aproximaciones (funcionales xc).

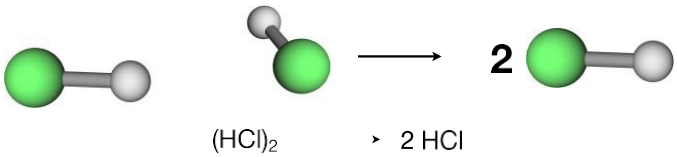
Comentario práctico: en el primer grupo de reacciones “fáciles”, es posible que te hayas dado cuenta de que casi todas las funciones dan buenos resultados. Después de todo, para cada reacción estamos comparando energías electrónicas de moléculas muy similares, ya que su conformación, o la configuración en el caso del pent-2-eno, es lo único que está cambiando. Su densidad de electrones exhibe muy pocos cambios, por lo que incluso si un funcional obtiene la densidad total fundamentalmente incorrecta en ciertas regiones, aún puede proporcionar buenos resultados, siempre que sea consistente en la descripción de los cambios. Es decir, los errores en las regiones de la densidad que no están sujetas a cambios se anulan al calcular la diferencia entre el producto y los reactivos. Esto se conoce como “cancelación de errores”. Sin embargo, las cancelaciones de errores no son fáciles de generalizar. Por lo tanto, no podemos decir que las rotaciones alrededor de un enlace simple, los cambios conformacionales o las isomerizaciones cis-trans sean fáciles de estudiar computacionalmente. De hecho, no hay forma de saber de antemano si la reacción en cuestión cae en la categoría fácil, media o difícil. Antes de iniciar una investigación computacional, debemos analizar con gran detalle los subconjuntos de las bases de datos que incluyen moléculas (o propiedades químicas) cercanas a las que queremos estudiar.

Para las reacciones del segundo grupo, necesitamos un buen funcional capaz de obtener una densidad electrónica correcta. Dado que los reactivos y los productos son muy diferentes entre sí en general, no podemos confiar en la beneficiosa cancelación de errores que ocurrió en los casos más simples. Por ejemplo, en el caso de la reacción de Diels-Alder, combinamos dos fragmentos para obtener una molécula más grande. Lo mismo es válido para las reacciones de isomerización, que son muy difíciles para la DFT. Las otras dos reacciones tratan con dímeros de moléculas simples que se mantienen unidas por interacciones de van der Waals. En la década de 1990, se reconoció que muchos funcionales de xc no logran describir la unión en la región de van der Waals [71–73]. En 2012, modelar correctamente las interacciones de van der Waals todavía figuraba como un desafío para la DFT [32,42,75–78], pero las dos correcciones más utilizadas disponibles en la actualidad son la corrección D de Grimme, Goerigk y colaboradores

[29,67,68], y el funcional VV10 de Vydrov y Van Voorhis [36]. Como habrás notado, la inclusión de la corrección D3 para B3LYP y PBE mejoró los resultados del funcional no corregido. Ciertamente es una buena idea usar la corrección D3 cuando sea posible, principalmente al realizar optimizaciones de geometría [71], pero también debemos tener en cuenta que a veces puede resultar en una sobrecorrección. Las aproximaciones incluyen la corrección VV10 y se encuentran entre los funcionales de mejor rendimiento en general. Se han propuesto correcciones D3 para los funcionales de Minnesota [2,72,73], pero dado que los funcionales se han parametrizado también mediante complejos de van der Waals, en principio no son necesarias.

El tercer grupo de reacciones es el que arrojó los peores resultados. Para este grupo no existe ningún funcional que funcione satisfactoriamente, salvo excepciones aleatorias (M11 funciona bien para CaO porque estaba incluido en su conjunto de entrenamiento, es decir, M11 fue parametrizado usando la base de datos que incluye esta molécula). Si lo pensamos bien, esas reacciones no parecen complicadas desde el punto de vista químico. De hecho, las dos moléculas más grandes tienen solo cinco átomos que son más grandes que el hidrógeno. Por eso no podemos tratar los cálculos a ciegas. Nuestra sugerencia es buscar siempre el consejo de un experto cuando diferentes funcionales proporcionen resultados muy discordantes. En un editorial reciente sobre organometálicos [74], Kathrin Hopmann habla sobre lo que hace que un artículo computacional sea interesante y dice: "No use DFT para mecanismos que no puede manejar". Esto resume muy bien el mensaje final de estas tres primeras experiencias. Todos los funcionales que aparecieron en la literatura durante los últimos 40 años tienen fortalezas y debilidades. Gran parte del esfuerzo en el desarrollo funcional se ha dedicado a ampliar los primeros y reducir los segundos. Como resultado, las aproximaciones al término de xc con funcionales están mejorando estadísticamente, lo que significa que, en general, los resultados obtenidos con los funcionales modernos son mejores que los obtenidos con los funcionales más antiguos. Pero siempre depende de los investigadores usar el juicio al elegir entre ellos.

Experiencia 4: Error de conjunto incompleto de base.



Esquema 1 La reacción estudiada en las Experiencias 4 y 5. Los átomos de hidrógeno son blancos, mientras que los átomos de cloro son verdes.

Parte A: Valencia dividida.

Después de seleccionar un funcional, el siguiente paso por elegir es un buen conjunto de bases. Hay muchas “familias” diferentes de conjuntos básicos [75,76], que han sido desarrollados en los últimos sesenta años por diferentes químicos computacionales. Head-Gordon y colaboradores [86] encontraron que los conjuntos base más rentables para DFT pertenecen a la familia Ahlrichs [44]. Analizaremos su desempeño para la reacción de disociación del dímero HCl, proveniente del subconjunto NCCE23 [12,77–80], de la base de datos Minnesota 2015B [6]. La energía de referencia para esta reacción de disociación es de 2,01 kcal mol^{-1}.

Sugerimos calcular la energía de disociación de esta reacción utilizando los siguientes funcionales: BVP86 [19-21], B3LYP [22,23,26–28], B3PW91 [20,22,24,27,29], PBE [30], HSEH1PBE [30], APFD [31,32], M11 [34], THCTHHYB y los métodos Hartree-Fock (HF) [39–42] y MP2 [43]. Por ahora, solo usamos los conjuntos básicos def2-SVP, def2-TZVP y def2-QZVP [44]. Estos conjuntos básicos se han redefinido recientemente, de ahí el prefijo "def2". SVP significa Split-Valence Polarized, TZVP significa Triple-Zeta (split-)Valence Polarized, y QZVP significa Quadruple-Zeta (split-)Valence Polarized. Lo de valencia dividida implica que incluyen un conjunto de funciones para los orbitales centrales, mientras que el número de funciones para los orbitales de valencia depende del tamaño del conjunto base, es decir, de la "zeta". Por ejemplo, los conjuntos de bases de doble ζ incluyen dos funciones para cada orbital de valencia, los de triple ζ incluyen tres, y así sucesivamente. El termino polarizado significa que el conjunto base incluye una función adicional que tiene un momento angular más alto que los orbitales de valencia. Por ejemplo, un conjunto de bases polarizadas de doble ζ para el carbono incluiría una función *d*, mientras que para el hierro incluiría una función *f*. Si algunos de los conjuntos de base def2 no están definidos para el programa que está utilizando (por ejemplo, los conjuntos de base aumentados no están definidos como una palabra clave en el programa gaussian), puede usarlos como conjunto de base externo o usar el conjunto de bases de Dunning [81–87]. Sin embargo, tenga en cuenta que están optimizados para cálculos de funciones de onda correlacionadas [81–87], y podrían no ser la mejor opción para DFT en términos de eficiencia [77]. Para estos conjuntos básicos, la notación abreviada es cc-pVnZ, donde cc significa “correlación consistente” y *n* es cualquier de los siguientes: D para ζ doble, T para ζ triple o Q para ζ cuádruple. Por lo tanto, los conjuntos básicos def2-SVP, def2-TZVP y def2-QZVP pueden sustituirse por cc-pVDZ, cc-pVTZ y cc-pVQZ respectivamente. Una vez más, sugerimos la cuadrícula UltraFine en Gaussian, o la (99,590) en Q-Chem. Después de enviar los cálculos con

cada una de estas combinaciones funcionales/conjunto base, debe calcular la energía de reacción y luego el error absoluto |e|. Reporte los errores absolutos en la siguiente tabla y luego responda las siguientes preguntas.

	B3LYP	HSEH1PBE	THCTHHYB	APFD
def2-SVP				
def2-TZVP				
def2-QZVP				
	PBEPBE	B3PW91	BVP86	M11
def2-SVP				
def2-TZVP				
def2-QZVP				

Problemas:

1) ¿Existe una tendencia al pasar de un conjunto de base doble a cuádruple? Más específicamente, ¿qué sucede con la magnitud del error? ¿Todos los funcionales siguen el mismo patrón?

---

2) ¿Cómo clasificaría el comportamiento de los funcionales que probó? Utilice una de estas definiciones para completar la siguiente tabla.

a) Funcionales cuyos errores disminuyen al aumentar el tamaño del conjunto base.

b) Funcionales cuyos errores aumentan al aumentar el tamaño del conjunto base.

c) Funcionales cuyo comportamiento no sigue un patrón específico.

	B3LYP	HSEH1PBE	THCTHHYB	APFD
Comportamiento				
	PBEPBE	B3PW91	BVP86	M11
Comportamiento				

En tu opinión, ¿qué funcionales son los mejores para esta reacción?

Parte B: Funciones Difusas.
Agregar funciones de base difusa es importante para describir el comportamiento de la densidad electrónica en regiones que están alejadas de los núcleos. Por ello, los exponentes de una función difusa son más pequeños que los que se utilizan en el resto de funciones.
Repita los cálculos para el dímero de ácido clorhídrico con los conjuntos básicos def2-SVPD, def2-TZVPD y def2-QZVPD50 (o los conjuntos básicos aug-cc-pVDZ, aug-cc-pVTZ y aug-cc-pVQZ) [81 –87]. Reporte los errores absolutos en la siguiente tabla y luego responda las siguientes preguntas.

	B3LYP	HSEH1PBE	THCTHHYB	APFD
def2-SVP				
def2-TZVP				
def2-QZVP				
	PBEPBE	B3PW91	BVP86	M11
def2-SVP				
def2-TZVP				
def2-QZVP				

Problemas:

1) ¿Ve tendencias similares a las del punto 1 de la Experiencia 4a?

2) ¿Crees que agregar funciones difusas ayuda para este sistema en particular?

3) ¿Para qué especies químicas crees que necesitamos más funciones difusas?

Sugerencia: Al realizar cálculos, los errores surgen de diferentes fuentes. En este punto, es posible que ya te hayas dado cuenta de que cada conjunto básico conlleva un error debido al hecho de que es una expansión truncada. Este error se llama "Error de incompletitud del conjunto de bases" o BSIE (por sus siglas en inglés). Para minimizar el impacto que el error BSIE tiene en los resultados finales, debe realizar los cálculos con el conjunto de bases más grande que pueda permitirse. Otra forma de expresar esta idea es decir que desea minimizar el error asociado con el conjunto base, de modo que el error que surge de la funcional xc sea siempre el mayor.

Experiencia 5: Error de superposición de conjunto de bases.

El error BSIE no es el único que afecta a los cálculos con base pequeña. Cuando se estudian sistemas ligados a través de interacciones no covalentes, surge otro error llamado “Error de Superposición de Conjunto de Bases”, o BSSE. La razón por la que los resultados pueden verse afectados por el BSSE es simple y puede explicarse por el hecho de que el cálculo en el sistema acotado puede beneficiarse de un mayor número de funciones de base que el cálculo en los dos monómeros. Esto sucede porque en el sistema ligado las funciones de base del monómero A pueden describir parcialmente al monómero B y viceversa. Obviamente, estas contribuciones adicionales están ausentes en los cálculos separados que se realizan sobre los monómeros. Como señalaron van Duijneveldt y colaboradores [98], esto es problemático solo cuando se calculan las energías de enlace, mientras que podría ser incluso beneficioso si solo interesa la energía del sistema enlazado.

Un método aproximado para resolver este problema es el método de corrección de contrapeso (CP) de Boys y Bernardi [89]. En este esquema, se realizan cálculos ligeramente modificados para calcular las energías de los monómeros, en los que se incluyen las funciones base del otro monómero, pero sin sus átomos [90]. Las llamadas “funciones fantasma” de las que se beneficia el dímero, ahora se contabilizan específicamente en los cálculos de ambos monómeros. Es importante notar que este método implica una elección de cómo se divide el complejo en diferentes fragmentos. Esto representa una limitación importante, ya que la forma en que podemos dividir un complejo en fragmentos no es única, especialmente cuando se trata de BSSE intramolecular [91] Extremando este procedimiento, y considerando cada átomo como un fragmento, se llega a métodos como el CP^{aa} de Galano y Alvarez-Idaboy [92] o la corrección ACP(x) de Jensen [93]. Kruse y Grimme, en cambio, introdujeron el contrapeso geométrico (gCP) [91], como una forma sencilla de corregir tanto el error BSSE intermolecular como intramolecular utilizando solo la geometría de la molécula. No exploraremos estos métodos, pero nos limitaremos al método CP original de Boys y Bernardi.

Cuando se trata de BSSE intermolecular, como en el caso del dímero HCl también utilizado para la experiencia anterior [6,12,77–80], se sugiere una forma sencilla de dividir el dímero en dos monómeros. Aplicará la corrección CP a este sistema usando todos los funcionales [5, 19–28, 30–33, 34–43], y conjuntos de bases [44, 81–87] que ya usó en las Experiencias 4a y 4b. En cuanto a las Experiencias 4a y 4b, si algunos de los conjuntos de bases def2 no están definidos, utilice los conjuntos de bases de Dunning en su lugar [81–87].

Problemas:

1) Una vez que pones los resultados en la siguiente tabla, ¿ves una mejora en los resultados? Compáralas con las tablas de las experiencias anteriores.

CP corregido	B3LYP	HSEH1PBE	THCTHHYB	APFD
def2-SVP				
def2-TZVP				
def2-QZVP				
CP corregido	PBEPBE	B3PW91	BVP86	M11
def2-SVP				

def2-TZVP				
def2-QZVP				
CP corregido	B3LYP	HSEH1PBE	THCTHHYB	APFD
def2-SVPD				
def2-TZVPD				
def2-QZVPD				
CP corregido	PBEPBE	B3PW91	BVP86	M11
def2-SVPD				
def2-TZVPD				
def2-QZVPD				

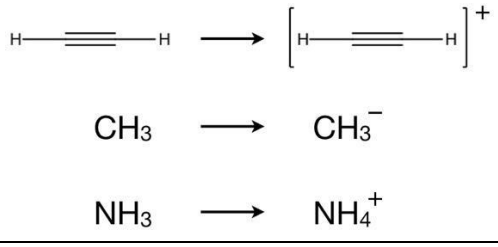
2) Si nota una mejoría, ¿dónde está? En otras palabras, qué conjuntos de bases se benefician más de la corrección CP?

3) ¿Hay algún funcional que no se beneficie en absoluto de la corrección de CP?

4) ¿Crees que sería una buena idea agregar también la corrección -D3 junto con la corrección CP? Antes de responder a esta pregunta, vuelva a ejecutar sus cálculos con B3LYP-D3(BJ) y PBE-D3(BJ).

Sugerencia: si responde afirmativamente a la pregunta 4, entonces debe saber que hay algunos casos en los que la validez del CP aún está en debate en la comunidad de química computacional [88,92-98]. La simple corrección CP en algunos casos sobre corrige los resultados [92,93]. Por lo tanto, no debe suponer que la corrección CP resuelve todos sus problemas, pero seguramente lo hace para conjuntos de base pequeños, para los cuales siempre debe usarse. En general, sin embargo, una mejor estrategia es usar un conjunto base que esté cerca del límite del conjunto base, ya que tanto BSSE como BSIE se minimizarán. Grimme, Goerigk y colaboradores [71], así como Mardirossian y Head-Gordon [37] han utilizado conjuntos de bases polarizadas cuádruples ζ exactamente para este propósito.

Experiencia 6: Revisión de Conjuntos de Bases Modernas.



Esquema 2 Reacciones estudiadas en la Experiencia 6.

Ahora que está familiarizado con lo que son los conjuntos de bases, puede aplicar este conocimiento a las reacciones químicas. En concreto, vas a analizar el efecto de sumar funciones difusas al estudiar la afinidad protónica del amoníaco, el potencial de ionización del acetileno y la afinidad electrónica del radical CH_3. A pesar de su pequeño tamaño, estas especies aún pueden usarse para demostrar buenas prácticas computacionales en el análisis de aniones y cationes. La afinidad protónica del amoníaco proviene del subconjunto PA8 [12,100] de la base de datos Minnesota 2015B [6], mientras que la reacción del acetileno proviene del subconjunto G21IP [101] y la afinidad electrónica del radical metilo proviene del subconjunto G21EA [101] de la base de datos GMTKN55 [2]. Los valores de referencia son 211,90 kcal mol^{-1} para la afinidad protónica del NH_3, 264,60 kcal mol^{-1} para el potencial de ionización del acetileno y 1,20 kcal mol^{-1} para la afinidad electrónica del radical metilo.

Los funcionales que usarás son los habituales: BVP86 [19-21], B3LYP [22,23,26–28], B3PW91 [20,22,24,27,29], PBE [30], HSEH1PBE [30], APFD [31,32], M11 [34], THCTHHYB y los métodos Hartree-Fock (HF) [39–42] y MP2 [43]. Para esta experiencia, utilizará los conjuntos básicos de Ahlrichs [44], o alternativamente los de Dunning [81–87]. Debe ejecutar los cálculos con def2-SVP, def2-SVPD, def2-TZVP, def2-TZVPD, def2-QZVP y def2-QZVPD (cc-pVDZ, aug-cc-pVDZ, cc-pVTZ, aug-cc-pVTZ, cc-pVQZ y aug-cc-pVQZ). Como de costumbre, recopile todos los resultados, es decir, los errores sin firmar, en las tablas siguientes.

IP of HCCH	def2-SVP	def2-SVPD	def2-TZVP	def2-TZVPD	def2-QZVP	def2-QZVPD
B3LYP						
HSEH1PBE						
THCTHHYB						
APFD						
PBEPBE						
B3PW91						

BVP86						
M11						
HF						

EA of $CH_3^{\cdot}$	**def2-SVP**	**def2-SVPD**	**def2-TZVP**	**def2-TZVPD**	**def2-QZVP**	**def2-QZVPD**
B3LYP						
HSEH1PBE						
THCTHHYB						
APFD						
PBEPBE						
B3PW91						
BVP86						
M11						
HF						

PA of NH_3	**def2-SVP**	**def2-SVPD**	**def2-TZVP**	**def2-TZVPD**	**def2-QZVP**	**def2-QZVPD**
B3LYP						
HSEH1PBE						
THCTHHYB						
APFD						
PBEPBE						
B3PW91						
BVP86						
M11						

HF						

Problemas:

1) ¿Qué funcional es la mejor opción con la familia de conjunto de base def2-nZVP?
   - Potencial de ionización del acetileno: _____
   - Afinidad electrónica de CH_3: _____
   - Afinidad protónica del amoníaco: _____

2) Compare los resultados informados anteriormente con los obtenidos con los conjuntos de base def2-nZVPD. ¿Mejoran o empeoran los resultados, en promedio?
   - Potencial de ionización del acetileno: _____
   - Afinidad electrónica de CH_3: _____
   - Afinidad protónica del amoníaco: _____

3) Con base en los resultados que recopilaste anteriormente, ¿crees que necesitamos funciones difusas cuando estudiamos cationes? ¿Qué pasa con los aniones?

_____

_____

4) Regrese a la Experiencia 4b, pregunta 3. ¿Qué respuesta dio allí?

_____

5) Los resultados obtenidos con pequeños conjuntos básicos para cationes se benefician de la adición de funciones difusas. Esto parece estar en contradicción con los puntos/experiencias anteriores. ¿De dónde crees que viene esta reducción de errores?

_____

_____

Sugerencia: el mensaje final de estas primeras seis experiencias es que debe confiar en la literatura para mantenerse actualizado con el desarrollo reciente del método en la DFT. Se garantiza que los funcionales modernos como los optimizados por Mardirossian y Head-Gordon [35-37] o los últimos funcionales de Minnesota como MN15 [6] son robustos y altamente transferibles, y se puede confiar en sus resultados en un amplio espectro de propiedades químicas. Cuando se combinan con un conjunto de base grande o un conjunto de base pequeño y la corrección de CP, ofrecen buenos resultados para varios sistemas. La elección de un conjunto base tampoco debe subestimarse. Según nuestra experiencia, recomendamos conjuntos base que están diseñados para cálculos en la DFT, como los introducidos por Ahlrichs [44] o Jensen [102–106], sobre los conjuntos básicos Dunning [81–87] o Pople [107–115]. Los conjuntos de bases de Ahlrich suelen evitar los problemas que surgen al mezclar conjuntos de bases de diferentes familias o tamaños cuando se tratan elementos más pesados [116], ya que tienen la ventaja adicional de estar definidos para la mayor parte de la tabla periódica (todos los elementos hasta Rn).

Comentario avanzado: Nos limitamos a conjuntos de bases que se definen como palabra clave en la mayoría de los paquetes de software. Los profesionales más avanzados suelen consultar el sitio web Basis Set Exchange [117–120] si necesitan otros conjuntos de bases. Ya se ha dado cuenta de que, si aumenta el tamaño de un conjunto base, el tiempo computacional necesario para el cálculo también aumenta. Los conjuntos de bases aumentadas son más grandes (y más caros) que los no aumentados. Al elegir un conjunto base, debe tener en cuenta la viabilidad, la fiabilidad, el rendimiento y la precisión de los resultados.
Con el fin de maximizar el rendimiento computacional, Truhlar y sus colaboradores propusieron un "aumento mínimo" para los conjuntos de base de Ahlrichs, llamados conjuntos de base "mínimamente aumentados" [121], y diferentes "niveles" de aumento para los conjuntos de bases de Dunning, a los que llamaron may-, jun- y jul-cc-p(n+d)Z [122].

Experiencia 7: ¿Por qué es tan exitoso el nivel de teoría B3LYP/6-31G*?

El funcional B3LYP [22,23,26–28] y el conjunto de base 6-31G* de Pople y colaboradores [107–115] es el método estándar de facto en la comunidad de química orgánica. Este funcional se basa en el esquema híbrido de tres parámetros que introdujo Becke en 1993 [27]. El funcional de tres parámetros propuesto originalmente usó su funcional de intercambio de 1988 [26] junto con el funcional de correlación de Perdew y Wang (PW91) [20,21,28]. El funcional B3LYP se introdujo un año más tarde por Frisch y colaboradores al reemplazar PW91 con el funcional LYP [26] ya que este último proporcionó mejores resultados en un estudio computacional de espectros de dicroísmo circular. pues ambos funcionales son casi idénticos. El conjunto básico 6-31G* es un conjunto básico polarizado de valencia dividida de doble ζ, [13–115] y, como tal, contiene una cantidad razonablemente moderada de funciones básicas. Su uso con B3LYP se hizo popular en la década de 1990 y principios de los 2000 porque permitía realizar cálculos en moléculas relativamente grandes que son relevantes para la química orgánica, con rendimientos drásticamente superiores a los métodos disponibles anteriormente, como semiempíricos y Hartree-Fock. Sin embargo, a la luz del aumento de las capacidades computacionales de las computadoras en las últimas dos décadas, el uso de un conjunto de bases de doble ζ en un entorno de investigación a veces es peligroso, como también vimos en las tres experiencias anteriores. En esta experiencia, probará el funcional B3LYP con varios conjuntos de bases polarizadas de doble ζ para comprender por qué el nivel de teoría B3LYP/6-31G* es tan "especial". Es decir, los conjuntos básicos que sugerimos son: 6-31G*, cc-pVDZ [81–87], def2-SVP [44], y pc-1 [102–106] (si este no está disponible, use def2-SV(P), el más pequeño de los conjuntos de bases de Ahrlichs). Los sistemas en los que debe ejecutar los cálculos son los de Experiencia 1, 2 y 3. Luego debe reportar los resultados en una tabla, junto con los resultados obtenidos en las Experiencias 1–3 con el conjunto base def2 QZVP. Recuerde reportar junto con el valor calculado también el error con respecto a la referencia, como se muestra en la Experiencia 1.

Problemas:

1) Elige una de las moléculas. ¿Cuántas funciones de base tienen los diferentes conjuntos de bases?

Molécula	6-31G*	def2-SVP	cc-pVDZ	pc-1 or def2-SV(P)
Basis functions (#)				

2) ¿Qué conjunto de bases da el error absoluto más pequeño para las moléculas en la Experiencia 1?

---

3) ¿Qué conjunto de bases da el error absoluto más pequeño para las moléculas en la Experiencia 2?

4) ¿Qué conjunto de bases da el error absoluto más pequeño para las moléculas en la Experiencia 3?

5) ¿Qué conjunto de bases da el error absoluto más pequeño en general?

6) Ahora vuelva a hacer todos los cálculos usando el funcional B3LYP [22,23,26–28] con los mismos conjuntos de base pequeños. ¿Qué sucede con los resultados? ¿Mejoran?

7) Pruebe otro funcional con el conjunto básico 6-31G*. ¿Observas las mismas tendencias?

8) Compare ahora los resultados de B3LYP B3LYP-D3(BJ)/def2-QZVP con los resultados de B3LYP/6-31G*, y para B3LYP B3LYP-D3(BJ)/def2-QZVP y B3LYP/6-31G*. ¿Ves algo extraño?

Sugerencia: Esta experiencia confirma que el nivel de teoría B3LYP/6-31G* es tan bueno porque se beneficia de una cancelación de error fortuito [71]. La teoría sugiere que el error cometido al usar un funcional con un conjunto base más pequeño siempre debe ser mayor que el error cometido al usar uno más grande. El hecho de que B3LYP/6-31G* no cumpla con esta expectativa en comparación con B3LYP/def2-QZVP es preocupante, ya que subraya una cancelación fortuita de errores. Sin embargo, dicha cancelación de errores puede tener un comportamiento impredecible: a veces puede funcionar a su favor, como en las reacciones de la Experiencia 2, pero a veces puede funcionar en su contra, como en las reacciones

de la Experiencia 3. El 6-31G* conjunto básico no hace que el B3LYP funcione mejor, ni sea más sólido físicamente. En lugar de tratar de comprender qué problemas químicos se pueden resolver con B3LYP/6-31G*, sugerimos comenzar a alejarse de él y comenzar a usar un funcional más moderno con un comportamiento predecible con respecto al conjunto básico. Si esto no es posible debido a capacidades computacionales limitadas, entonces se deben implementar correcciones más apropiadas, como las sugeridas por Kruse, Goerigk y Grimme, además de contar con la ayuda de un experto. En 2012, Cohen, Mori-Sánchez y Yang dijeron que un rendimiento uniforme mejor que B3LYP seguía siendo un desafío para las aproximaciones de los funcionales al termino de xc [56]. Las revisiones recientes en el campo demostraron que esto ya no es cierto [3,11–13], y creemos que el verdadero desafío es deshacerse de B3LYP/6-31G* por completo. Como dijeron Kruse, Goerigk y Grimme, no esperamos que esto suceda pronto.

Experiencia 8: Mallas de integración 1: El dímero de argón.

Para esta experiencia, analizará un problema sutil que podría afectar potencialmente los resultados de cada funcional de intercambio-correlación. La DFT basada en los teoremas de Kohn-Sham (KS) [123,124] requiere el uso de una cuadrícula en el espacio real para la evaluación de las integrales de intercambio-correlación. La mayoría del software utiliza un algoritmo de construcción de cuadrículas introducido por primera vez por Becke en 1988 [125]. En la práctica, las integrales se calculan como sumas ponderadas sobre una cantidad finita de puntos de cuadrícula generalmente definidos en la superficie de una esfera. El "grosor" de estas cuadrículas suele estar representado por el número correspondiente de puntos radiales y angulares. La elección de la cuadrícula puede afectar significativamente los resultados calculados, como lo muestran, por ejemplo, Wheeler y Houk [126] para la familia M06 [33] de los funcionales de Minnesota, y Mardirossian y Head-Gordon [1,127] para dos generaciones de funcionales de Minnesota desarrollados entre 2006 y 2016. También señalan que los funcionales meta-GGA generalmente requieren cuadrículas más finas que los funcionales GGA.

La cuestión de elegir la cuadrícula o malla de integración más adecuada a menudo se pasa por alto, y muchos profesionales de la computación no son conscientes de ello. Confiar en la configuración predeterminada de los programas de química cuántica no es una buena idea, incluso si algunos desarrolladores implementaron requisitos de cuadrícula variable que dependen de la funcionalidad elegida [126] Para esta experiencia, estudiará la curva de disociación del dímero de argón. Las geometrías de estos sistemas provienen del subconjunto RG10 [127] de la base de datos MGCDB84 [6]. La distancia entre los dos átomos de argón varía entre 3,0 y 6,0 Ångströms. Los datos de referencia (kcal mol^{-1}) se reportan en la siguiente tabla.

Tabla 1 Valores de Referencia en kcal mol^{-1} para el dímero de argón, presentando la distancia entre Ar-Ar en Å.

Distancia	Energía	Distancia	Energía	Distancia	Energía
3.0	2.24	4.0	–0.25	5.0	–0.08
3.1	1.21	4.1	–0.23	5.1	–0.07
3.2	0.57	4.2	–0.21	5.2	–0.06
3.3	0.18	4.3	–0.18	5.3	–0.05
3.4	–0.06	4.4	–0.16	5.4	–0.05
3.5	–0.19	4.5	–0.14	5.5	–0.04
3.6	–0.26	4.6	–0.13	5.6	–0.04
3.7	–0.28	4.7	–0.11	5.7	–0.03
3.8	–0.28	4.8	–0.10	5.8	–0.03
3.9	–0.27	4.9	–0.09	5.9 6.0	–0.03 –0.02

Sugerimos usar los siguientes funcionales: : BVP86 [19-21], B3LYP [22,23,26–28], B3PW91 [20,22,24,27,29], PBE [30], HSEH1PBE [30], APFD [31,32], M11 [34], THCTHHYB. Como de costumbre, siéntase libre de agregar los funcionales que desee. El conjunto base elegido es def2-QZVP [44] para todos los cálculos. Probaremos diferentes opciones de cuadrícula: SG1 (si está disponible), (75 302), (99 590) y (175 974). Estas grillas corresponden a las grillas SG1, Fine, UltraFine y SuperFine en Gaussian, y se pueden solicitar con la palabra clave XCGRID en Q-Chem.

Cuando presente los resultados, trate de usar un diagrama de disociación como el que se muestra en la Figura 4 para el funcional VSXC [134,135]. Antes de generar los gráficos, recopile todos los resultados en una tabla con el formato de la Tabla 2 en la página siguiente.

Tabla 2: Datos utilizados para generar la curva SG1 en la Figura 4. La energía se reporta en kcal mol^{-1}, mientras que la distancia está en Å.

Funcional:		**VSXC**	**Grid:**		**SG1**
Distancia	**Energía**	**Distancia**	**Energía**	**Distancia**	**Energía**
3.0	2.33	4.1	–0.18	5.2	–0.12
3.1	0.71	4.2	–0.04	5.3	–0.09
3.2	–0.63	4.3	–0.17	5.4	–0.05
3.3	–1.04	4.4	–0.20	5.5	–0.01
3.4	–0.53	4.5	–0.13	5.6	0.01
3.5	–0.18	4.6	–0.18	5.7	0.02
3.6	–0.50	4.7	–0.24	5.8	–0.01
3.7	–0.81	4.8	–0.08	5.9	–0.06
3.8	–0.69	4.9	0.00	6.0	–0.10
3.9	–0.66	5.0	–0.06		
4.0	–0.58	5.1	–0.12		

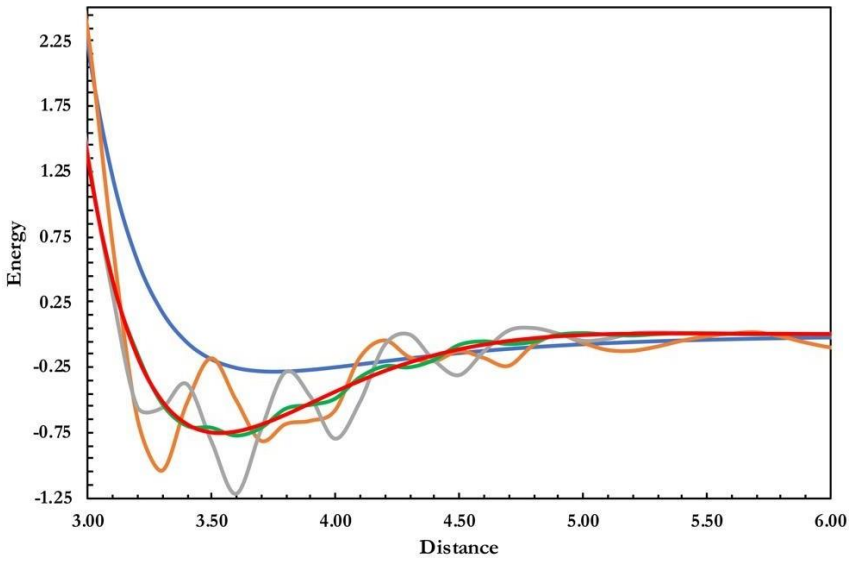


Figura 4 Comportamiento del funcional VSXC en la descripción del dímero Ar con diferentes grillas de integración: SG1 (curva naranja), (75,302) (curva gris),

(99,590) (curva verde), (175,974) (curva roja). La curva de referencia se muestra en azul. La energía está en kcal mol^{-1}, mientras que la distancia está en Å.

Problemas:

1) ¿Encontraste algún funcional problemático?

---

2) ¿Por qué observas varios mínimos y máximos en estas curvas? ¿Crees que podrían ser algo sin sentido físicos?

---

3) Compara los resultados obtenidos utilizando la cuadrícula o malla que tiene el menor y el mayor número de puntos. ¿Qué funcionales están bien convergidos con respecto a estas grillas?

---

4) ¿Cuáles son los requisitos mínimos de la red para los funcionales probados?

---

5) ¿Notó alguna diferencia en el tiempo necesario para los cálculos? Recopile los resultados en una tabla y represente el tiempo necesario y el tamaño de la cuadrícula en un gráfico. ¿Qué tan notable es el aumento de tiempo?

---

6) Teniendo en cuenta los resultados que recopiló, ¿qué cuadrícula(s) recomendaría?

---

7) ¿El método HF sufre este problema? ¿Por qué?

Experiencia avanzada: esta experiencia es opcional, pero enfatiza un problema sutil al disociar una molécula en dos fragmentos diferentes.

Ejecute los cálculos con un funcional de su elección (y el conjunto básico def2-QZVP) en el catión dímero de argón, Ar_2^+. Puede usar las mismas geometrías de la base de datos RG10 pero cambie la carga y la multiplicidad a +1 y 2, respectivamente. Traza las energías en un gráfico similar a la Figura 4. ¿Qué ves?

Comentario avanzado: En teoría, la energía de dos fragmentos mantenidos a una distancia muy grande entre sí debería ser igual a la suma de las energías de los dos fragmentos separados. En términos matemáticos, podemos expresar este concepto como E (A+B) = E(A) + E(B). Esta propiedad se denomina consistencia de tamaño [136]
Desafortunadamente, esta propiedad es violada por la mayoría de los funcionales [137] cuando se disocian los cationes de los dímeros de gases nobles (X_2^+), que dan curvas de disociación como la que se muestra en la Figura 5 a continuación para Ar_2^+ calculado con B3LYP/def2-QZVP. Este comportamiento no es físico, y siempre debe buscar el consejo de un experto para que lo guíe en tales situaciones.

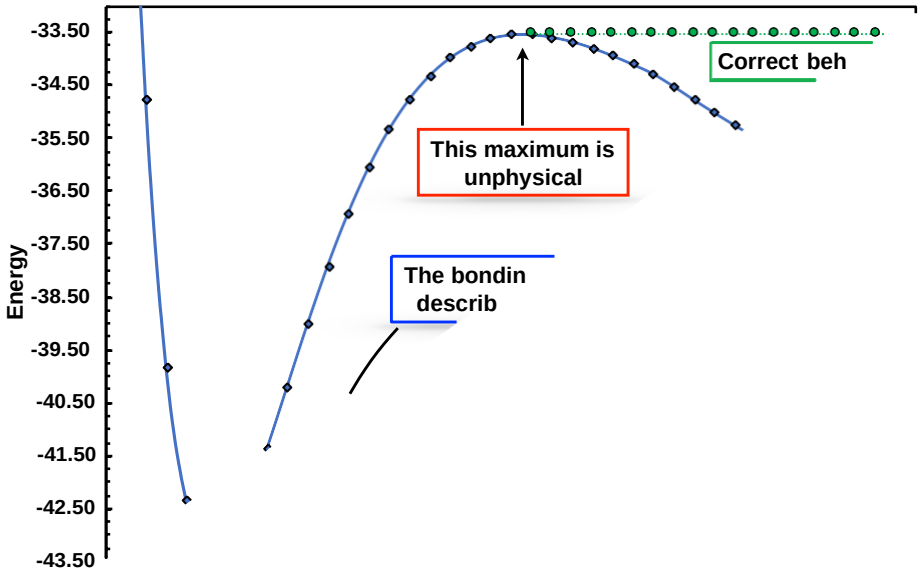


Figura 5 Curva de disociación de la molécula Ar_2^+ calculada con B3LYP/def2-QZVP. La curva azul no es física, mientras que la curva verde representa el comportamiento asintótico correcto.

Experiencia 9: Mallas de Integración 2: El Caso de But-2-yne.

En una publicación reciente [138], Bootsma y Wheeler descubrieron que algunos funcionales son sensibles a la elección de la malla de integración no solo para energías electrónicas [2,126,127], sino también para propiedades termoquímicas como entropías y energías libres de Gibbs. Reportaron cinco reacciones diferentes y, en esta experiencia, nos centraremos en la primera reacción que informaron: la reacción de isomerización del butadieno a but-2-ino. Los cálculos que vas a realizar solo involucran but-2-ino y sus propiedades termoquímicas. Para configurar esta experiencia, utilizará el funcional B97-D GGA [67] y el conjunto básico def2-TZVP [44]. Debe utilizar cuatro cuadrículas de integración: la cuadrícula SG1 podada, y las mallas de Lebedev (75,302), (99,590) y (175,974). Los tres últimos corresponden a la cuadrícula Fina, Ultrafina y Superfina en Gaussian, respectivamente. Dado que se necesita calcular las propiedades termodinámicas de la molécula, debe configurar un cálculo de frecuencia. Tenga en cuenta que el cálculo de la frecuencia también proporcionará energía electrónica SCF. No se requiere optimización de la geometría.

Problemas:

1) Eche un vistazo a las geometrías con un programa de visualización molecular. ¿Son iguales las moléculas? ¿Que cambios notas?

_____

_____

2) Recopile las energías electrónicas de las moléculas. ¿Como se ven?

_____

_____

3) Analice la frecuencia más pequeña de las moléculas. ¿Las moléculas están en su mínimo, estructuras de transición, u otra?

_____

_____

4) ¿Cuánto aumentó el tiempo computacional al pasar del SG1 a la cuadrícula (175,974)?

_____

_____

5) ¿Qué conclusión puedes sacar de esta experiencia? ¿Hay una cuadrícula de integración recomendada?

6) Vuelva a ejecutar esta experiencia usando la palabra clave "NoSymm" en Gaussian/Q-Chem (o el equivalente para su programa de elección). ¿Notas alguna diferencia?

7) Pruebe un funcional de su elección. ¿Está afectado por el mismo problema?

8) ¿Crees que este problema afecta a todas las funciones, o solo a algunas de ellas?

Sugerencia: el problema de las cuadrículas de integración es sutil y es imposible saber de antemano qué moléculas, o qué funciones, podrían verse afectadas. La malla de integración (99,590) representa el requisito mínimo que nos permite obtener resultados confiables sin un incremento sustancial en el tiempo computacional. Bootsma y Wheeler también demostraron que las mallas de integración no son invariantes con respecto a la rotación de la molécula. Para evitar este problema, los programas de química cuántica giran la geometría de entrada a la denominada "orientación estándar" de forma predeterminada y luego la giran hacia atrás antes de que se presenten los resultados. La palabra clave en gaussian "NoSymm" o el equivalente "SYMMETRY = FALSE" en Q-Chem evitan este procedimiento. Como pauta general, los practicantes principiantes deben evitar el uso de la simetría, ya que podría generar problemas numéricos, convergencia a estructuras moleculares restringidas u otros inconvenientes. Sin embargo, si la simetría es necesaria para la reducción de costos, se recomienda buscar la ayuda de un experto.

Experiencia 10: Análisis de Estabilidad y Metales de Transición.

Ya debería estar familiarizado con el hecho de que los coeficientes de las funciones base en los orbitales de Kohn-Sham se obtienen al final de un procedimiento iterativo llamado procedimiento de campo autoconsistente (SCF). Dado un conjunto inicial de coeficientes para las funciones base (llamada matriz de densidad inicial, o simplemente "suposición"), entrega los coeficientes finales optimizados. Los químicos computacionales generalmente se refieren a este conjunto final como la solución SCF convergente.

La solución convergente del procedimiento SCF es un punto estacionario, pero no podemos saber de antemano si es un mínimo, un máximo o un punto silla. Para asegurarnos de obtener la energía más baja, debemos caracterizar el punto estacionario y corregirlo si es necesario. Este procedimiento implica el cálculo de las segundas derivadas de la función de onda y se denomina análisis de estabilidad. Tenga en cuenta que un análisis de estabilidad podría realizarse más de una vez. Hay dos restricciones diferentes que podemos levantar, como se explica en la ref. [137]: para bajar la energía del sistema una vez que se detecta una inestabilidad. Podríamos cambiar el marco teórico que estamos adoptando (restringido o no restringido) o podríamos usar orbitales complejos en lugar de reales. Se dice que la función de onda que obtenemos después de un análisis de estabilidad es estable y entrega una energía más baja que una solución inestable. Puede preguntarse cómo se puede llegar a una solución inestable. Por ejemplo, el procedimiento SCF puede terminar erróneamente en un estado excitado mientras realiza cálculos en el estado fundamental de una molécula. Si la suposición que proporcionamos es razonable, tal evento es poco probable para moléculas que incluyen principalmente elementos del grupo principal, pero es más probable para sistemas con estados bajos casi degenerados. La función de onda correspondiente sería entonces inestable, y sin el análisis de estabilidad no podemos corregirla. Una buena descripción del procedimiento de análisis de estabilidad se encuentra en las refs. [137–141].

Para esta experiencia, utilizará los funcionales habituales : BVP86 [19-21], B3LYP [22,23,26–28], B3PW91 [20,22,24,27,29], PBE [30], HSEH1PBE [30], APFD [31,32], M11 [34] , THCTHHYB y Hartree-Fock (HF) [39–42] y MP2 [43] con la base def2-QZVP establecida [44].

Realizará cálculos para las energías de excitación de dos átomos, Fe y Ru^{+}. Los valores de referencia son 34,32 kcal mol^{-1} para Fe y 21,96 kcal mol^{-1} para Ru^{+}. Los datos provienen de los subconjuntos 3dAEE8 [142–144] y 4dAEE5 [145] de la base de datos Minnesota 2015B [6]. Debe realizar dos conjuntos de experiencias: en el primero, calcula la energía de excitación a partir del cálculo sin usar el análisis de estabilidad. En el segundo, debe repetir los cálculos con el análisis de estabilidad y reoptimizar la función de onda si se encuentran inestabilidades (palabra clave Stable=Opt en Gaussian). Como de costumbre, calcule los errores entre sus resultados y el valor de referencia.

Problemas:

1) Compare los resultados de los errores: ¿cree que necesita el análisis de estabilidad?

2) ¿Cómo funciona el método HF? ¿Por qué crees que funciona de esta manera?

3) ¿Qué funcional recomendarías en este caso?

4) Vuelva a ejecutar los cálculos para las reacciones en las Experiencias 3 y 7, incluido el análisis de estabilidad. ¿Notas algún cambio?

Sugerencia: este ejemplo muestra de manera simple como realizar cálculos que involucran metales de transición no es una tarea fácil. Incluso para una reacción que involucre solo especies atómicas, los resultados pueden mostrar errores muy grandes y, en casos extremos [146], no es sencillo entender qué funcionales dan los resultados correctos. Como consejo general, siempre es una buena idea realizar el análisis de estabilidad de la solución final, y es obligatorio para sistemas con estados bajos, como los metales de transición, y más aún si se utiliza un conjunto base pequeño. Siempre se recomienda buscar la ayuda de un teórico en estos casos problemáticos.

Experiencia 11: Comparación con resultados experimentales.

Esta experiencia analiza el resultado de dos reacciones de disociación de enlaces de la base de datos Minnesota 2015B [6]. Una proviene del subconjunto SR-MGN-BE107 [12,128,147], y la otra del subconjunto SR TM BE17 [12,133,148–150]. Además, también examinará algunas energías de atomización totales del subconjunto W4-subset [151] de la base de datos GMTKN55 [2].

Primero, vas a analizar la disociación del enlace C-O en el alcohol terc-butílico para dar el radical terc-butilo y el radical OH. La energía de referencia para esta reacción es de 115,02 kcal mol^{-1}. En el segundo caso, tendrá en cuenta la molécula FeCl [150] y su disociación en cloro atómico y hierro. La energía electrónica de referencia para su disociación es de 78,5 kcal mol^{-1}.

No olvide utilizar el análisis de estabilidad para ambos casos. Recuerde también utilizar al menos la cuadrícula (99.590) y el conjunto de bases def2-QZVP [44]. Cuando se trata de funcionales, sugerimos utilizar los funcionales M11[34], B3LYP [22,23,26-28] y PBE [30], y el método HF [39-42]. Antes de responder a las preguntas a continuación, le sugerimos que recopile los datos y los coloque en una tabla, incluidos los resultados y el error calculados con respecto a la energía de referencia.

Problemas para el alcohol terc-butílico.

1) ¿Cuál funcional es el mejor y cual el peor?

---

2) ¿Cómo funciona el método HF? ¿Te sorprende??

---

3) Pruebe el funcional B3LYP con el conjunto de base 6-31G* de Pople [116–124]. ¿Ve alguna mejora? ¿De dónde crees que viene esta mejora?

Problemas de FeCl.

1) ¿Qué funcional es el de mejor desempeño?

---

2) ¿Cómo funciona el método HF? ¿Te sorprende??

---

3) Vaya al sitio web del Nist Webbook of Chemistry y busque la entalpía de formación en la fase gaseosa de FeCl. ¿Cuánto es, en kcal mol^{-1}? ¿Ve alguna diferencia con los datos de referencia que está utilizando?

---

4) Dada la referencia del NIST y la energía electrónica, ¿crees que puedes comparar directamente el resultado teórico con el experimental? ¿Por qué?

Ahora centraremos la atención en las reacciones de atomización [151]. Recuerde que la definición de reacción de atomización implica que descomponemos la molécula en sus átomos constituyentes (véase, por ejemplo, la reacción 3c en la Experiencia 3). Tenga en cuenta la molécula H_2, metano, agua, amoníaco y ozono. Los valores de referencia (en kcal mol^{-1}) se indican en la siguiente tabla (Tabla 3).
Tenga en cuenta que para la molécula H_2, la reacción de atomización y la reacción de disociación de enlace son conceptualmente las mismas.
Sugerimos utilizar los mismos métodos utilizados anteriormente. Las energías de atomización total no son muy importantes desde el punto de vista químico, pero son interesantes desde el punto de vista computacional porque representan un reto en el desarrollo funcional y porque son una buena prueba de la robustez y fiabilidad de las aproximaciones funcionales. Recuerde recopilar los datos en una tabla antes de continuar y responder las siguientes preguntas.

Molécula	**Energía de Referencia (kcal mol^{-1})**
H_2	109.5
CH_4	420.4
NH_3	298.0
H_2O	233.0
O_3	147.4

Tabla 3 Energías de referencia para las energías de atomización totales del subconjunto W4-11.

Problemas del subconjunto W4-11.

1) ¿Cuál funcional es el mejor y cual el peor?

---

2) ¿cómo funciona el método HF? ¿te sorprende?

---

3) En tu opinión ¿cuál es la causa de que el método HF fallara?

4) ¿Ve la misma tendencia tanto en las reacciones de disociación de enlaces como en las reacciones de atomización?

---

5) ¿Se puede comparar directamente la energía de atomización con la entalpía de formación de una sustancia? ¿Por qué?

Comentario de práctica: Las energías de disociación de enlaces son omnipresentes en las bases de datos [2-6], porque es muy importante para los funcionales xc describirlas correctamente. Sin embargo, incluso las reacciones simples pueden ser complicadas: la disociación del enlace C-O es en principio fácil, pero da como resultado dos moléculas radicales cuya descripción es desafiante para los métodos teóricos (recuerde que en principio KS-DFT es una teoría de determinante único como HF [152]).
Para la energía de disociación de FeCl, el escenario se complica aún más cuando se trata de elementos más pesados, o cuando se disocian moléculas (cargadas) en los fragmentos constituyentes, especialmente si son diferentes (como en el caso de Ar_2^+ descrito al final de la Experiencia 8). Además, recuerde que no hay una manera sencilla de comparar el resultado calculado con el resultado experimental, como lo explicaron Truhlar y sus colaboradores [148,150]. Sin entrar en muchos detalles, hay que tener en cuenta diferentes correcciones para pasar del valor experimental al puramente teórico. Es por eso por lo que se desaconseja la comparación directa entre los valores experimentales y los calculados, a menos que busque ayuda de un teórico. Llegando al último conjunto de reacciones, pueden verse como un ejemplo adicional de por qué debe confiar en los funcionales más modernos. No solo necesitamos un funcional que sea capaz de describir la molécula de la manera más adecuada, sino que también necesita describir correctamente los átomos, que son una especie química completamente diferente. El método HF no tiene en cuenta la correlación de electrones y, por lo tanto, su descripción de los átomos es realmente mala, lo que resulta en errores mayores. Por otro lado, todos los DFA (por sus siglas en inglés) representan la correlación electrónica de alguna manera aproximada [153–157], y sus resultados para los átomos son mejores. Una vez más aconsejamos a los no expertos que busquen ayuda de un teórico para tales casos.

Comentario avanzado: Las aplicaciones modernas de la química cuántica de la DFT no se limitan al cálculo de energías electrónicas. De hecho, otras cantidades como las energías libres, entalpías y entropías de Gibbs se pueden calcular fácilmente solicitando un cálculo de termoquímica (o "frecuencia"). Estas cantidades también se pueden medir experimentalmente, y puede ser tentador comparar los resultados experimentales y calculados entre sí. Tenga en cuenta que las configuraciones computacionales y experimentales son diferentes, por lo que tal comparación directa no es posible, y debe evitarse [158].
Las constantes de equilibrio o disociación también se pueden asociar con los valores de energía libre de Gibbs. Calcular estos valores no es fácil, principalmente debido a las limitaciones asociadas con la simulación de interacciones solvente-soluto. A pesar de los muy buenos avances en este campo [159], debe evitarse la comparación directa de valores experimentales y calculados.

Experiencia 12: Revisión final.

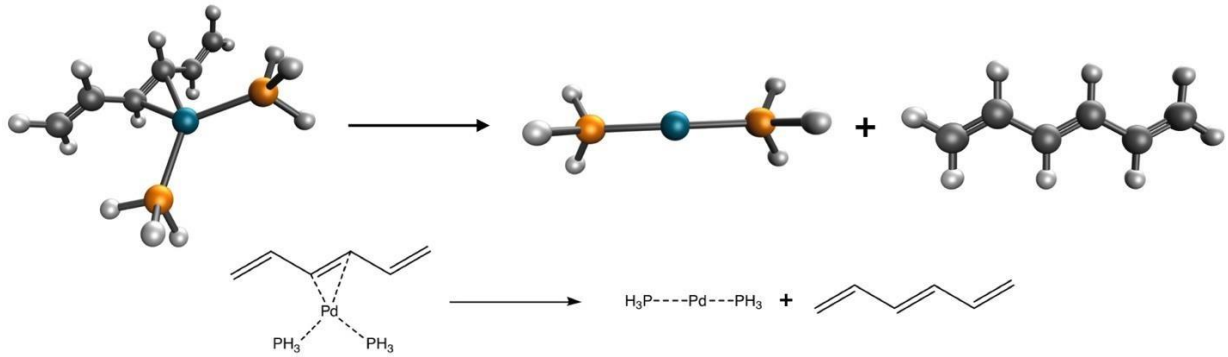


Figura 6 Reacción de disociación del complejo difosfinohexatrienopaladio (0). Los átomos de carbono son negros, los átomos de fósforo son naranjas, los átomos de hidrógeno son blancos, el átomo de paladio es azul claro. Las líneas punteadas muestran enlaces dativos.

Imagine que alguien en su departamento está estudiando el siguiente complejo de Pd (estas estructuras se toman del subconjunto PdBE2 [149] de la base de datos Minnesota 2015B [6]), y le piden que calcule la energía de disociación del enlace para la reacción reportada a continuación (figura 6). Ahora está a cargo de configurar los cálculos. Elija el conjunto de funcional y base que desea utilizar en función de la literatura publicada (por ejemplo, consulte estas publicaciones recientes [2–7,12,160–162]) y de la disponibilidad que tenga con base al programa que tenga y su experiencia con las experiencias anteriores. Recuerde también que realizar cálculos es un compromiso entre la precisión y las capacidades computacionales. Después de terminar todos los cálculos y recopilar todos los resultados, escriba un breve informe (~ 300 palabras) indicando el método que adoptó y por qué lo eligió. Además, describa la configuración computacional adecuadamente, presente los resultados que obtuvo y compárelos con el valor de referencia de 16.20 kcal mol^{-1}. No olvide agregar un título y una sección de referencia apropiada

MANUAL DE MACHINE LEARNING ENFOCADO A LA QUÍMICA

PRODUCED BY: CARLOS ALDAIR PACHECO GONZALEZ
IRVING EDUARDO BALANZAR URZUA
JORGE MARTINEZ VEGA
INGRID MICHELLE GUTIERREZ GARCIA
DAPHNE HANIEL ROMERO TORRES
MARICELA MALAGON ALVAREZ

CA.PACHECOGONZALEZ@UGTO.MX
IE.BALANZARURZUA@UGTO.MX
J.MARTINEZVEGA@UGTO.MX
IM.GUTIERREZGARCIA@UGTO.MX
DH.ROMEROTORRES@UGTO.MX
M.MALAGON.ALVAREZ@UGTO.MX

ASESORES: JESUS EDUARDO CASTELLANOS AGUILA
OLEKSIY SHULIKA

Índice

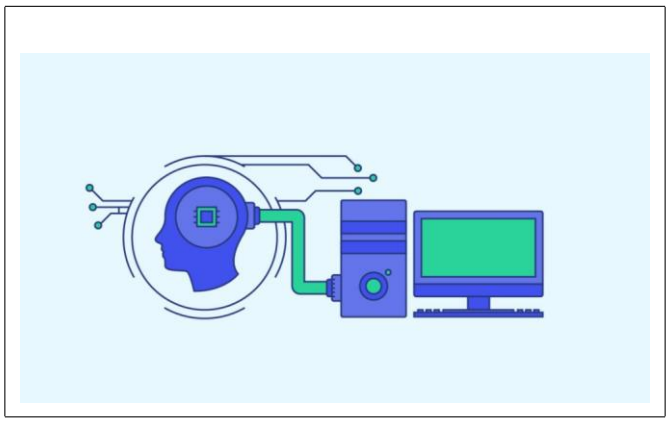
SECTION

MACHINE LEARNING

1.1. ¿Qué es el Machine Learning?

Es una rama de la inteligencia artificial que se enfoca en enseñar a las computadoras a aprender y mejorar por sí mismas a partir de la experiencia y los datos, en lugar de programar explícitamente un conjunto de reglas para realizar una tarea.

El Machine Learning permite a la computadora aprender patrones y relaciones en los datos de entrada para hacer predicciones o tomar decisiones. Se utiliza en una amplia variedad de aplicaciones, desde la detección de fraudes en transacciones financieras hasta la recomendación de productos en línea, y es una herramienta cada vez más importante en la tecnología moderna.



1.2. ¿Cómo funciona el Machine Learning?

El primer paso es seleccionar y preparar un conjunto de datos de entrenamiento. Esos datos se utilizarán para alimentar el modelo de Machine Leraning para aprender a resolver el problema para el que se ha diseñado. Los datos se pueden etiquetar para indicarle al modelo las características que debe identificar. También pueden estar sin etiquetar, entonces será el modelo el que deberá detectar y extraer características recurrentes por sí mismo. En ambos casos, los datos deben prepararse, organizarse y limpiarse cuidadosamente. De lo contrario, el entrenamiento del modelo Machine Learning puede estar sesgado. Los resultados de sus predicciones futuras se verán afectados directamente. El segundo paso es seleccionar el algoritmo a ejecutar. La sección del algoritmo depende del tipo y volumen de datos de entrenamiento, así como del tipo de problema a resolver. El tercer paso es entrenar el algoritmo. Es un proceso de repetición. Las variables se ejecutan a través del algoritmo y los resultados se comparan con los que debería haber producido. Los (pesos) y el sesgo se pueden ajustar para aumentar la precisión del resultado. Después, se vuelven a ejecutar las variables hasta que el algoritmo produzca los resultados correctos en la mayoría de los casos. El algoritmo entrenado es el modelo de Machine Learning. El cuarto y último paso es el uso y la mejora del modelo. Utilizamos el modelo sobre nuevos datos, cuyo origen depende del problema que haya que resolver. Por ejemplo, en los correos electrónicos se usara un modelo de Machine Learning diseñado para detectar spam.

1.3. ¿Cuáles son los principales algoritmos de Machine Learning?

El Machine Learning (Aprendizaje Automático) utiliza una variedad de algoritmos para entrenar modelos y realizar predicciones o tomar decisiones basadas en datos.Como por ejemplo:

- Regresión lineal: Un algoritmo de aprendizaje supervisado utilizado para predecir valores numéricos continuos. Busca encontrar la mejor línea recta que se ajuste a los datos.
- Regresión logística: Otro algoritmo de aprendizaje supervisado utilizado para la clasificación binaria. Estima la probabilidad de que una instancia pertenezca a una clase específica.
- Árboles de decisión: Un algoritmo de aprendizaje supervisado que crea una estructura de árbol para tomar decisiones basadas en características de los datos. Puede ser utilizado tanto para problemas de clasificación como de regresión.
- Random Forest: Es una técnica que combina múltiples árboles de decisión para realizar predicciones. Cada árbol se entrena con una submuestra de los datos y las predicciones finales se obtienen promediando las predicciones de todos los árboles.

- Naive Bayes: Un algoritmo de aprendizaje supervisado basado en el teorema de Bayes. Es especialmente efectivo para el análisis de texto y clasificación de documentos.
- Máquinas de vectores de soporte (SVM): Un algoritmo de aprendizaje supervisado que encuentra un hiper plano óptimo para separar las clases de datos. Es útil para problemas de clasificación lineal y no lineal.
- K-means: Un algoritmo de aprendizaje no supervisado utilizado para la agrupación de datos. Agrupa las instancias en k grupos basándose en la similitud de sus características.
- Redes neuronales: Algoritmos de aprendizaje profundo que intentan imitar el funcionamiento del cerebro humano. Consisten en capas de nodos interconectados que aprenden a partir de los datos.

Estos son solo algunos de los algoritmos más comunes en el campo del Machine Learning. Cada uno tiene sus propias características y se adapta mejor a diferentes tipos de problemas y conjuntos de datos.

1.4. ¿Qué es el Deep Learning?

El Deep Learning (Aprendizaje Profundo) es una subrama del campo del Machine Learning que se enfoca en el entrenamiento de redes neuronales artificiales profundas. Estas redes están compuestas por múltiples capas de neuronas interconectadas, lo que les permite aprender y extraer representaciones cada vez más abstractas de los datos de entrada.
El término "profundo" se refiere a la profundidad de la arquitectura de la red, que puede tener muchas capas ocultas. Cada capa procesa la información recibida de la capa anterior y la pasa a la siguiente capa, permitiendo un aprendizaje jerárquico y progresivo de las características de los datos. A medida que los datos pasan a través de las capas, la red neuronal puede aprender representaciones más complejas y significativas, capturando características y patrones de alto nivel.
El Deep Learning ha demostrado ser especialmente efectivo en áreas como el procesamiento de imágenes y videos, el procesamiento de lenguaje natural y el reconocimiento de voz. Por ejemplo, en el procesamiento de imágenes, las redes neuronales profundas pueden aprender automáticamente a reconocer objetos, rostros y características relevantes en las imágenes. En el procesamiento de lenguaje natural, pueden comprender y generar texto de manera más precisa.
Una de las razones del éxito del Deep Learning es su capacidad para aprender características y representaciones directamente de los datos, sin requerir una ingeniería manual intensiva de características. Además, el uso de técnicas como la retro propagación del error y el descenso de gradiente permite entrenar redes neuronales profundas de manera eficiente.

Sin embargo, el Deep Learning también presenta desafíos, como la necesidad de grandes cantidades de datos de entrenamiento, la elección adecuada de la arquitectura de red y los requisitos computacionales intensivos para el entrenamiento. Además, la interpretación y explicación de los resultados de las redes neuronales profundas pueden ser difíciles debido a su naturaleza altamente no lineal y su complejidad.

A pesar de estos desafíos, el Deep Learning ha impulsado importantes avances en áreas como el reconocimiento de imágenes, el procesamiento de voz, la traducción automática y muchas otras aplicaciones. Sigue siendo un campo activo de investigación y desarrollo, con un gran potencial para seguir transformando diversas industrias y dominios.

SECTION

Python

Es un lenguaje de programación de alto nivel de programación y fácil de comprender. Cuenta con estructuras de datos eficientes y un sistema de programación orientada a objetos con el que podremos visualizar información de distintas maneras.
El intérprete de Python cuenta con una extensa biblioteca de librerías disponibles de manera gratuita para la mayoría de plataformas.

2.1. Números

En python trabajar con números desde nuestro interprete(Colab) es muy sencillo, de manera cómoda se podría utilizar como una calculadora. Dentro cada linea que escribimos en el interprete es una instrucción de código, si escribimos una instrucción y luego la ejecutamos con la combinación `crtl + enter` podremos visualizar nuestro resultado.

Listing 1: Números

```
1 3+4
```

Resultado:

```
7
```

Algo importante dentro de Google Colab es la posibilidad de contar con celdas de texto que nos permitirán dejar comentarios y observaciones acerca de nuestro código. Este puede ser

usado pulsando +Texto que se encuentra en la barra de herramientas.
Python tiene la posibilidad de trabajar con números flotantes y enteros, de esta manera se puede realizar operaciones sencillas como: Suma, resta, multiplicación, división, modulo, potencia, sin la necesidad de crear o realizar líneas de código complejas.

2.2. Variables

Una variables es un espacio de memoria al que nosotros podemos poner nombre y definir un valor. Este puede ser un valor literal o un valor creado previamente. Las variables son un elemento fundamental en cualquier lenguaje de programación; Python nos facilita bastante el uso de ellas, debido a que no necesitamos definir que tipo de variable utilizaremos.
Para la creación de una variable colocaremos dentro de Google Colab N=3. Después procederemos a hacer un salto de linea y colocar N, de esta manera asignaremos a nuestra variable un nombre (N) y un valor, que se mostrara cada que llamemos a nuestra función.

Listing 2: Ejemplo Python de asignación de un valor a una variable (o definición de una nueva variable)

```
1 n = 3
2 n
```

Resultado:

```
3
```

Al haber definido nuestra variable con un valor numérico, podremos usarlo para las diversas operaciones:

Listing 3: Operaciones con los numeros y variables asignadas.

```
1 n + 6
```

Resultado:

```
9
```

Ejemplo 2:

Listing 4: Operaciones con los números y variables asignadas.

```
1 n*2
```

Resultado:

```
6
```

De esta manera podremos realizar las distintas operaciones con el uso de variables, pero estas contienen una serie de reglas a cumplir para el buen funcionamiento:

Numero de variables
No puede comenzar por un número.
No puede contener espacios.
No puede contener caracteres especiales.
Se puede utilizar (_) para sustituir los espacios.

Textos

Dentro de la programación los textos son conocidos como cadenas de caracteres y se pueden expresar de varias formas.

Dentro de Python normalmente las cadenas de caracteres usando comando `Print()`, el cual nos ofrece una serie de ventajas por encima de algunas otras.

Listing 5: Ejemplo de cadena de textos

```
1 Print(" Hola Mundo!")
```

Resultado:

```
Hola Mundo
```

El `Print()`, También nos permite el uso de caracteres especiales, tabulaciones y espacios:

Listing 6: Tabulación ^

```
1 Print("Un Texto \tuna Tabulación")
```

Resultado:

```
Un texto    una
tabulaci\'on
```

Listing 7: Salto de línea \n

```
1 Print("Un Texto \nuna línea nueva")
```

Resultado:

```
Un Texto
una línea nueva
```

Algo a tomar en cuenta es que una cadena de texto en Python está compuesta por la unión de distintos caracteres.

2.3. Tipos de Datos

En Python hay tipos específicos de valores que podemos usar. En este notebook vamos a presentar los valores “int”, “float”, “str”, “list” & “boolean”:
Boolean(bool): es una variable que puede ser verdadera o falsa.
Integer (int): representa números enteros positivos o negativos, como **8 a -678**.
Floating point number (float): representa números reales, como **14159 a -6.5**.
String de caracteres (llamado “string”, str): son cadenas de caracteres, es decir, valores de texto, por ejemplo "hola.º "banana".
Listas (list) : este tipo de valor es una lista, ya sea de valores integer o float. Por ejemplo: [1,2,3,4] o [1.23, 2, 3.54, 4]
Podemos checar el tipo de valor con: type( ) y el nombre de la variable, de esta forma:

Listing 8: Uso del Type()

```
1       Print (" type (3) ")
```

Resultado:

```
    int
```

Un string es un grupo de caracteres contables que puede contener números o no, al incluirse caracteres diferentes a los números, nuestro string pierde cualquier axioma matemático conocido.

Listing 9: String

```
1       '12' + '34' + 'A'
```

Resultado:

```
    '1234A'
```

Las listas son tipos de datos predeterminados que permiten almacenar múltiples elementos en un espacio de memoria. Los elementos están ordenados numéricamente y se pueden modificar después de la creación de la lista.
Se puede accesar a cada elemento de nuestra lista mediante su índice. Cada elemento está enumerado y solo necesitamos llamar nuestra función con su respectivo índice comenzando desde index[0], usando list[N]:

Listing 10: List

```
1 list = ["Element1", 2, 3.14]
2 list[0]
```

Resultado:

```
'Element1'
```

2.4. Lectura por teclado

La lectura por teclado nos permitirá guardar un valor en una variable que introduzcamos por teclado, es decir nos deja escribir desde el teclado, para ello usaremos la función `input();`

Por ejemplo, si quiero guardar un valor desde el teclado, creare una variable llamada "`valor`" y a esta le asignaré el valor de `input()`.

Listing 11: `input()`

```
1         valor = input()
```

Resultado:

```
|_________|
```

El programa nos proporcionara una caja para colocar una valor que se guardará en nuestra variable.

la variable definida por `input()` guardara lo escrito como una cadena de caracteres e imprimirá solo una caja vacía, estos se puede modificar añadiendo un tipo de variable antes del `input()` y un texto con instrucciones dentro de nuestro `input()`, por ejemplo:

Listing 12:

```
1 Valor = int(input("Ingrese un valor: "))
```

Resultado:

```
Ingrese un valor:| ________|
```

2.5. Control de flujo

La sentencia `if` es posiblemente la más famosa y utilizada en toda la programación, ya que nos permite condicionar el flujo de un programa y dividir la ejecución en diferentes caminos.

Para entenderlo más claro, es como darle la capacidad de distinguir varias opciones al ordenador y que este actué de manera distinta en cada caso. El uso del `if` es demasiado sencillo y también nos permitirá instaurar el concepto de "bloque de código", por ejemplo:

Listing 13:

```
1          if True:
2              print("Se cumple la condición")
```

Resultado:

```
Se cumple la condición
```

Se define un bloque de instrucciones después de una sentencia de control con dos puntitos, es importante que todo el bloque que se ejecute lleve un cierto orden, como mantenerlo por delante del `if` con un salto de línea, esto le indicara al IDE que esta dentro del bloque y que es ejecutable.
Se utiliza los comandos `if`, `elif`, `else`, para elaborar secuencias con base a respuestas, por ejemplo:

Listing 14:

```
1          comando = "SALIR"
2          if comando == "ENTRAR"
3              print("Bienvenido al sistema")
4          elif comando == "SALUDAR"
5              print("Holar mundo")
6          elif comando == "SALIR"
7              print("Saliendo del sistema...")
8           else:
9              print("Este comando no se reconoce")
```

El comando `else` por lo general se deja para dar un mensaje genérico que abarque todos aquellos caminos que no se tomaron en cuenta dentro del código bloque anterior.
Además como el comando tiene como valor "Salir", el código imprime el siguiente mensaje:
Resultado:

```
Saliendo del sistema...
```

2.6. Funciones

Nuestra función estará definida como una pieza reusable de código, que solo es ejecutada cuando se la llama. Python tiene algunas funciones integradas, como `print()` (que usamos previamente), `sum()` y `pow()`.

Fuera de estas funciones definidas, nosotros podemos definir y crear nuestras propias funciones. De esta manera podemos reutilizar código para calcular resultados basados en datos diferentes. Por lo cual es bastante común condensar código en una sola función. Para empezar una función, empezamos escribiendo la palabra "def", seguido por un espacio en blanco que nos ayudara a separar nuestros códigos.

Para comprender mejor a una función, debemos entender el concepto de parámetro y argumento:

Parámetro: Un parámetro es una variable que se define al declarar una función. Se escriben en un paréntesis separados por comas.

Argumento: El argumento de un parámetro dado es el valor real del parámetro que se pasa a la función.

Adicionalmente, debemos definir y especificar cuáles serán nuestras salidas de la función para que las podamos llamar individualmente según lo necesitemos. Las salidas se definen como nuevas variables que dependen de nuestros parámetros previamente definidos. Están determinados por la palabra "return".

Listing 15: Funciones

```
1     def function_add (arg1, arg2):
2         #El alcance de nuestra variable result esta definida por
              function_add y se le llama local.
3     result = arg1 + arg2
4     result2 = arg1 * arg2
5     return result, result2
6
7    function_add(4,2)
```

Resultado:

```
(6,8)
```

Como podemos observar en nuestra función anterior function_add contiene los valores y return nos regresara las operaciones matemáticas definidas en result.

2.7. Ejercicio 1.1 Calculadora

Hacer una calculadora definiendo funciones de operación. En particular:

- Realizar una función que sume números.
- Realizar una función que reste números.
- Realizar una función que multiplique números.
- Realizar una función que divida números.

- Realizar una función que calcule una potencia de un número.
- Realizar una función que calcule la raíz cuadrada de un número.
- Ejecutar la calculadora y verificar que tipo (“type”) de resultado obtuvo.
- Imprimir un `string` en la pantalla que diga el resultado de los cálculos (usar la función `str()` para convertir los números en `strings`.

2.8. Ejercicio Resuelto

Implementando los conocimiento adquiridos elaboraremos un código en Google Colab que calcule todo lo requerido basándonos en datos que nosotros le proporcionaremos:

Listing 16: Paso 1: Importar Libreria

```
# En este caso solo se importara la libreria math, para las operaciones matematicas
import math
```

De acuerdo a lo antes visto se abrirá una nueva línea de código donde se empezaran a generar el código de nuestras operaciones.
Posterior a definir las librerías que usaremos, definiremos una por una las funciones que utilizaremos para cada operación matemática de nuestra calculadora.
Colocaremos el formato `def` seguido del nombre operativo y las variables a usar. Definiremos la operación que se ejecutara en `return variable, operador, variable` como se muestra en el ejemplo.

Ejemplo:

Listing 17: Paso 2: Funciones

```
def sumar(a, b):
return a + b
```

Tenemos que realizar en total 6 funciones correspondientes a cada operación propuesta, tenemos dos condiciones aplicables en 2 operaciones correspondientes a división y raíz cuadrada.
Ambas funciones tienen condiciones que cumplir y para ellas usaremos `if & else`. En la división tiene que ser divisible entre un numero ajeno al 0 y nuestra función raíz no puede obtener la raíz de números negativos.
Ejemplos:

Listing 18: Paso 3: Condicionales

```
1
2       def dividir(a, b):
3       if b != 0: #!= nos indica que b debe ser diferente de 0 para
            cumplir la función.
4           return a / b
5       else:
6           print("Error: No se puede dividir entre cero.")
7           return None
8         #al no cumplirse la funcion regresa un mensaje de error.
9
10      def calcular_raiz_cuadrada(numero):
11      if numero >= 0: #define que el numero debe ser mayor que 0
12          return  math.sqrt(numero)
13
14      else:
15          print("Error: No se puede calcular la raiz cuadrada de un
                numero negativo.")
16          return None
```

Tras obtener nuestras 6 funciones definiremos y solicitaremos las variables que usaremos para realizar nuestras operaciones. La mayoría de nuestras funciones solo requieren de 2 variables que nombraremos `a & b`, solicitamos al usuario darnos el valor de `a & b` y los guardamos en sus respectivas variables usando la función `input()`.
Ejemplo:

Listing 19: Paso 4: Variables

```
1       # Solicitar variables a y b al usuario
2       a = float(input("Ingrese el valor de a: "))
3       b = float(input("Ingrese el valor de b: "))
```

Generaremos una lista que nos ayudará a clasificar y seleccionar las operaciones a realizar.
Ejemplo:

Listing 20: Paso 5: Lista

```
1     # Lista de eleccion de operaciones
2   operaciones = [
3       "Suma",
4       "     Resta     ",
5       "Multiplicación",
6       "   Divisi   ón",
7       "Potencia",
8       "Raíz cuadrada"
```

```
]
```

Esta función nos ayudara a imprimir y enumerar la lista de operaciones, haciéndolas visible para el usuario y solicitando seleccionar una:

Listing 21: Paso 6: secuencia

```
# Mostrar lista de operaciones al usuario
print("Seleccione una operación:")
for i, operación in enumerate(operaciones):
    print(f"{i + 1}. {operación}")
```

Aquí se utiliza un bucle `for` junto con la función `enumerate()` para recorrer la lista de operaciones o. La función `enumerate()` devuelve pares de elementos que consisten en el índice y el valor del elemento actual.
En cada iteración del bucle, se asigna el índice a la variable `i` y el valor del elemento a la variable `operación`. Luego, se utiliza la función `print()` para mostrar en la consola la enumeración de cada operación.
En este caso, propondremos un sumador `{i + 1}`, este es utilizado para mostrar el número de índice incrementado en 1 (ya que los índices en Python comienzan en 0), seguido de un punto y el nombre de la operación.
Resultado:

```
1. Suma
2. Resta
3. Multiplicación
```

Haremos uso de `input()` para solicitarle al usuario la operación que quiera ejecutar.
Ejemplo:

Listing 22: Paso 7: Variable L

```
# Solicitar elección de operación al usuario
opción = int(input("Ingrese el número correspondiente a la
   operación elegida: "))
```

Usando la funcion `if-else`, se podrá realizar una operación basada en la respuesta anterior, cada una con su respectivo caso. Para posterior mente usar la variable `resultado` que imprimiremos al finalizar el programa.

Listing 23: Paso 8: Secuencia-Print

```
# Realizar la operación seleccionada
if opción == 1:
    resultado = sumar(a, b)
elif opción == 6:
```

```
    resultado = calcular_raíz_cuadrada(a) if a >= 0 else None
else:
    print("Opción invalida.")
    resultado = None

# Imprimir el resultado
if resultado is not None:
    print("El resultado es:", resultado)
```

Código Calculadora

Listing 24: Calculadora

```
import math

def sumar(a, b):
    return a + b

def restar(a, b):
    return a - b

def multiplicar(a, b):
    return a * b

def dividir(a, b):
    if b != 0:
        return a / b
    else:
        print("Error: No se puede dividir entre cero.")
        return None

def calcular_potencia(base, exponente):
    return base ** exponente

def calcular_raíz_cuadrada(numero):
    if numero >= 0:
        return  math.sqrt(numero)
    else:
        print("Error: No se puede calcular la raíz cuadrada de un
            número negativo.")
        return None

# Solicitar variables a y b al usuario
```

```
a = float(input("Ingrese el valor de a: "))
b = float(input("Ingrese el valor de b: "))

# Lista de eleccion de operaciones
operaciones = [
    "Suma",
    "Resta",
    "Multiplicación",
    "Division",
    "Potencia",
    "Raiz  cuadrada"
]

# Mostrar lista de operaciones al usuario
print("Seleccione una operacion:")
for i, operacion in enumerate(operaciones):
    print(f"{i + 1}. {operacion}")

# Solicitar elección de operación al usuario
opcion = int(input("Ingrese el número correspondiente a la
   operación elegida: "))

# Realizar la operacion seleccionada
if opcion == 1:
    resultado = sumar(a, b)
elif opcion == 2:
    resultado = restar(a, b)
elif opcion == 3:
    resultado = multiplicar(a, b)
elif opcion == 4:
    resultado = dividir(a, b)
elif opcion == 5:
    resultado = calcular_potencia(a, b)
elif opcion == 6:
    resultado = calcular_raiz_cuadrada(a) if a >= 0 else None
else:
    print("Opcion invalida.")
    resultado = None

# Imprimir el resultado
if resultado is not None:
    print("El resultado es:", resultado)
```

2.9. Ejercicios Complementarios

Ejercicios Complementarios: Click aqui
Solucionario: Click aqui

SECTION

Librerías Python

Las librerías dentro de un lenguaje de programación son un conjunto de funciones que permiten realizar distintas operaciones, ahorrándonos tiempo y manteniendo un esfuerzo mínimo dentro de nuestro IDE. Dentro de Python, existen un gran número de librerías gratuitas realizadas por la comunidad.

Las librerías por exelencia usadas parala visualización y graficado de datos dentro de machine learning son: *Numpy*, *Sklearn*, *Seaborn*, *Pandas*, *Matplotlib*.

Para empezar a trabajar con los datos, primero debemos importar "módulos", que son funciones de nuestras librerías. Como ejemplo importaremos dos módulos conocidos para la gestión de datos. "Pandas" proporciona funciones que generarán tablas y nos permitirán hacer diferentes tipos de procesamiento de datos, mientras "Seaborn" nos permite trabajar con las funciones estadísticas habituales y visualizar grandes conjuntos de datos gráficos con una sintaxis simple. Para trabajar "Seaborn" también necesitamos importar la librería gráfica "Matplotlib".

SECTION

4 Importación

En la siguiente sección, aprenderás la manera de importar archivos que contengan bases de datos, los cuales serán necesarios para desarrollar nuestras prácticas y ejemplos prácticos que se presentarán a continuación.

4.1. Importar el Dataset

Con el comando `import` podemos importar los módulos a usar y que están en nuestro ordenador, en caso de estar usando Google Colab, los módulos estarán instalados en una computadora externa.
Cargaremos un conjunto de datos preexistentes dentro del módulo Seaborn. El conjunto de datos con el que trabajaremos llevara por nombre "Cuarteto Anscombe".

Listing 25: `Import`

```
1    import pandas as pd
2    import seaborn as sns
3    #El comando "as" dice el nombre que le queremos dar a el
        modulo dentro de nuestro IDE.
```

Posteriormente carguemos nuestro dataset elegido y vamos a establecerlo como nuestro DataFrame.

Listing 26:

```
df = sns.load_dataset('anscoombe')
```

```
type(df)
```

Resultado:

```
pandas.core.frame.DataFrame
```

Descripción del dataset:
Este dataset contiene información sobre vinos tintos. Algunas variables son de composición química y variables fisicoquímicas, y la calidad es una clasificación sensitiva.
Url del dataset: Click aqui

Número de instancias
1599 vinos tintos
Información de las variables
1.fixed acidity
2.volatile acidity
3.citric acid
4. residual sugar
5.chlorides
6.free sulfur dioxide
7.total sulfur dioxide
8.density
9.pH
10. sulphantes
11.alcohol
12.quality

4.2. Importar de un archivo a Google drive o acceder a uno existente

Estos son los pasos para subir un archivo desde nuestro ordenador a Google Colab

a) Descargamos el dataset desde la URL a la PC.

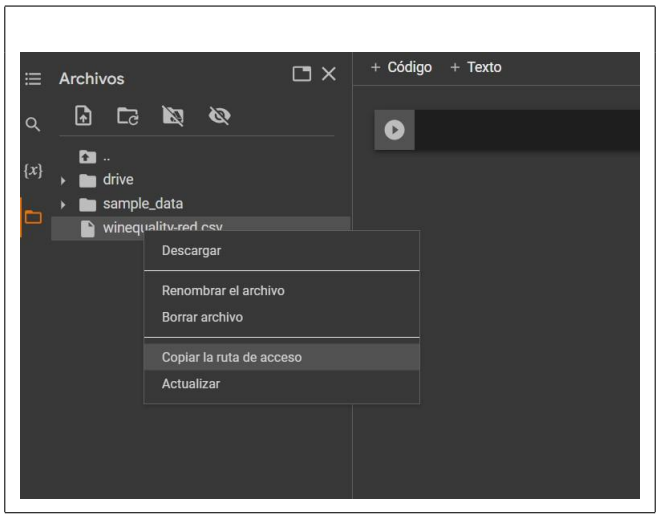
b) Vamos al último icono de la barra lateral a la izquierda de la pantalla (“Archivos” o “Files”).

c) Presionamos el primer icono (una hoja de papel con una flecha hacia arriba) y seleccionamos el archivo descargado previamente.

d) Ejecutar la celda.

e) Visitar el URL mencionado.

f) Copiar el link que aparece y pegarlo en la caja que dice “Enter your authorization code:” como se muestra a continuacion.



Observación 1: El archivo “winequality-red.csv”, es el nombre del archivo que sera usando en esta Notebook. Para utilizar otros archivos, utilizar: “ArchivoQueElijas.csv”

Observación 2: Si el archivo ya se encuentra en Google Drive, comenzar en el paso 4.

Listing 27: Import

```
import google.colab import drive
drive.mount('/content/drive')
```

Ahora verás tus archivos de Google Drive en el panel izquierdo (explorador de archivos). Haga click derecho en el archivo que necesita importar y seleccione copiar ruta. Luego, importe como de costumbre en pandas y pegue la ruta copiada

Listing 28: Import

```
import pandas as pd  df=pd.
read_csv('FilePath')
```

4.3. Importar de una URL

Si nuestro dataset esta en una página web, puede subirse a Google Colab copiando la URL y pegándole entre las comillas en el comando **df=pd.read_csv("")**. Pueden agregarse otros parámetros para refinar el procesamiento de este archivo. Por ejemplo, el delimitador es una secuencia de caracteres que separa regiones en un texto plano.

Listing 29: Import

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Listing 30: Import

```
df = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv", delimiter=";")

df2 = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv", delimiter=";")
```

Listing 31: Import

```

df
```

Resultado:

	Fixed acidity	volatitle acidity	citric acid	...	alcohol	quality
0	7.4	0.7000	0.00	...	9.4	5
1	7.8	0.880	0.00	...	9.8	5
2	7.8	0.760	0.04	...	9.8	5
3	11.2	0.280	0.56	...	9.8	6
4	7.4	0.700	0.00	...	9.4	5
...	...	...	...	...	...	...
1594	6.2	0.600	0.08	...	10.5	5
1595	5.9	0.600	0.08	...	11.2	6
1596	6.3	0.510	0.13	...	11.0	6
1597	5.9	0.645	0.12	...	10.2	5

Cuadro 1: Ejemplo ilustrativo del comando **df = pd.read_csv()**

4.4. Importar módulos que no esten presentes en Google Colab

Podemos agregar nuevos módulos ya que estamos trabajando en la nube, con grandes computadoras pertenecientes a Google, las cuales contienen Python 3. Utilizando el simbolo "!" podemos ejecutar comandos como si estuviéramos utilizando la terminal. En el caso de Jupiter Notebook, utilizamos un tipo de consola de linea de comando de Python llamado IPython. En esta ocasión instalaremos un paquete de estadística llamado "Pinguoin". El comando "pip install" sera utilizado para descargar e Instalar librerías externas.

Listing 32: Import

```
1          !pip intall pingouin
```

Resultado:

```
    Collecting pingouin
  Downloading https://files.pythonhosted.org/packages/e6/5f/4618
     f878765a8b7037b8831f19105c5c2764b26e5e9afa4a29c58fc11d26/pingouin-0.3.8.tar.gz (223
     kB)
    |________________________________| 225kB 7.6MB/s
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.6/dist-packages (
    from pingouin) (1.19.5)
Requirement already satisfied: scipy>=1.3 in /usr/local/lib/python3.6/dist-packages (from
     pingouin) (1.4.1).......
```

SECTION

Base de Datos y Estadísticas

5.1. Manipulación de Tablas

Haremos uso del comando “import” para importar los módulos necesarios en cada proyecto, sin la necesidad de tenerlos instalados en nuestro equipo, ya que al depender de un Notebook Online, este ya se encuentra equipado con esas herramientas.
Cargaremos un conjunto de datos preexistentes dentro del módulo Seaborn. El conjunto de datos con el que trabajemos se llama “Cuarteto Anscombe”.

Listing 33: Import

```
import pandas as pd
import seaborn as sns
#El comando "as" nos indica el nombre que le queremos dar al
   modulo para despues llamarlo de una formaabreviada.
```

Cargaremos nuestro dataset elegido y vamos a establecerlo como nuestro dataframe.

Listing 34: Import

```
df = sns.load_dataset('anscombe')
```

Listing 35: Import

```
type(df)
```

Resultado:

```
pandas.core.frame.DataFrame
```

Ahora, para visualizar los datos, se puede imprimir estos datos en forma de tabla. Una vez hecho esto, podemos ver que hay una columna que clasifica nuestros datos en 3 regiones: I, II y III. Estas presentan diferentes distribuciones. Podemos hacer un simple análisis estadísticos de cada una de estas regiones.

Listing 36: Import

```
# Veamos la forma de los datos que importamos , los notebooks son capaces de imprimir los datos en forma de tablas
df
# Notemos que existe una columna que clasifica los datos en 3 regiones I, II Y III, donde simplemente cambia la tendencia de la funcion y permite hacer analisis variados y sencillos de muestra en esos dominios.
```

Resultado:

dataset		x	y
0	I	10.0	8.04
1	I	8.0	6.95
2	I	13.0	7.58
3	I	9.0	8.81
4	I	11.0	8.33
5	I	14.0	9.96
#	#	–.-	-.–
40	IV	19.0	12.50
41	IV	8.0	5.56
42	IV	8.0	7.91
43	IV	8.0	6.89

Cuadro 2: Ejemplo ilustrativo de la ejecución del comando **df**

En nuestra tabla 2, tenemos 4 columnas. La primera columna se llama “index” y solo enumera los datos. El resto de las columnas son específicas para el conjunto de datos que elegimos.

En programación, cualquier valor, función o DataFrame se llama objeto, y cada objeto tiene una clase. Los DataFrames son un tipo de “clase”, esencialmente porque se puede hacer el mismo tipo de cosas con todos los DataFrames. Podemos decir que cada vez que llamamos la función type() a cualquier objeto de Python estamos preguntando por su clase:

Listing 37: Import

```
1     print(type(df))
```

Resultado:

```
    <class 'pandas.core.frame.DataFrame'>
```

Las clases incorporan información sobre el comportamiento. La información sobre comportamiento está contenida en métodos. En este sentido, las columnas son un método de la clase DataFrame.

Algunas funciones integradas de Pandas nos permite visualizar ciertas características de nuestro DataFrame, por ejemplo, la función columns imprime el nombre otorgado a cada conjunto de datos existentes en nuestro dataset..

Listing 38: Import

```
1 # Algunas funciones integradas de pandas permiten visualizar
     ciertas caracteristicas de nuestro DataFrame, por ejemplo, la
     funcion columns nos imprime el nombre de todas las columnas de
     la tabla.
2 df.columns
```

Resultado:

```
    Index(['dataset', 'x', 'y'], dtype='object')
```

También podemos transponer filas y columnas usando:

Listing 39: Import

```
1 df.T
```

Resultado:
Si quisiéramos que ordenar los valores numéricos según un criterio determinado, podemos utilizar la función sort_values(). Debemos tener en cuenta que tenemos que espe-

	0	1	2	3	4	5	#	40	41	42	43
dataset	I	I	I	I	I	I	#	IV	IV	IV	IV
x	10	8	13	9	11	14	#	19	8	8	8
y	8.04	6.95	7.58	8.81	8.33	9.96	#	12.5	5.56	7.91	6.89

Cuadro 3: Ejemplo ilustrativo de la función **df.T**

cificar que valores en que columnas queremos ordenar. Con el parametro "ascending" o "descending" podemos especificar si queremos ordenarlos en orden ascendente o descendente.
Esta función, como cualquiera otra, tiene parámetros adicionales que podemos cambiar según nuestro gusto o necesidades, por ejemplo, el parámetro "ascending" es "True" por defecto. Si no lo especificamos, permanecera como "True", pero si lo necesitamos, podríamos cambiarlo a "False".

En este caso, ordenaremos los valores de la columna "x" en orden ascendente.

Listing 40: Import

```
1 df.sort_values(by='x', ascending = True)
```

dataset		x	y
18	II	4.0	3.10
29	III	4.0	5.39
7	I	4.0	4.26
21	II	5.0	4.74
32	III	5.0	5.73
10	I	5.0	5.68
#	#	–.-	-.–
16	II	14.0	8.10
27	III	14.0	8.84
5	I	14.0	9.96
40	IV	19.0	12.50

Cuadro 4: Ejemplo ilustrativo en el cual se coloca la variable "x" de menor a mayor con la funcion **df.sort_values**

También podríamos seleccionar una sola columna y generar una tabla que solo contenga el índice (index) y la columna seleccionada, como se muestra a continuación. Este tipo de tabla se llama “Serie”.

Listing 41:

```
df_X= df[' x' ]
```

Listing 42:

```
type(df_X)
```

Resultado:

```
pandas.core.series.Series
```

Listing 43:

```
df_X
```

Resultado:

0	10.0
1	8.0
2	13.0
3	9.0
4	11.0
5	14.0
6	6.0
–.-	-.–
40	19.0
41	8.0
42	8.0
43	8.0
Name: x	**dtype: float64**

Cuadro 5: Ejemplo ilustrativo de una tabla en Serie.

También podríamos seleccionar un rango de filas según su índice.

Listing 44:

```
# Podemos seleccionar rangos de filas segun su indice
df[0:3]
```

Resultado:

dataset		x	y
0	I	10.0	8.04
1	I	8.0	6.95
2	I	13.0	7.58

Cuadro 6: Ejemplo ilustrativo de una tabla mediante el uso de la función **df[0:3]**

Y podríamos combinar las dos acciones previas y seleccionar un rango de filas y también columnas específicas.

Listing 45:

```
df.loc[5:9, ['x', 'y']]
```

Resultado:

	x	y
5	14.0	9.96
6	6.0	7.24
7	4.0	4.26
8	12.0	10.84
9	7.0	4.82

Cuadro 7: Ejemplo ilustrativo de la funcion **df.loc[]**

También podemos usar operadores de comparación o booleanos en los valores de la columnas. Este código, por ejemplo, crea una serie que muestra que filas de la columna "x" tienen un valor superior a 8.

Listing 46:

```
df['x'] > 8
```

Resultado:

0	True
1	False
2	True
3	True
4	True
5	True
6	False
–.-	-.–
40	True
41	False
42	False
43	False
Name: x	**dtype: bool**

Cuadro 8: Ejemplo ilustrativo de Serie tomando como “True” un dato mayor a 8

La Serie creada anteriormente puede ser usada como un filtro del conjunto de datos original. Este código crea un DataFrame que solo contiene aquellas filas del DataFrame original que tengan un valor superior a 8 en la columna “x”.

Listing 47:

```
df[df['x']>8]
```

dataset		x	y
0	I	10.0	8.04
2	I	13.0	7.58
3	I	9.0	8.81
4	I	11.0	8.33
5	I	14.0	9.96
8	I	12.0	10.84
#	#	–.-	-.–
26	III	11.0	7.81
27	III	14.0	8.84
30	III	12.0	8.15
40	IV	19.0	12.50

Cuadro 9: Ejemplo ilustrativo de un filtro por parámetros

5.2. Funciones

También podemos filtrar valores no numéricos, por ejemplo, el siguiente código crea un DataFrame que solo contiene las filas de la región definida como “I”.

Listing 48:

```
#Resulta conveniente filtrar los datos por su tipo, es decir las diferentes regiones mencionadas antes de tipo I, II, III.
# Este filtrado se puede realizar con el modulo de pandas , que permite filtrar los datos colocando ″condiciones″ como indices , en este caso la condicion de que la columna ″dataset″ tenga el valor I, II o III.

df_I = df[ df[' dataset ']== ' I']
df_II = df[ df[' dataset ']== ' II ']
df_III = df[df[' dataset']==' III']
df_IV = df[df[' dataset']==' IV']

df_I
```

dataset		x	y
0	I	10.0	8.04
1	I	8.0	6.95
2	I	13.0	7.58
3	I	9.0	8.81
4	I	11.0	8.33
5	I	14.0	9.96
6	I	6.0	7.24
7	I	4.0	4.26
8	I	12.0	10.84
9	I	7.0	4.82
10	I	5.0	5.68

Cuadro 10: Ejemplo ilustrativo de filtrado por parámetros no numéricos

Definamos una función que nos permita mostrar varias tablas una al lado de la otra. A esto lo hacemos solo con fines de demostración, para saber cómo escribir una función y utilizarla. Para esta función que queremos crear, importaremos `display_html` de `IPython.display`.

Listing 49:

```
from IPython.display import display_html
def display_side_by_side(*args):
    html_str=
    for df in arags:
     html_str        +=         df.         to_html        ()
    display_html(html_str.replace('table','table style="display:
       inline"'),raw=True)
```

Dentro de esta función estoy definiendo una nueva variable: `html_str`, y asignándole un valor de texto en blanco, luego le digo a la computadora que tome el DataFrame (`df`) y lo convertia a una tabla HTML (la misma tabla en un formato diferente), agregando la tabla (ahora en formato HTML) como valor a la variable `html_str`, y mostrando esto junto con el parámetro: `table style = "display:inline"` . Ahora tenemos que "alimentar" nuestra función con los argumentos, que son las tablas que queremos mostrar:

Listing 50:

```
display_side_by_side(df_I, df_II, df_III, df_IV)
```

dataset	x	y	dataset	x	y	dataset	x	y	dataset	x	y
0 I	10.0	8.04	**11** II	10.0	9.14	**22** III	10.0	7.46	**33** IV	8.0	6.58
1 I	8.0	6.95	**12** II	8.0	8.14	**23** III	8.0	6.77	**34** IV	8.0	5.76
2 I	13.0	7.58	**13** II	13.0	8.74	**24** III	13.0	12.74	**35** IV	8.0	7.71
# I	#	#	**#** II	#	#	**#** III	#	#	**#** IV	#	#
8 I	12.0	10.84	**19** II	12.0	9.13	**30** III	12.0	8.15	**41** IV	8.0	5.56
9 I	7.0	4.82	**20** II	7.0	7.26	**31** III	7.0	6.42	**42** IV	8.0	7.91
10 I	5.0	5.68	**21** II	5.0	4.74	**32** III	5.0	5.73	**43** IV	8.0	6.89

Cuadro 11: Ejemplo ilustrativo de tabla html con parámetros de clasificación definidos

5.3. Descripción estadística de los datos

La librería Pandas tiene algunas funciones que permiten una descripción estadística de un dataset.

Exploraremos `info()`, `head()` y `tail()`.

El método `df.info()` imprime información sobre un DataFrame, incluyendo el número de columnas, el dtype (tipo de dato), cantidad de valores no nulos (Non-Null) y el uso de memoria. Esto es útil para checar que el objeto tiene el tipo de datos correcto.

Listing 51:

```
df.info()
```

El método `df.head()` imprime las primeras filas de objetos. Por default, imprime 6 archivos, pero puede indicarse un número distinto. Esto puede utilizarse para observar rápidamente algunas variables de los datos. También podemos utilizar `df.tail()`, que imprime las últimas filas.

Listing 52:

```
df.head()
```

5.4. Métodos útiles para describir estadísticamente

`max()` es un método que muestra el máximo valores de una columna o fila.

Listing 53:

```
#Buscamos el maximo de la columna pH
df['pH'].max()
```

Listing 54:

```
#Buscamos el maximo de la columna pH
df.max(axis=0)
```

Los métodos `min()`, `mean()`, `median()` y `std()` funcionan de manera análoga. Estos métodos se utilizan para calcular el mínimo, la media, la mediana y desvió estándar.

Existe una forma conveniente de describir los datos de un DataFrame. El método `df.describe()` se utiliza para visualizar algunos detalles estadísticos como los percentiles, la media, el desvió estándar, mínimo y máximo.

Listing 55:

```
df.describe()
```

Resultado:

	Fixed acidity	volatitle acidity	citric acid	...	alcohol	quality
count	1599.00	1599.00	1599.00	...	1599.00	1599.00
mean	8.31963	0.52782	0.27097	...	10.4229	5.63
std	1.741096	0.17906	0.19480	...	1.06566	0.80
min	4.60000	0.12000	0.00000	...	8.4000	3.00
25%	7.10000	0.39000	0.09000	...	9.5000	5.00
50%	7.90000	0.52000	0.26000	...	10.2000	6.00
75%	9.20000	0.64000	0.42000	...	11.1000	6.00
max	15.9000	1.58000	1.00000	...	14.9000	8.00

Cuadro 12: Ejemplo ilustrativo de la funcion **df.describe()**

5.5. Intervalo de confianza

Calculemos el intervalo de confianza alrededor de la media. En este caso lo haremos con el contenido de alcohol en vinos de calidad 3. La formula para el intervalo de confianza es:

$$CI = \text{mean} \pm z \cdot SE$$

donde z es valor-z y SE es el error estándar. El valor-z es el percentil correspondiente a la distribución normal. Valores usuales de z para distintos intervalos de confianza son:

Nivel de confianza	Valor Z
90%	1.645
95%	1.96
99%	2.575

Cuadro 13: Ejemplo ilustrativo del valor z

El error estándar de la media indica la incertidumbre alrededor de nuestra estimación:

$$SE = \frac{SD}{\sqrt{n}}$$

Utilizaremos la librería Numpy para calcular el intervalo de confianza.

Definamos los componentes necesarios para estimar el intervalo de confianza:

Listing 56:

```
import numpy as np                                df_quality
3_alcohol=df[df['quality']==3]['alcohol']mean = np.
mean(df_quality3_alcohol)
```

```
N = np_size(df_quality3_alcohol)
SD = np.std(df_quality3_alcohol)
SE = SD /np.sqrt(N)

print("mean: %.2f N: %.2f Standar Error %.2f" % (mean, N, SE))
```

Resultado:

```
mean: 9.96 N: 10.00 Standar Error: 0.25
```

Y ahora calcularemos los extremos superior e inferior del intervalo a nivel de confianza 95%, utilizando la fórmula mostrada previamente:

Listing 57:

```
lower_limit = mean -1.96*SE
upper_limit = mean +196*SE

print("lower limit: \t%.2f \nmean: \t\t%.2f \nupper limit: \t%.2f
    " % (lower_limit, mean, upper_limit))
```

Resultado:

```
lower limit: 9.47
mean:        9.96
upper limit: 10.44
```

La función `print()` permite personalizar la salida. En los ejemplos previos, utilizamos para colocar una tabulación y un final de linea, respectivamente. Además, utilizando el comando`%.2f` el resultado solo muestra dos decimales.

SECTION

Graficado de Datos

La representación gráfica de los datos es una parte esencial de Machine Learning. Los gráficos nos permite interpretar los resultados y comunicar de manera visual nuestros hallazgos.

6.1. Gráficos Simples

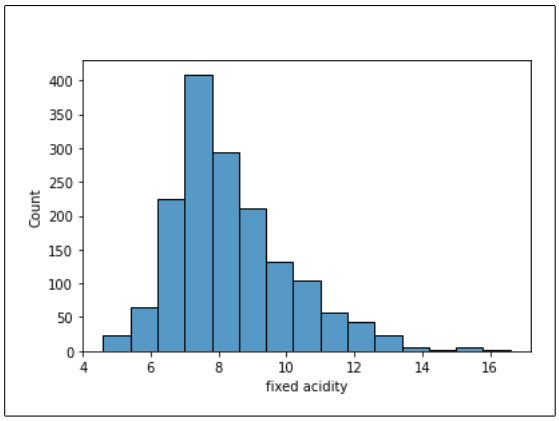
Seaborn nos permite visualizar datos de cualquier DataFrame. Su simple sintaxis es ideal para hacer fácilmente una gran variedad de gráficos, tales como gráficos de dispersión, lineales, histogramas, etc. En esta sección introduciremos cómo hacer algunos gráficos básicos a partir de los datos de nuestro dataset.
Para hacerlos, podemos usar, como estuvimos haciendo en las secciones anteriores, las funciones integradas al módulo. En este caso haremos un gráfico de dispersión.

Listing 58:

```
1 sns.histplot(x="fixed acidity", binwidth=0.8, data=df)
```

Resultado:

```
    <matplotlib.axes._subplots.AxesSubplot at 0x7f60dc6f54a8>
```



Cuadro 14: Historigrama referente a una variable de nuestra base de datos

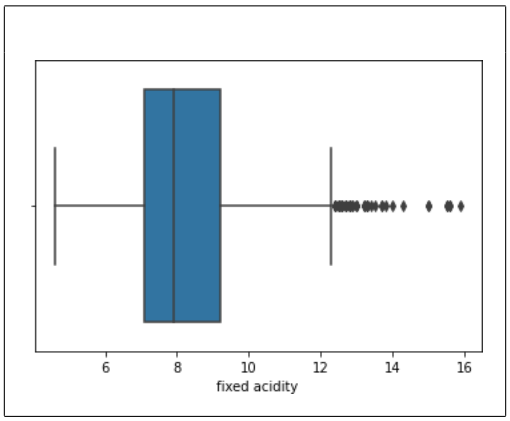
Otra forma de mostrar los datos es utilizando el boxplot. En este se muestra el valor mínimo, máximo, mediana, primer cuartil(percentil 25, extremo inferior de la caja) y tercer cuartil(percentil 75, extremo superior de la caja).

Listing 59:

```
sns.boxplot(x="fixed  acidity",  data=df)
```

Resultado:

```
<matplotlib.axes_subplots.AxesSubplot at 0x7f60d31e2b38>
```



Cuadro 15: Boxplot referente a una variable de nuestra base de datos

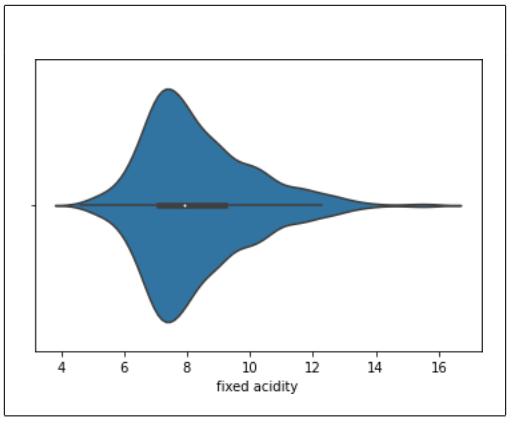
El gráfico de violín combina la ventaja de graficar los cuartiles(igual que en el boxplot) y la distribución de la variable numérica(como en el histograma). Para darle una forma suave, se utiliza una estrategia denominada “Kernel Density Estimación” o por su siglas “KDE”.

Listing 60:

```
sns.violinplot(x="fixed  acidity",  data=df)
```

Resultado:

```
<matplotlib.axes_subplots.AxesSubplot at 0x7f60d316d7f0>
```



Cuadro 16: Gráfica de violín referente a una variable de nuestra base de datos

6.2. Gráficos simples Seaborn

Seaborn nos permite visualizar datos de cualquier DataFrame. Su simple sintaxis es ideal para realizar fácilmente una gran variedad de gráficos, de dispersión, lineales, histogramas, etc. En esta sección introduciremos cómo hacer algunos gráficos básicos a partir de los datos de nuestra dataset.

Para hacerlos, podemos usar, las funciones integradas al módulo. En este caso, haremos un gráfico de dispersión. La función crea una figura (con un tamaño por defecto) y recibe los parámetros que queremos que trace. Por ejemplo, el siguiente código muestra como graficar los valores numéricos de las columnas “x” e “y” de nuestro DataFrame.

Listing 61:

```
1 sns.scatterplot(x="x", y="y", data=df)
```

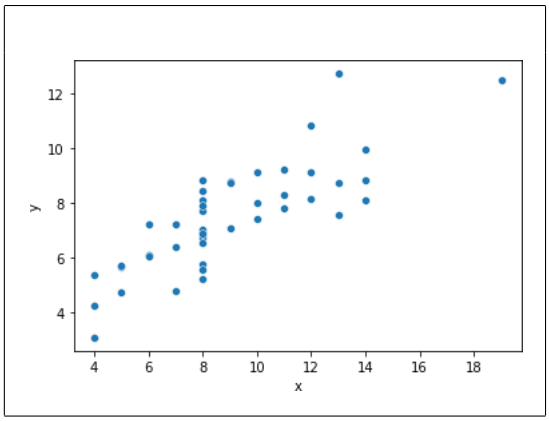
Resultado:

```
<matplotlib.axes_subplots.AxesSubplot at 0x7f48d5475c50>
```

Podemos cambiar muchas características del gráfico, como el color, los ejes, el título, el tamaño, la forma de los puntos, etc.

Por ejemplo, Seaborn nos permite añadir un parámetro llamado “hue”, que podemos utilizar para visualizar con más detalle nuestro conjunto de datos. Su función es agrupar variables y producir puntos con diferentes colores. La agrupación depende de otra columna del dataset y puede ser categórica o numérica.

Listing 62:

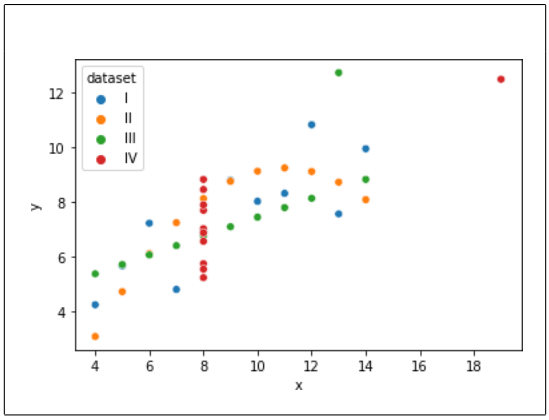


Cuadro 17: Imagen ilustrativa de un gráfico de dispersión

```
sns.scatterplot(x="x", y="y", hue="dataset", data=df)
```

Resultado:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f48d5463e90>
```



Cuadro 18: Graficado de dispersión con clasificación de datos por colores

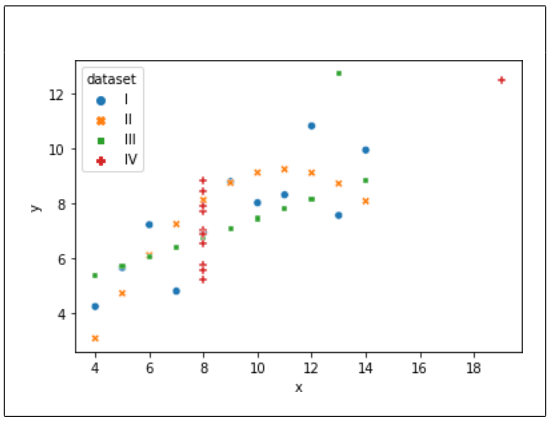
El parámetro “style” también nos permite agrupar datos según otra columna, funciona de la misma manera que “hue”, pero en lugar de cambiar el color de los puntos, cambia su forma.

Listing 63:

```
sns.scatterplot(x="x", y="y", hue="dataset", style="dataset",
    data=df)
```

Resultado:

```
<matplotlib.axes_subplots.AxesSubplot at 0x7f48d536e790>
```



Cuadro 19: Representación de gráfica de dispersión con distinción de elementos

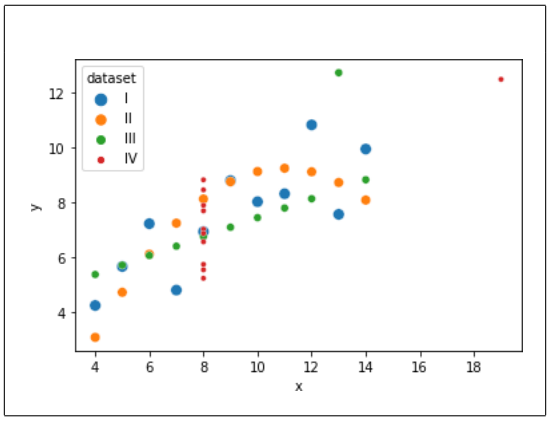
El parámetro “size” cambia el tamaño de los puntos según la agrupación.

Listing 64:

```
sns.scatterplot(x="x", y="y", hue="dataset", size="dataset", data=df)
```

Resultado:

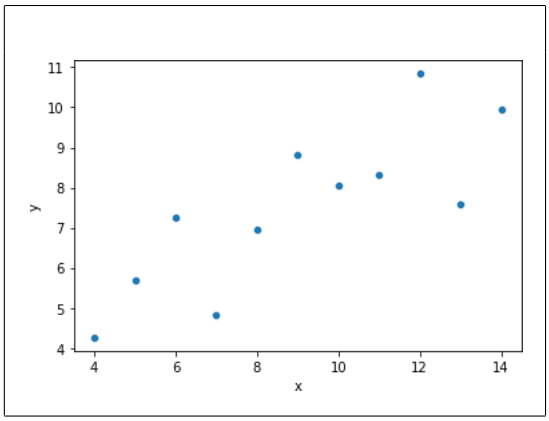
```
<matplotlib.axes_subplots.AxesSubplot at 0x7f48d5307950>
```



También podemos graficar conjuntos de datos filtrados de forma análoga a como hicimos anteriormente:

Listing 65:

```
plot=sns.scatterplot(x="x", y="y", data=df[df['dataset']=='I'])
```



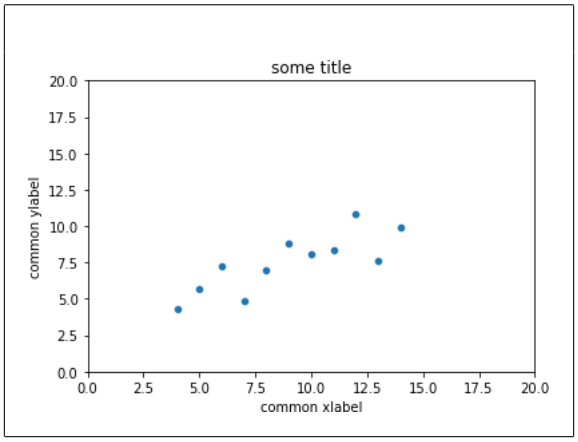
Podemos guardar el gráfico como una variable, de esta manera, podemos cambiar algunas de sus características más tarde, como ser los valores límite de los ejes, por ejemplo:

Listing 66:

```
plot=sns.scatterplot(x="x", y="y", data=df[df['dataset']=='I'])
plot.set(xlim=(0, 20), xlabel='common xlabel', ylim=(0,20),
   ylabel='common ylabel', title='some title')
```

Resultado:

```
[(0.0, 20.0),
Text(0, 0.5, 'common ylabel'),
(0.0, 20.0),
Text(0.5, 0, 'common xlabel'),
Text(0.5, 0, 'some title')]
```



SECTION

Bibliografía

Referencias

1) Mardirossian, N.; Head-Gordon, M. *J. Chem. Theory Comput.* **2016**, *12*, 4303–4325.
2) Goerigk, L.; Hansen, A.; Bauer, C.; Ehrlich, S.; Najibi, A.; Grimme, S. *Phys. Chem. Chem. Phys.* **2017**, *19* (48), 32184–32215.
3) Mardirossian, N.; Head-Gordon, M. *Mol. Phys.* **2017**, *115* (19), 2315–2372.
4) Yu, H. S.; Zhang, W.; Verma, P.; He, X.; Truhlar, D. G. *Phys. Chem. Chem. Phys.* **2015**, *17* (18), 12146–12160.
5) Yu, H. S.; He, X.; Truhlar, D. G. *J. Chem. Theory Comput.* **2016**, *12* (3), 1280–1293.
6) Yu, H. S.; He, X.; Li, S. L.; Truhlar, D. G. *Chem. Sci.* **2016**, *7* (8), 5032–5051.
7) Mehta, N.; Casanova-Páez, M.; Goerigk, L. *Phys. Chem. Chem. Phys.* **2018**, *20* (36), 23175–23194.
8) Najibi, A.; Goerigk, L. *J. Chem. Theory Comput.* **2018**, *14* (11), 5725–5738.
9) Rappoport, D.; Crawford, N. R. M.; Furche, F.; Burke, K. “Approximate Density Functionals: Which Should I Choose?” In *Encyclopedia of Inorganic and Bioinorganic Chemistry*; King, R. B., Crabtree, R. H., Lukehart, C. M., Atwood, D. A., Scott, R. A., Eds.; John Wiley & Sons, Ltd: Chichester, UK, **2009**.
10) Cramer, C. J.; Truhlar, D. G. *Phys. Chem. Chem. Phys.* **2009**, *11* (46), 10757–10816.
11) Goerigk, L.; Mehta, N. *Aust. J. Chem.* **2019**, *72* (8), 563–573.
12) Peverati, R.; Truhlar, D. G. *Philos. Trans. R. Soc. A* **2014**, *372*, 20120476.
13) Yu, H. S.; Li, S. L.; Truhlar, D. G. *J. Chem. Phys.* **2016**, *145* (13), 130901.
14) Perdew, J. P.; Schmidt, K. *AIP Conf. Proc.* **2001**, *577*, 1–20.
15) Koch, W.; Holthausen, M. C. “A Chemist’s Guide to Density Functional Theory”, 2nd ed.; John Wiley & Sons, **2001**.
16) Kozuch, S.; Bachrach, S. M.; Martin, J. M. L. *J. Phys. Chem. A* **2014**, *118* (1), 293–303.
17) Friedrich, J.; Hänchen, J. *J. Chem. Theory Comput.* **2013**, *9* (12), 5381–5394.
18) Friedrich, J. *J. Chem. Theory Comput.* **2015**, *11* (8), 3596–3609.
19) Slater, J. C. *Phys. Rev.* **1951**, *81* (3), 385–390.
20) Perdew, J. P.; Wang, Y. *Phys. Rev. B* **1992**, *45* (23), 13244–13249.
21) Perdew, J. P.; Wang, Y. *Phys. Rev. B* **2018**, *98* (7), 079904. (Erratum)
22) Vosko, S. H.; Wilk, L.; Nusair, M. *Can. J. Phys.* **1980**, *58* (8), 1200–1211.
23) Becke, A. D. *Phys. Rev. A* **1988**, *38* (6), 3098–3100.
24) Perdew, J. P. *Phys. Rev. B* **1986**, *33* (12), 8822–8824.
25) Perdew, J. P. *Phys. Rev. B* **1986**, *34* (10), 7406–7406. (Erratum)
26) Lee, C.; Yang, W.; Parr, R. G. *Phys. Rev. B* **1988**, *37*, 785–789.
27) Becke, A. D. *J. Chem. Phys.* **1993**, *98* (7), 5648–5652.
28) Stephens, P. J.; Devlin, F. J.; Chabalowski, C. F.; Frisch, M. J. *J. Phys. Chem.* **1994**, *98* (45), 11623–11627.
29) Perdew, J. P.; Chevary, J. A.; Vosko, S. H.; Jackson, K. A.; Pederson, M. R.; Singh, D. J.; Fiolhais, C. *Phys. Rev. B* **1992**, *46* (11), 6671–6687.
30) Perdew, J. P.; Burke, K.; Ernzerhof, M. *Phys. Rev. Lett.* **1996**, *77*, 3865–3868.
31) Adamo, C.; Barone, V. *J. Chem. Phys.* **1999**, *110* (13), 6158–6170.

32) Ernzerhof, M.; Scuseria, G. E. *J. Chem. Phys.* **1999**, *110* (11), 5029–5036.
33) Zhao, Y.; Truhlar, D. G. *Theor. Chem. Acc.* **2008**, *120* (1), 215–241.
34) Peverati, R.; Truhlar, D. G. *J. Phys. Chem. Lett.* **2011**, *2*, 2810–2817.
35) Mardirossian, N.; Head-Gordon, M. *J. Chem. Phys.* **2015**, *142* (7), 074111 32.
36) Vydrov, O. A.; van Voorhis, T. *J. Chem. Phys.* **2010**, *133* (24), 244103.
37) Mardirossian, N.; Head-Gordon, M. *J. Chem. Phys.* **2016**, *144* (21), 214110.
38) Chai, J.-D.; Head-Gordon, M. **2008**, *10* (44), 6615–6620.
39) Hartree, D. R. *Math. Proc. Camb. Philos. Soc.* **1928**, *24* (1), 89–110.
40) Hartree, D. R. *Math. Proc. Camb. Philos. Soc.* **1928**, *24* (1), 111–132.
41) Fock, V. *Z. Für Phys.* **1930**, *61* (1–2), 126–148.
42) Slater, J. C. *Phys. Rev.* **1930**, *35* (2), 210–211.
43) Møller, C.; Plesset, M. S. *Phys. Rev.* **1934**, *46* (7), 618–622.
44) Weigend, F.; Ahlrichs, R. *Phys. Chem. Chem. Phys.* **2005**, *7* (18), 3297–3305.
45) Goerigk, L.; Grimme, S. *J. Chem. Theory Comput.* **2010**, *6* (1), 107–126.
46) Johnson, E. R.; Mori-Sánchez, P.; Cohen, A. J.; Yang, W. *J. Chem. Phys.* **2008**, *129* (20), 204112.
47) Grimme, S.; Antony, J.; Ehrlich, S.; Krieg, H. *J. Chem. Phys.* **2010**, *132* (15), 154104.
47) Curtiss, L. A.; Raghavachari, K.; Redfern, P. C.; Pople, J. A. *J. Chem. Phys.* **1997**, *106* (3), 1063–1079.
48) Karton, A.; Gruzman, D.; Martin, J. M. L. *J. Phys. Chem. A* **2009**, *113* (29), 8434–8447.
49) Řezáč, J.; Riley, K. E.; Hobza, P. *J. Chem. Theory Comput.* **2011**, *7* (8), 2427–2438.
50) Řezáč, J.; Riley, K. E.; Hobza, P. *J. Chem. Theory Comput.* **2011**, *7* (11), 3466–3470.
51) Grimme, S. *J. Chem. Phys.* **2006**, *124* (3), 034108.
52) Grimme, S.; Mück-Lichtenfeld, C.; Würthwein, E.-U.; Ehlers, A. W.; Goumans, T. P. M.; Lammertsma, K. *J. Phys. Chem. A* **2006**, *110* (8), 2583–2586.
53) Piacenza, M.; Grimme, S. *J. Comput. Chem.* **2004**, *25* (1), 83–99.
54) Woodcock, H. L.; Schaefer III, H. F.; Schreiner, P. R. *J. Phys. Chem. A* **2002**, *106* (49), 11923–11931.
55) Schreiner, P. R.; Fokin, A. A.; Pascal, R. A.; de Meijere, A. *Org. Lett.* **2006**, *8* (17), 3635–3638.
56) Lepetit, C.; Chermette, H.; Gicquel, M.; Heully, J.-L.; Chauvin, R. *J. Phys. Chem. A* **2007**, *111* (1), 136–149.
57) Lee, J. S. *J. Phys. Chem. A* **2005**, *109* (51), 11927–11932.
58) Karton, A.; Martin, J. M. L. *Mol. Phys.* **2012**, *110* (19–20), 2477–2491.
59) Zhao, Y.; Tishchenko, O.; Gour, J. R.; Li, W.; Lutz, J. J.; Piecuch, P.; Truhlar, D. G. *J. Phys. Chem. A* **2009**, *113* (19), 5786–5799.
60) Manna, D.; Martin, J. M. L. *J. Phys. Chem. A* **2015**, *120* (1), 153–160.
61) Yu, H.; Truhlar, D. G. *J. Chem. Theory Comput.* **2015**, *11*, 2968–2983.
62) Peverati, R.; Truhlar, D. G. *J. Chem. Theory Comput.* **2012**, *8*, 2310–2319.
63) Karton, A.; Sylvetsky, N.; Martin, J. M. L. *J. Comput. Chem.* **2017**, *38* (24), 2063–2075.
64) Kristyán, S.; Pulay, P. *Chem. Phys. Lett.* **1994**, *229* (3), 175–180.

65) Hobza, P.; Sponer, J.; Reschel, T. *J. Comput. Chem.* **1995**, *16* (11), 1315–1325.
66) Allen, M. J.; Tozer, D. J. *J. Chem. Phys.* **2002**, *117* (24), 11113–11120.
66) Cohen, A. J.; Mori-Sánchez, P.; Yang, W. *Chem. Rev.* **2012**, *112* (1), 289–320.
67) Grimme, S. *J. Comput. Chem.* **2006**, *27* (15), 1787–1799.
68) Caldeweyher, E.; Ehlert, S.; Hansen, A.; Neugebauer, H.; Spicher, S.; Bannwarth, C.; Grimme, S. *J. Chem. Phys.* **2019**, *150* (15), 154122.
69) Dion, M.; Rydberg, H.; Schröder, E.; Langreth, D. C.; Lundqvist, B. I. *Phys. Rev. Lett.* **2004**, *92* (24), 246401.
70) Lee, K.; Murray, É. D.; Kong, L.; Lundqvist, B. I.; Langreth, D. C. *Phys. Rev. B* **2010**, *82* (8), 081101.
71) Kruse, H.; Goerigk, L.; Grimme, S. *J. Org. Chem.* **2012**, *77* (23), 10824–10834.
72) Goerigk, L.; Grimme, S. *Phys. Chem. Chem. Phys.* **2011**, *13* (14), 6670–6688.
73) Goerigk, L. *J. Phys. Chem. Lett.* **2015**, *6* (19), 3891–3896.
74) Hopmann, K. H. *Organometallics* **2019**, *38* (3), 603–605.
75) Jensen, F. *WIREs Comput. Mol. Sci.* **2013**, *3* (3), 273–295.
76) Nagy, B.; Jensen, F. Basis Sets in Quantum Chemistry. In *Reviews in Computational Chemistry*; Parrill, A. L., Lipkowitz, K. B., Eds.; John Wiley & Sons, Inc.: Hoboken, NJ, USA, **2017**; pp 93–149.Witte, J.; Neaton, J. B.; Head-Gordon, M. *J. Chem. Phys.* **2016**, *144* (19), 194306.
77) Marshall, M. S.; Burns, L. A.; Sherrill, C. D. *J. Chem. Phys.* **2011**, *135* (19), 194102.
78) Vydrov, O. A.; van Voorhis, T. *J. Chem. Theory Comput.* **2012**, *8* (6), 1929–1934.
79) de Lange, K. M.; Lane, J. R. *J. Chem. Phys.* **2011**, *134* (3), 034301–10.
80) McMahon, J. D.; Lane, J. R. *J. Chem. Phys.* **2011**, *135* (15), 154309–10.
81) Dunning, T. H. *J. Chem. Phys.* **1989**, *90*, 1007–1023.
82) Woon, D. E.; Dunning Jr., T. H. *J. Chem. Phys.* **1993**, *98* (2), 1358–1371.
83) Woon, D. E.; Dunning, T. H. *J. Chem. Phys.* **1994**, *100* (4), 2975–2988.
84) Wilson, A. K.; Woon, D. E.; Peterson, K. A.; Dunning, T. H. *J. Chem. Phys.* **1999**, *110* (16), 7667–7676.
85) Koput, J.; Peterson, K. A. *J. Phys. Chem. A* **2002**, *106* (41), 9595–9599.
86) Balabanov, N. B.; Peterson, K. A. *J. Chem. Phys.* **2005**, *123* (6), 064107.
87) Prascher, B. P.; Woon, D. E.; Peterson, K. A.; Dunning, T. H.; Wilson, A. K. *Theor. Chem. Acc.* **2011**, *128* (1), 69–82.
88) van Duijneveldt, F. B.; van Duijneveldt-van de Rijdt, J. G. C. M.; van Lenthe, J. H. *Chem. Rev.* **1994**, *94* (7), 1873–1885.
89) Boys, S. F.; Bernardi, F. *Mol. Phys.* **1970**, *19* (4), 553–566.
90) Jensen, F. “Introduction to Computational Chemistry”, 3rd edition.; John Wiley & Sons, **2017**; pages 226–228.
91) Kruse, H.; Grimme, S. *J. Chem. Phys.* **2012**, *136* (15), 154101.
92) Galano, A.; Alvarez–Idaboy, J. R. *J. Comput. Chem.* **2006**, *27* (11), 1203–1210.
93) Jensen, F. *J. Chem. Theory Comput.* **2010**, *6* (1), 100–106.
94) Liedl, K. R. *J. Chem. Phys.* **1998**, *108* (8), 3199–3204.
95) Dunning, T. H. *J. Phys. Chem. A* **2000**, *104* (40), 9062–9080.
96) Halkier, A.; Klopper, W.; Helgaker, T.; Jørgensen, P.; Taylor, P. R. *J. Chem. Phys.* **1999**, *111* (20), 9157–9167.

97) Wieczorek, R.; Haskamp, L.; Dannenberg, J. J. *J. Phys. Chem. A* **2004**, 88-*108* (32), 6713–6723.
99) Mayer, I.; Turi, L. *J. Mol. Struc. THEOCHEM* **1991**, *227*, 43–65.
100) Zhao, Y.; Truhlar, D. G. *J. Phys. Chem. A* **2006**, *110* (35), 10478–10486.
101) Curtiss, L. A.; Raghavachari, K.; Trucks, G. W.; Pople, J. A. *J. Chem. Phys.* **1998**, *94* (11), 7221–7230.
102) Jensen, F. *J. Phys. Chem. A* **2007**, *111* (44), 11198–11204.
103) Jensen, F. *J. Chem. Phys.* **2001**, *115* (20), 9113–9125.
104) Jensen, F. *J. Chem. Phys.* **2002**, *116* (8), 3502–3502. (Erratum)
105) Jensen, F.; Helgaker, T. *J. Chem. Phys.* **2004**, *121* (8), 3463–3470.
106) Jensen, F. *J. Chem. Phys.* **2012**, *136* (11), 114107.
107) Ditchfield, R.; Hehre, W. J.; Pople, J. A. *J. Chem. Phys.* **1971**, *54* (2), 724–728.
108) Hehre, W. J.; Ditchfield, R.; Pople, J. A. *J. Chem. Phys.* **1972**, *56* (5), 2257–2261.
109) Hariharan, P. C.; Pople, J. A. *Theor. Chim. Acta* **1973**, *28* (3), 213–222.
110) Dill, J. D.; Pople, J. A. *J. Chem. Phys.* **1975**, *62* (7), 2921–2923.
111) Binkley, J. S.; Pople, J. A. *J. Chem. Phys.* **1977**, *66* (2), 879–880.
112) Francl, M. M.; Pietro, W. J.; Hehre, W. J.; Binkley, J. S.; Gordon, M. S.; DeFrees, D. J.; Pople, J. A. *J. Chem. Phys.* **1982**, *77* (7), 3654–3665.
113) Gordon, M. S.; Binkley, J. S.; Pople, J. A.; Pietro, W. J.; Hehre, W. J. *J. Am. Chem. Soc.* **1982**, *104* (10), 2797–2803.
114) Rassolov, V. A.; Pople, J. A.; Ratner, M. A.; Windus, T. L. *J. Chem. Phys.* **1998**, *109* (4), 1223–1229.
115) Rassolov, V. A.; Ratner, M. A.; Pople, J. A.; Redfern, P. C.; Curtiss, L. A. *J. Comput. Chem.* **2001**, *22* (9), 976–984.
116) Xu, X.; Truhlar, D. G. *J. Chem. Theory Comput.* **2012**, *8* (1), 80–90.
117) Feller, D. *J. Comput. Chem.* **1996**, *17* (13), 1571–1586.
118) Schuchardt, K. L.; Didier, B. T.; Elsethagen, T.; Sun, L.; Gurumoorthi, V.; Chase, J.; Li, J.; Windus, T. L. *J. Chem. Inf. Model.* **2007**, *47* (3), 1045–1052.
119) Pritchard, B. P.; Altarawy, D.; Didier, B.; Gibson, T. D.; Windus, T. L. *J. Chem. Inf. Model.* **2019**, *59* (11), 4814–4820.
120) https://www.basissetexchange.org. (Accessed May 2020)
121) Zheng, J.; Xu, X.; Truhlar, D. G. *Theor. Chem. Acc.* **2011**, *128* (3), 295–305.
122) Papajak, E.; Truhlar, D. G. *J. Chem. Theory Comput.* **2011**, *7* (1), 10–18.
123) Kohn, W.; Sham, L. *Phys. Rev.* **1965**, *140* (4A), A1133–A1138.
124) Hohenberg, P.; Kohn, W. *Phys. Rev.* **1964**, *136* (3B), B864–B871.
125) Becke, A. D. *J. Chem. Phys.* **1988**, *88* (4), 2547–2553.
126) Wheeler, S. E.; Houk, K. N. *J. Chem. Theory Comput.* **2009**, *6* (2), 395–404.
127) Mardirossian, N.; Head-Gordon, M. *J. Chem. Theory Comput.* **2013**, *9* (10), 4453–4461.
128) For example, Q-Chem 5.2 uses: a (50,194) Grid for LDA, and for Most GGAs; a (75,302) Grid for MGGAs, and the B95- or B97-Based Functionals; a (99,590) Grid for the Minnesota Functionals. See Page 160 of the pdf version of the manual or manual for Q-Chem 5.2: https://manual.q-chem.com/5.2. (Accessed May 2020).
129) Tang, K. T.; Toennies, J. P. *J. Chem. Phys.* **2003**, *118* (11), 4976–4983.
130) Zhao, Y.; Schultz, N. E.; Truhlar, D. G. *J. Chem. Theory Comput.* **2006**, *2* (2), 364–382.

131) Zhao, Y.; Truhlar, D. G. *J. Chem. Phys.* **2006**, *125* (19), 194101.
132) Zhao, Y.; Truhlar, D. G. *J. Phys. Chem. A* **2006**, *110* (49), 13126–1130.
133) Peverati, R.; Truhlar, D. G. M11-L: *J. Phys. Chem. Lett.* **2012**, *3*, 117–124.
134) van Voorhis, T.; Scuseria, G. E. *J. Chem. Phys.* **1998**, *109* (2), 400–410.
135) van Voorhis, T.; Scuseria, G. E. *J. Chem. Phys.* **2008**, *129* (21), 219901.(Erratum)
136) Gori-Giorgi, P.; Savin, A. *J. Phys. Conf. Ser.* **2008**, *117*, 012017.137) Savin, A. *Chem. Phys.* **2009**, *356* (1–3), 91–97.
138) Bootsma, A. N.; Wheeler, S. E. "Popular Integration Grids Can Result in Large Errors in DFT-Computed Free Energies", **2019**, DOI: 10.26434/chemrxiv.8864204.v5, ChemRxiv preprint.
139) Seeger, R.; Pople, J. A. *J. Chem. Phys.* **1977**, *66* (7), 3045–3050.
140) Bauernschmitt, R.; Ahlrichs, R. *J. Chem. Phys.* **1996**, *104* (22), 9047–9052.
141) Fukutome, H. *Int. J. Quantum Chem.* **1981**, *20* (5), 955–1065.
142) Hoyer, C. E.; Li Manni, G.; Truhlar, D. G.; Gagliardi, L. *J. Chem. Phys.* **2014**, *141* (20), 204309.
143) Luo, S.; Averkiev, B. B.; Yang, K. R.; Xu, X.; Truhlar, D. G. *J. Chem. TheoryComput.* **2014**, *10* (1), 102–121.
144) Yu, H.; Truhlar, D. G. *J. Chem. Theory Comput.* **2014**, *10* (6), 2291–2305.
145) Luo, S.; Truhlar, D. G. *J. Chem. Theory Comput.* **2012**, *8* (11), 4112–4126.146) Cao, L.; Ryde, U. *Phys. Chem. Chem. Phys.* **2019**, *21* (5), 2480–2488.
147) Peverati, R.; Truhlar, D. G. *J. Chem. Phys.* **2011**, *135* (19), 191102.
148) Zhang, W.; Truhlar, D. G.; Tang, M. *J. Chem. Theory Comput.* **2013**, *9* (9), 3965–3977.
149) Averkiev, B. B.; Zhao, Y.; Truhlar, D. G. *J. Mol. Catal. Chem.* **2010**, *324* (1–2),80–88.
150) Xu, X.; Zhang, W.; Tang, M.; Truhlar, D. G. *J. Chem. Theory Comput.* **2015**, *11* (5), 2036–2052.
151) Karton, A.; Daon, S.; Martin, J. M. L. *Chem. Phys. Lett.* **2011**, *510* (4–6), 165–178.
152) Hait, D.; Rettig, A.; Head-Gordon, M. *J. Chem. Phys.* **2019**, *150* (9), 094115.
153) He, Y.; Gräfenstein, J.; Kraka, E.; Cremer, D. *Mol. Phys.* **2000**, *98*, 1639–1658.
154) Cremer, D. *Mol. Phys.* **2001**, *99* (23), 1899–1940.
155) Polo, V.; Kraka, E.; Cremer, D. *Theor. Chem. Acc.* **2002**, *107* (5), 291–303.
156) Mok, D. K. W.; Neumann, R.; Handy, N. C. *J. Phys. Chem.* **1996**, *100* (15),

6225–6230.
157) Handy, N. C.; Cohen, A. J. *Mol. Phys.* **2001**, *99*, 403–412.
158) Ryu, H.; Park, J.; Kim, H. K.; Park, J. Y.; Kim, S.-T.; Baik, M.-H. *Organometallics*
2018, *37* (19), 3228–3239.
159) Seybold, P. G.; Shields, G. C. *WIREs Comput. Mol. Sci.* **2015**, *5* (3), 290–297.
160) Chan, B.; Gill, P. M. W.; Kimura, M. *J. Chem.*
Theory Comput. **2019**, *15* (6),3610–3622.
161) Chan, B. *J. Phys. Chem. A* **2019**, *123* (27), 5781–5788.
162) Dohm, S.; Hansen, A.; Steinmetz, M.; Grimme, S.;
Checinski, M. P. *J. Chem.Theory. Comput.*, **2018**, *14*,
2596–2608.