Capítulo 4

Descriptores Moleculares

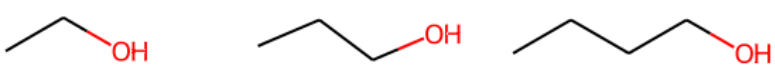
Los descriptores moleculares son la representación numérica de las moléculas, constituyen una de las herramientas fundamentales en la química computacional moderna, ya que permiten traducir la información estructural de una molécula en valores numéricos que pueden ser analizados, comparados y utilizados en modelos estadísticos o de aprendizaje automático. Estos descriptores capturan propiedades fisicoquímicas, topológicas, geométricas y electrónicas, facilitando la relación entre la estructura molecular y su comportamiento en distintos contextos, como la actividad biológica, la reactividad química o la afinidad por un receptor.

RDKit proporciona un conjunto amplio de descriptores moleculares que permiten caracterizar una molécula a partir de su estructura sin necesidad de información experimental adicional. Para un nivel introductorio, es suficiente familiarizarse con aquellos descriptores que describen propiedades fisicoquímicas generales, la composición atómica y aspectos simples de la topología molecular.

Los descriptores moleculares pueden clasificarse en diferentes categorias de acuerdo con la cantidad y el tipo de información estructural que incorporan. Una clasificación ampliamente aceptada los organiza en cinco clases, que van desde descriptores muy simples, independientes de la geometria, hasta descriptores que consideran multiples conformaciones y condiciones externas. Esta clasificación permite comprender el nivel de complejidad y el alcance de la información que describe cada tipo de descriptor.

Los descriptores de cero dimensiones (0D) corresponden a propiedades globales de la molécula que no dependen de la disposición espacial de los átomos ni de la conectividad detallada. Incluyen, por ejemplo, el peso molecular, la formula molecular, el numero total de átomos o el conteo de átomos pesados. Estos descriptores ofrecen una caracterización básica del compuesto y suelen emplearse en etapas iniciales de filtrado.

Los descriptores de una dimensión (1D) incorporan información sobre la composición molecular y la presencia de determinados tipos de átomos o grupos funcionales, sin considerar aun la topología completa de la molécula. En esta categoría se encuentran los conteos de enlaces, el numero de donadores y aceptores de enlaces de hidrógeno, y otras propiedades derivadas directamente de la formula estructural simplificada.



Los descriptores de dos dimensiones (2D) describen la topología molecular, es decir, la forma en que los átomos están conectados entre si, sin requerir coordenadas tridimensionales. Incluyen indices topológicos, conteos de anillos, aromaticidad, superficie polar topológica (TPSA) y diversos descriptores de conectividad. Estos descriptores son ampliamente utilizados debido a su bajo costo computacional y su buena capacidad para capturar información estructural relevante.

Los descriptores de tres dimensiones (3D) incorporan información espacial derivada de coordenadas tridimensionales de la molécula. Consideran la geometría molecular, la distribución de volumen y la forma del compuesto. Para su calculo es necesario generar conformaciones 3D, y ejemplos incluyen descriptores de volumen molecular, area de superficie accesible y distribuciones espaciales de carga. Estos descriptores resultan especialmente utiles para estudios relacionados con interacciones moleculares y reconocimiento molecular.

Finalmente, los descriptores de cuatro dimensiones (4D) extienden el concepto tridimensional al considerar múltiples conformaciones de una misma molécula y, en algunos casos, condiciones externas como el entorno o el tiempo. Estos descriptores buscan capturar el comportamiento dinamico del compuesto y son utilizados principalmente en estudios avanzados de modelado molecular y simulaciones, debido a su mayor complejidad computacional.

Entre los descriptores más básicos y populares se encuentra el peso molecular (*Molecular Weight*), que representa la suma de las masas atómicas de todos los átomos presentes en la molécula. Este descriptor es útil para clasificar compuestos y establecer rangos de tamaño molecular. Otro descriptor fundamental es el número de átomos pesados (*Heavy Atom Count*), que contabiliza los átomos distintos del hidrógeno y proporciona una medida sencilla de la complejidad estructural.

El coeficiente de partición octanol/agua calculado (LogP) es uno de los descriptores más empleados en química medicinal, ya que ofrece una estimación de la lipofilia de una molécula y su posible comportamiento en sistemas biológicos. Asociado a este, la superficie polar topológica (TPSA) permite evaluar la polaridad de la molécula y está relacionada con propiedades como la permeabilidad de membrana y la biodisponibilidad.

Otros descriptores básicos incluyen el número de enlaces rotables, que indica el grado de flexibilidad conformacional de la molécula, y el conteo de donadores y aceptores de enlaces de hidrógeno, ampliamente utilizados en reglas empíricas como las reglas de Lipinski. Finalmente, los anillos aromáticos y el número total de anillos ofrecen información topológica relevante para el análisis estructural y la comparación entre moléculas.

El uso conjunto de estos descriptores permite construir una primera representación cuantitativa de una molécula, sentando las bases para análisis más avanzados como la comparación de compuestos, la reducción de dimensionalidad y la aplicación de modelos de aprendizaje automático en quimioinformática.

Estos descriptores constituyen un conjunto inicial suficiente para describir cuantitativamente una molécula y son ampliamente utilizados en análisis exploratorio de datos, filtrado de compuestos y modelos predictivos básicos en química computacional.

Una vez identificados los descriptores moleculares básicos, es posible calcularlos de forma automática utilizando RDKit y Python. A partir de una representación SMILES, RDKit permite generar un objeto molecular y evaluar múltiples propiedades fisicoquímicas mediante funciones predefinidas. Este enfoque resulta especialmente útil cuando se trabaja con conjuntos de moléculas, ya que permite construir tablas de descriptores que pueden emplearse posteriormente en análisis estadísticos o modelos de aprendizaje automático.

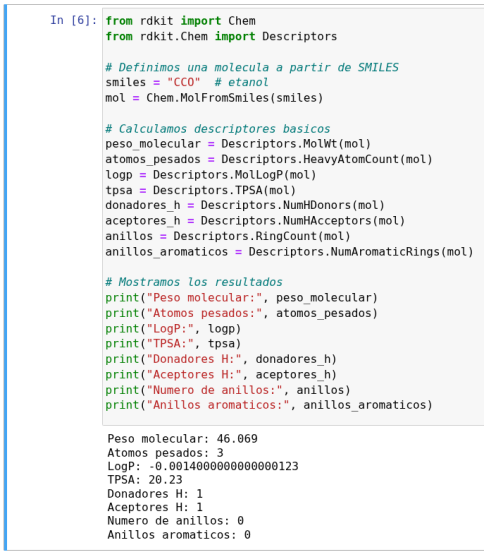
```
1    from rdkit import Chem
2    from rdkit.Chem import Descriptors
```

Cuadro 4.1: Descriptores moleculares básicos disponibles en RDKit

Descriptor	Nombre en RDKit	Descripción
Peso molecular	`MolWt`	Suma de las masas atómicas de todos los átomos de la molécula. Indica el tamaño molecular.
Número de átomos pesados	`HeavyAtomCount`	Cantidad de átomos distintos del hidrógeno presentes en la molécula.
LogP (lipofilia)	`MolLogP`	Estimación del coeficiente de partición octanol/agua; refleja la afinidad por fases lipídicas.
Superficie polar topológica	`TPSA`	Área asociada a átomos polares; relacionada con permeabilidad y biodisponibilidad.
Donadores de enlace de hidrógeno	`NumHDonors`	Número de grupos capaces de donar enlaces de hidrógeno.
Aceptores de enlace de hidrógeno	`NumHAcceptors`	Número de grupos capaces de aceptar enlaces de hidrógeno.
Número de anillos	`RingCount`	Total de anillos presentes en la estructura molecular.
Anillos aromáticos	`NumAromaticRings`	Número de anillos que presentan aromaticidad.

```
3
4       # Definimos una molecula a partir de SMILES
5       smiles = "CCO"  # etanol
6       mol = Chem.MolFromSmiles(smiles)
7
8       # Calculamos descriptores basicos
9       peso_molecular = Descriptors.MolWt(mol)
10      atomos_pesados = Descriptors.HeavyAtomCount(mol)
11      logp = Descriptors.MolLogP(mol)
12      tpsa = Descriptors.TPSA(mol)
13      donadores_h = Descriptors.NumHDonors(mol)
14      aceptores_h = Descriptors.NumHAcceptors(mol)
15      anillos = Descriptors.RingCount(mol)
16      anillos_aromaticos = Descriptors.NumAromaticRings(mol)
17
18      # Mostramos los resultados
19      print("Peso molecular:", peso_molecular)
20      print("Atomos pesados:", atomos_pesados)
21      print("LogP:", logp)
22      print("TPSA:", tpsa)
23      print("Donadores H:", donadores_h)
24      print("Aceptores H:", aceptores_h)
25      print("Numero de anillos:", anillos)
26      print("Anillos aromaticos:", anillos_aromaticos)
```

Este procedimiento puede extenderse facilmente a múltiples moléculas almacenadas en una lista de SMILES, permitiendo generar tablas completas de descriptores que pueden exportarse a formatos como CSV para su posterior análisis.



```
In [6]: from rdkit import Chem
        from rdkit.Chem import Descriptors

        # Definimos una molecula a partir de SMILES
        smiles = "CCO"  # etanol
        mol = Chem.MolFromSmiles(smiles)

        # Calculamos descriptores basicos
        peso_molecular = Descriptors.MolWt(mol)
        atomos_pesados = Descriptors.HeavyAtomCount(mol)
        logp = Descriptors.MolLogP(mol)
        tpsa = Descriptors.TPSA(mol)
        donadores_h = Descriptors.NumHDonors(mol)
        aceptores_h = Descriptors.NumHAcceptors(mol)
        anillos = Descriptors.RingCount(mol)
        anillos_aromaticos = Descriptors.NumAromaticRings(mol)

        # Mostramos los resultados
        print("Peso molecular:", peso_molecular)
        print("Atomos pesados:", atomos_pesados)
        print("LogP:", logp)
        print("TPSA:", tpsa)
        print("Donadores H:", donadores_h)
        print("Aceptores H:", aceptores_h)
        print("Numero de anillos:", anillos)
        print("Anillos aromaticos:", anillos_aromaticos)

        Peso molecular: 46.069
        Atomos pesados: 3
        LogP: -0.0014000000000000123
        TPSA: 20.23
        Donadores H: 1
        Aceptores H: 1
        Numero de anillos: 0
        Anillos aromaticos: 0
```

Figura 4.1: Visualización de este código desde Jupyter notebook.

4.1. Ejercicios

La visualización molecular es un paso fundamental en química computacional, ya que permite inspeccionar estructuras, verificar conectividad y explorar relaciones estructura–propiedad. A continuación se muestran ejemplos prácticos utilizando RDKit. En todos los casos se parte de una molécula definida mediante una cadena SMILES.

4.1.1. Descriptores de cero dimensiones (0D)

Los descriptores de cero dimensiones describen propiedades globales de la molécula sin considerar conectividad ni geometría.

```
1  from rdkit import Chem
2  from rdkit.Chem import Descriptors
3
4  mol = Chem.MolFromSmiles("CCO")
5
6  print("Peso molecular:", Descriptors.MolWt(mol))
7  print("Numero de atomos pesados:", Descriptors.HeavyAtomCount
       (mol))
```

4.1.2. Descriptores de una dimensión (1D)

Los descriptores de una dimensión incorporan información sobre la composición química y ciertos tipos de grupos funcionales.

```
1  print("Donadores de H:", Descriptors.NumHDonors(mol))
2  print("Aceptores de H:", Descriptors.NumHAcceptors(mol))
```

4.1.3. Descriptores de dos dimensiones (2D)

Los descriptores de dos dimensiones consideran la topología molecular, es decir, la conectividad entre átomos.

```
    print("LogP:", Descriptors.MolLogP(mol))
    print("TPSA:", Descriptors.TPSA(mol))
    print("Numero de anillos:", Descriptors.RingCount(mol))
```

4.1.4. Descriptores de tres dimensiones (3D)

Los descriptores de tres dimensiones requieren coordenadas espaciales y describen la forma y el volumen molecular.

```
    from rdkit.Chem import AllChem

    mol_3d = Chem.AddHs(mol)
    AllChem.EmbedMolecule(mol_3d, randomSeed=42)
    AllChem.UFFOptimizeMolecule(mol_3d)

    print("Fraccion CSP3:", Descriptors.FractionCSP3(mol_3d))
```

4.1.5. Descriptores de cuatro dimensiones (4D)

Los descriptores de cuatro dimensiones consideran múltiples conformaciones de una molécula y permiten capturar su comportamiento dinámico.

```
    conformers = AllChem.EmbedMultipleConfs(mol_3d, numConfs=5)

    print("Numero de conformaciones generadas:", len(conformers))
```

Cuadro 4.2: Descriptores moleculares básicos y su clasificación

Clase	Descriptor	Función RDKit
0D	Peso molecular	`MolWt`
0D	Átomos pesados	`HeavyAtomCount`
1D	Donadores H	`NumHDonors`
1D	Aceptores H	`NumHAcceptors`
2D	LogP	`MolLogP`
2D	TPSA	`TPSA`
2D	Anillos	`RingCount`
3D	Fracción CSP3	`FractionCSP3`
4D	Conformaciones	`EmbedMultipleConfs`

Estos ejemplos muestran como los descriptores moleculares aumentan progresivamente en complejidad a medida que se incorpora mayor información estructural. Comprender esta clasificación permite seleccionar de manera adecuada los descriptores según el objetivo del análisis, desde estudios exploratorios hasta aplicaciones avanzadas en modelado molecular y aprendizaje automático.

4.1.6. Ejercicio 1. Importación de RDKit y creación de una molécula

Este ejercicio muestra como importar RDKit y crear un objeto molecular a partir de una cadena SMILES.

```
import rdkit
from rdkit import Chem

molecula = Chem.MolFromSmiles("CCO")
molecula
```

4.1.7. Ejercicio 2. Uso explicito del modulo Draw

En este ejercicio se emplea el módulo `Draw` para generar la imagen molecular de forma explícita.

```
from rdkit import Chem
from rdkit.Chem import Draw

molecula = Chem.MolFromSmiles("CCO")
Draw.MolToImage(molecula)
```

Compara esta representación con la obtenida en el ejercicio anterior.

4.1.8. Ejercicio 3. Visualización de molécula aromática

Mediante el uso de minúsculas podemos incluir átomos en una molécula con aromatícidad.

```
from rdkit import Chem

molecula = Chem.MolFromSmiles("c1ccccc1")
molecula
```

Identifica la molécula representada y describe los átomos y enlaces que observas.

4.1.9. Ejercicio 4. Visualización de múltiples moléculas

RDKit permite visualizar varias moléculas simultáneamente utilizando listas.

```
from rdkit import Chem
from rdkit.Chem import Draw

mismoleculas = ["CCO", "CCCO", "CCCCO"]
milista = []

for i in mismoleculas:
    mol = Chem.MolFromSmiles(i)
    milista.append(mol)

Draw.MolsToGridImage(milista)
```

¿En qué casos será útil usar esto?

4.1.10. Ejercicio 5. Comparación estructural entre un alcano y un alqueno

Crea dos moléculas, una saturada y otra insaturada, y compara el número de enlaces presentes en cada una.

```
from rdkit import Chem

alcano = Chem.MolFromSmiles("CCCC")
alqueno = Chem.MolFromSmiles("C=CCC")

print("Enlaces alcano:", alcano.GetNumBonds())
print("Enlaces alqueno:", alqueno.GetNumBonds())
```

Analiza cómo la presencia de un doble enlace modifica la estructura molecular.

4.1.11. Ejercicio 6. Identificación automática de anillos

Determina si una molécula contiene uno o más anillos y cuántos son.

```
from rdkit import Chem

mol = Chem.MolFromSmiles("C1CCCCC1")

print('Tiene anillos?', mol.GetRingInfo().NumRings() > 0)
print("Numero de anillos:", mol.GetRingInfo().NumRings())
```

4.1.12. Ejercicio 7. Diferenciación entre moléculas aromáticas y no aromáticas

Evalúa el carácter aromático de los átomos en una molécula.

```
from rdkit import Chem

mol = Chem.MolFromSmiles("c1ncccc1")

for atomo in mol.GetAtoms():
    print(atomo.GetSymbol(), atomo.GetIsAromatic())
```

Discute qué átomos participan en la aromaticidad.

4.1.13. Ejercicio 8. Conteo de heteroátomos

Calcula cuántos átomos distintos de carbono e hidrógeno contiene una molécula.

```
from rdkit import Chem

mol = Chem.MolFromSmiles("CC(=O)NC")

heteroatomos = [a for a in mol.GetAtoms() if a.GetSymbol()
   not in ["C", "H"]]
print("Numero de heteroatomos:", len(heteroatomos))
```

4.1.14. Ejercicio 9. Evaluación de complejidad molecular

Calcula el índice de complejidad molecular de Bertz.

```
    from rdkit import Chem
    from rdkit.Chem import Descriptors

    mol = Chem.MolFromSmiles("CC1=CC=CC=C1O")
    complejidad = Descriptors.BertzCT(mol)

    print(complejidad)
```

Interpreta el valor obtenido en función de la estructura.

4.1.15. Ejercicio 10. Comparación de solubilidad relativa mediante LogP

Calcula el LogP de dos moléculas con diferente polaridad.

```
    from rdkit import Chem
    from rdkit.Chem import Descriptors

    mol1 = Chem.MolFromSmiles("CCOC")
    mol2 = Chem.MolFromSmiles("CC(=O)O")

    print("LogP eter:", Descriptors.MolLogP(mol1))
    print("LogP acido:", Descriptors.MolLogP(mol2))
```

Explica la diferencia observada en términos químicos.

4.1.16. Ejercicio 11. Cálculo del número de enlaces rotables

Analiza la flexibilidad conformacional de una molécula alifática ramificada.

```
    from rdkit import Chem
    from rdkit.Chem import Descriptors

    mol = Chem.MolFromSmiles("CC(C)CCO")
    print("Enlaces rotables:", Descriptors.NumRotatableBonds(mol)
        )
```

4.1.17. Ejercicio 12. Superficie polar y permeabilidad

Calcula la superficie polar topológica de una amida.

```
    from rdkit import Chem
    from rdkit.Chem import Descriptors

    mol = Chem.MolFromSmiles("CC(=O)NCC")
    tpsa = Descriptors.TPSA(mol)

    print("TPSA:", tpsa)
```

Relaciona este valor con posibles propiedades biológicas.

4.1.18. Ejercicio 13. Generación y optimización de una estructura 3D

Genera la geometría tridimensional de una molécula aromática sustituida.

```
from rdkit import Chem
from rdkit.Chem import AllChem

mol = Chem.MolFromSmiles("CC1=CC=CC=C1")
mol = Chem.AddHs(mol)

AllChem.EmbedMolecule(mol)
AllChem.UFFOptimizeMolecule(mol)

mol
```

4.1.19. Ejercicio 14. Comparación global de descriptores entre dos fármacos simples

Calcula varios descriptores y compara sus valores.

```
from rdkit import Chem
from rdkit.Chem import Descriptors

mol1 = Chem.MolFromSmiles("CC(=O)NC1=CC=CC=C1")
mol2 = Chem.MolFromSmiles("COC1=CC=CC=C1")

for mol, nombre in zip([mol1, mol2], ["Molecula A", "Molecula
    B"]):
    print(nombre)
    print("Peso molecular:", Descriptors.MolWt(mol))
    print("LogP:", Descriptors.MolLogP(mol))
    print("TPSA:", Descriptors.TPSA(mol))
    print()
```

Discute cuál de las dos sería más polar y por qué.

4.1.20. Ejercicio 15. Cálculo de cargas parciales de Gasteiger

En este ejercicio se introducen las **cargas parciales atómicas**, un descriptor electrónico fundamental en química computacional. Calcula las cargas de Gasteiger de una molécula orgánica compleja y verifica que cada átomo tenga asignado un valor.

```
from rdkit import Chem
from rdkit.Chem import AllChem

mol = Chem.MolFromSmiles(
'COc1cccc2cc(C(=O)NCCCCN3CCN(c4cccc5nccnc54)CC3)oc21'
)

AllChem.ComputeGasteigerCharges(mol)

for i, atomo in enumerate(mol.GetAtoms()):
carga = atomo.GetDoubleProp('_GasteigerCharge')
```

```
12      print(f"Atomo {i} ({atomo.GetSymbol()}): {carga:.4f}")
```

Reflexiona sobre qué átomos presentan mayor carácter negativo o positivo y por qué.

4.1.21. Ejercicio 16. Extracción de contribuciones atómicas en una lista

Una vez calculadas las cargas parciales, es posible almacenarlas en una estructura de datos para su posterior análisis o visualización. Construye una lista que contenga las cargas de Gasteiger de todos los átomos de la molécula.

```
1       contribs = [
2       mol.GetAtomWithIdx(i).GetDoubleProp('_GasteigerCharge')
3       for i in range(mol.GetNumAtoms())
4       ]
5
6       print(contribs)
```

Explica por qué el orden de esta lista es crucial para una correcta visualización posterior.

4.1.22. Ejercicio 17. Visualización de mapas de similitud basados en cargas

En este ejercicio final se integran los conceptos anteriores para **visualizar un mapa de contribuciones atómicas**, donde el color representa la influencia de cada átomo según su carga parcial.

```
1       from rdkit.Chem import Draw
2       from rdkit.Chem.Draw import SimilarityMaps
3
4       fig = SimilarityMaps.GetSimilarityMapFromWeights(
5       mol,
6       contribs,
7       colorMap='jet',
8       contourLines=10
9       )
```

Describe cómo interpretar los colores del mapa y qué información química adicional aporta esta representación frente a un dibujo molecular convencional.