

## ACTIVIDAD PRÁCTICA 2.2

### Modelización estructural de interacciones proteína-proteína (PPI)

En los ejercicios que se proponen a continuación se estudiará como caso modelo el complejo formado entre las proteínas **savinasa** de la bacteria *B. lentus* y **BASI** de cebada (Micheelsen et al. 2008). BASI (por sus siglas en inglés: *Barley  $\alpha$ -Amylase/Subtilisin Inhibitor*) es un inhibidor que se encuentra en la semilla de cebada y la protege de las proteasas secretadas por patógenos, como la savinasa, mediante la formación de un complejo de alta afinidad. Este complejo entre savinasa y BASI fue uno de los casos propuestos ("Target 32") en el experimento CAPRI (Janin et al. 2003), para el que los grupos participantes fueron invitados a enviar predicciones antes de que la estructura del complejo se hiciera pública (Pons et al. 2010).

La información de partida disponible durante el experimento CAPRI fue la siguiente:

- Estructura individual de savinasa: código **PDB 1svn**
- Estructura de BASI en complejo con  $\alpha$ -amilasa: código **PDB 1ava** (cadena C)

Dado que ya está disponible la estructura cristalográfica del complejo savinasa/BASI (código **PDB 3bx1**, cadenas A y C), podemos usar este caso para evaluar la capacidad predictiva de diferentes métodos computacionales en condiciones lo más realistas posibles.

## 1. DOCKING Y PUNTUACIÓN ENERGÉTICA: PYDOCK

**Nota:** Se utilizará pyDock (Cheng et al 2007) para llevar a cabo una ejecución completa de docking, incluyendo búsqueda de orientaciones (docking de cuerpo rígido) y puntuación energética de los modelos (*scoring*). Se empleará ZDOCK para generar orientaciones de *docking* para la interacción savinasa/BASI (caso de CAPRI número 32), pero esta vez desde pyDock, de forma totalmente automática, y evaluaremos los modelos resultantes mediante la función de puntuación de pyDock. Una vez obtenidos los resultados se analizarán los modelos de mejor energía y se compararán con la estructura del complejo de referencia.

### 1.1. Instrucciones generales de pyDock

**Nota:** En general, los trabajos de pyDock se ejecutan llamando al programa binario, y usando como primer argumento el nombre base del proyecto (se define por el usuario), y como segundo argumento, el módulo que se desee ejecutar. Por ejemplo:

```
>pyDock3 dockname module
```

En el ejemplo actual, se empleará **T32** como *dockname*. En los apartados siguientes por tanto algunos ficheros que se utilizarán llevarán este nombre base.

En cuanto al segundo argumento, en la tabla siguiente se listan los módulos que se pueden usar en PyDock con los respectivos ficheros de entrada y de salida:

	Módulo	Ficheros de entrada	Ficheros de salida
<b>Docking</b>	setup	dockname.ini	dockname_rec.pdb dockname_lig.pdb
	ftdock (or zdock)	dockname_rec.pdb dockname_lig.pdb	dockname.ftdock (or dockname.zdock)
	rotftdock (or rotzdock)	dockname.ftdock (or dockname.zdock)	dockname.rot
	dockser	dockname_rec.pdb dockname_lig.pdb dockname.rot	dockname.ene
<b>Otras funciones</b>	dockrst	dockname.ini dockname_rec.pdb dockname_lig.pdb dockname.rot dockname.ene	dockname.eneRST dockname.rst
	patch	dockname_rec.pdb dockname_lig.pdb dockname.rot dockname.ene	dockname.recNIP dockname_rec.pdb.nip dockname.ligNIP dockname_lig.pdb.nip
	oda	X.pdb Y.pdb	dockname_rec.pdb.oda dockname_rec.pdb.oda.ODAtab dockname_lig.pdb.oda dockname_lig.pdb.oda.ODAtab

**Nota:** En la siguiente página web se puede consultar documentación adicional sobre pyDock:

<https://life.bsc.es/pid/pydock/>

## 1.2. Preparación de los archivos de partida

**Nota:** Antes de llevar a cabo un proyecto de *docking* con pyDock, es necesario generar los ficheros de entrada que se emplearán en las diferentes fases del protocolo. Habrá que definir un **receptor** y un **ligando**. Normalmente, el receptor será la proteína de mayor tamaño, ya que es la que va a permanecer estática, y el ligando será la proteína que será rotada y trasladada alrededor del receptor.

En el ejemplo actual, la proteína savinasa será el receptor, y el inhibidor BASI será el ligando. Las coordenadas para dichas proteínas estarán en los ficheros PDB que ya deben haber sido descargados de [www.rcsb.org](http://www.rcsb.org) al principio de esta actividad:

- Receptor: savinasa (1svn.pdb, cadena A)
- Ligando: BASI (1ava.pdb, cadena C)

### 1.2.1. Creación de fichero de texto T32.ini

A continuación, es necesario definir un fichero "*dockname.ini*", que contendrá la información necesaria sobre los ficheros PDB de partida, y las cadenas (chain ID) que definirán el receptor y el ligando en los cálculos de *docking*. Para comparar los resultados con una estructura de referencia, es posible indicar dicha estructura en este fichero.

```
[receptor]
pdb       = 1svn.pdb
mol       = A
newmol    = A
```

```
[ligand]
pdb       = ?????
mol       = ?????
newmol    = ?????
```

```
[reference]
pdb       = ?????
recmol    = ?????
ligmol    = ?????
newrecmol = ?????
newligmol = ?????
```

**Nota:** El campo "*pdb*" tiene que corresponder exactamente con el nombre del fichero PDB de cada proteína (receptor, ligando, y si es el caso, referencia). El campo "*mol*" indica el nombre original de la cadena de interés (*chain ID*) en los ficheros de entrada indicados por "*pdb*". Por su parte, "*newmol*" indicaría el nombre que dicha proteína tendrá en los modelos de *docking*. Este campo se puede usar para renombrar las cadenas de las proteínas en los modelos de *docking*, de forma que receptor y ligando

se llamen de forma diferente, y así puedan ser fácilmente distinguibles al analizar dichos modelos.

**Nota:** En el caso de usar una estructura de referencia, mediante "*recmol*" y "*ligmol*" se deberán indicar las cadenas del fichero "*pdb*" que forman el complejo, y "*newrecmol*" y "*newligmol*" serán los nombres que tendrán las cadenas equivalentes en los modelos de *docking* (es decir, "*newmol*" en receptor y ligando, respectivamente), para que el programa los identifique automáticamente. En el ejemplo propuesto, se han ya indicado los campos que se utilizan para el receptor. El resto de campos del fichero *T32.ini* deberán completarse sustituyendo las partes indicadas con *?????*.

#### **Importante:**

Si un fichero PDB no tiene nombre definido para la cadena proteica, se usará "-" en el campo "*mol*".

Si un fichero PDB contiene varias copias para la misma proteína, se indicará solo el nombre correspondiente a una de ellas.

Si una proteína tiene varias cadenas (por ejemplo, cadenas L y H en anticuerpos), se indicarán todas ellas en el campo "*mol*", separadas por comas. En el campo "*newmol*" se podrán seguir usando los nombres originales para las dos cadenas, o elegir el mismo nombre para las dos cadenas, en ambos casos separadas por comas.

#### **1.2.2. Lanzamiento de módulo de preparación de pyDock**

Una vez completado el fichero inicial (en este caso *T32.ini*), lanza el módulo de preparación (*setup*) desde la terminal:

```
>pyDock3 T32 setup
```

**Nota:** Este comando creará nuevos ficheros PDB para receptor y ligando (en este caso *T32\_rec.pdb* y *T32\_lig.pdb*), además de otros ficheros necesarios para ejecutar los diferentes módulos de pyDock. Si se ha indicado una estructura de referencia, también se creará un nuevo fichero PDB para dicha estructura (*T32\_ref.pdb*).

En el caso de que los ficheros PDB de partida tengan las cadenas laterales incompletas, éstas serán automáticamente reconstruidas mediante el programa SCWRL 3.0, implementado en la instalación de pyDock.

#### **1.3. Lanzamiento de ZDOCK desde pyDock**

Para lanzar ZDOCK automáticamente desde pyDock puedes usar el siguiente comando:

```
>pyDock3 T32 zdock > pydock.zdock.log &  
(este cálculo durará unos 5 minutos)
```

**Nota:** Cuando termine la ejecución, las orientaciones de *docking* generadas se almacenarán en el siguiente fichero: *T32.zdock*

### 1.3.1. Transformación del fichero de salida de ZDOCK al formato pyDock

**Nota:** El fichero de salida de ZDOCK (en este caso `T32.zdock`), donde cada orientación de *docking* se representa por la posición del centro de coordenadas del ligando y su rotación basada en ángulos de Euler, deberá ser transformado en un fichero de formato adecuado para pyDock. El nuevo fichero transformado contendrá la orientación de *docking* representada por la matriz de rotación y traslación que transforma el ligando original (en este caso `T32_lig.pdb`) en la orientación de *docking* correspondiente. Para efectuar dicha transformación ejecuta el siguiente comando:

```
>pyDock3 T32 rotzdock
```

#### Importante:

Este comando se deberá lanzar desde el mismo directorio donde están los ficheros de receptor y ligando originalmente usados en el docking (en este caso `T32_rec.pdb` y `T32_lig.pdb`). Si estos ficheros han sido modificados, los resultados serán incorrectos.

La ejecución es muy rápida. Se generará el fichero `T32.rot` que será procesado en las siguientes etapas con pyDock.

### 1.4. Puntuación de los modelos de docking mediante la función de energía de pyDock

Ahora se utilizará pyDock para aplicar su función de puntuación a las orientaciones almacenadas en el fichero de rotaciones y traslaciones (`T32.rot`).

**Nota:** Procesar todas las orientaciones de *docking* generadas tomará un tiempo. Para simplificar el cálculo es necesario editar el fichero `T32.rot` para dejar solo **200 orientaciones** (p. ej. eligiendo 200 líneas al azar). Antes de editarlo, se aconseja hacer una copia del fichero `T32.rot` original para tener guardado el conjunto entero de orientaciones de *docking*.

#### 1.4.1. Aplicación de la función de puntuación de pyDock (módulo dockser) a las orientaciones de docking (T32.rot)

Para aplicar la función de puntuación de pyDock (módulo *dockser*) lanza el siguiente comando:

```
<pyDock3 T32 dockser > dockser.log &
```

(este cálculo tardará unos 5 minutos, siempre que se hayan seleccionado previamente 200 orientaciones).

#### Importante:

Es necesario tener en el directorio de trabajo todos los ficheros `*pdb`, `*amber` y `*H` generados con el módulo *setup* de pyDock. Si estos ficheros se eliminan o se modifican, los resultados de este cálculo serán incorrectos. No se puede cambiar el nombre del archivo `T32` porque existen archivos dependientes.

**Nota:** Cuando acabe el cálculo, se habrá generado un fichero de energías (en nuestro caso `T32.ene`) similar al que se muestra a continuación (aunque puede tener diferentes valores), en el que las orientaciones de *docking* están ordenadas por la puntuación energética total de pyDock:

Conf (1)	Ele (2)	Desolv (3)	VDW (4)	Total (5)	RMSD (6)	RANK (7)
162	-19.582	9.471	37.603	-6.351	25.087	1
16	-7.434	-0.979	23.483	-6.064	45.660	2
120	-18.423	6.740	64.540	-5.229	34.547	3
123	-17.490	13.201	-8.382	-5.127	65.864	4
115	-11.731	-5.529	125.148	-4.745	64.336	5

(1) Conf: número de conformación de la orientación de *docking* original de acuerdo al método de *docking* usado, en este caso ZDOCK (indicado en el fichero `T32.rot`, última columna).

(2) Ele: energía electrostática

(3) Desolv: energía de desolvatación

(4) VDW: energía de van der Waals

(5) Total: puntuación energética total (Ele + Desolv + 0.1\*VDW)

(6) RMSD: si se ha indicado una estructura de referencia para el complejo, esta columna mostrará los valores de desviación cuadrática media del ligando para cada orientación de *docking* con respecto a dicha estructura de referencia

(7) RANK: orden de cada orientación de acuerdo a la puntuación energética total de pyDock

**NOTA:** para facilitar el análisis, con el editor vi (o vim) en modo de “no inserción”, se pueden ordenar estas orientaciones por su valor RMSD, si existe, mediante el siguiente comando:

```
:%! sort -n -k6
```

**NOTA:** en modo de “no inserción” si se desea deshacer la ordenación anterior se puede ejecutar el siguiente comando:

```
:u
```

## 1.5. Análisis de los modelos de docking

Mediante el módulo de pyDock *makePDB* se pueden generar los ficheros PDB correspondientes a un rango de modelos de *docking*, definido por los valores de ranking inicial y final en la tabla de energías (columna *RANK*). Por ejemplo, en el caso en estudio, para generar los 10 modelos con mejor puntuación de pyDock, se lanzará el siguiente comando:

```
>pyDock3 T32 makePDB 1 10
```

**Nota:** Este comando generará nuevos ficheros PDB correspondientes a las orientaciones seleccionadas, que se llamarán en este caso T32\_XXX.pdb, donde XXX será el número de conformación en el fichero de energías T32.ene (columna *Conf*). Si se desea generar el fichero PDB para una orientación específica, bastará indicar el mismo valor en los dos argumentos correspondientes al ranking inicial y final (p. ej., para generar la orientación de mejor puntuación de pyDock, se puede lanzar:

```
>pyDock3 T32 makePDB 1 1
```