A	B	C	D
Worflow	**a/d**	**HR**	**EF**
Estructura 1 PRC	3/14	0,21	13
Estructura 2 PRC	1/9	0,11	6,74
Con MS	2/11	0,18	11
Sin MS	3/12	0,25	15,16
Whole PRC combined	3/22	0,14	8,27
PRC + ECR	3/10	0,3	18,2

Fig 1. PRC hecho con lck con 2 estructuras

PRC + ECR
Este es el pipeline con el que mejores resultados obtuve, no recupere active ligands, pero si conseguí bajar la cantidad de decoys de 14 a 10.

Próximos pasos:

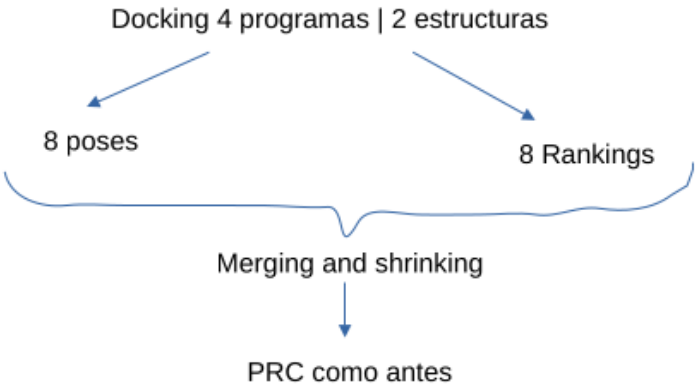
1. En la carpeta de receptores, deben ser preparados para poder correr todos los dockings: dock6, rdock, icm, PL, ad y vina
2. Correr el codigo ya hecho y ver como dan los resultados y como mejora PRC-RED el PRC
    a. El codigo es **"prc_multiple_strucutres.ipynb"**

→ ¿De qué consta el código?:
importante: Para que funcione, necesitamos haber corrido los dockings con todos los programas, y tener guardado en una carpeta todos los outputs en .sdf.
Este archivo .sdf tiene que tener todas las moléculas con diferente nombre (la primera conversión que vemos de Openbabel en el código es para justamente agregarle un diferencial a cada nombre) porque con estos vamos a correr el RMSD entre todas las estructuras.
En el archivo "**calc_rmsd_1.icm**" podemos ver el ejemplo para el receptor de LCK de como queda el archivo para correr con ICM el RMSD.

Ahora sí, siguiendo el código desde "*Armar .csv con rank de cada programa y los rmsd*":
Esta sección del código lo que hace es pasar todos los .sdf con sus columnas de Score y la de RMSD (que la de RMSD está solo en uno de los sdf) junto con el IX a un archivo .csv para poder facilitar más el análisis de estos datos.

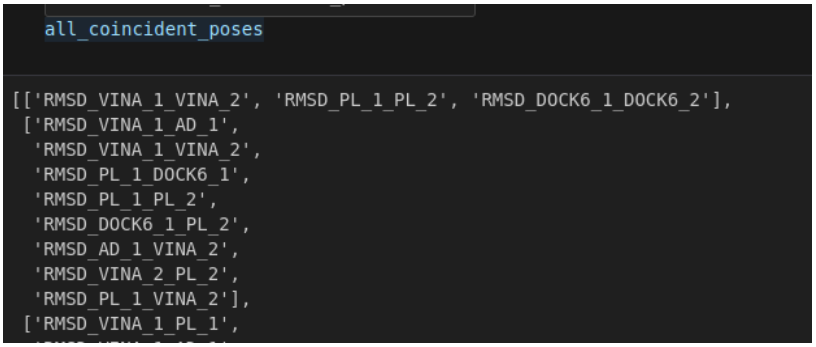
Después de "*Listo el csv, ahora hacer cantidad de poses coincidentes*" hay 3 posibles formas de llevar a cabo PRC-RED que se me ocurrieron (los resultados de cada una los encontramos en la Figura 1.)

1. **Con Merging and shrinking**



a.
b. Con este método, lo que se hizo fue calcular primero el MS, que se fija cual es para una misma molécula, la de menor score/rank y quedarse con esa. llena 3 arrays MS_new, MS_old y MS_ por cada programa.
c. Después mergea todo en un rank de rank_ms por programa
d. Una vez que tenemos esto, pasamos a calcular los exponentes para podes calcular por programa de el conjunto de MS el ECR
e. EL método "`select_rmsds(df)`" elije la columna indicada de RMSD (entre la estructura old o new para cada fila.)
f. Luego sigue la coincidencia de poses: para cada fila (o sea, cada molecula) almacenamos en un array las non_null_columns, es decir nos quedamos con el nombre de la columna cuyo rank es el que hay que usar, ya sea new or old (sólo va a ser elegido uno por programa). Entonces se ve tal que asi:
    i.
```
non_null_columns

[['PL_old', 'dock6_new', 'vina_old', 'AD_old'],
 ['PL_new', 'dock6_old', 'vina_new', 'AD_old'],
 ['PL_old', 'dock6_old', 'vina_new', 'AD_old'],
 ['PL_old', 'dock6_old', 'vina_old', 'AD_old'],
 ['PL_old', 'dock6_old', 'vina_old', 'AD_old'],
 ['PL_new', 'dock6_new', 'vina_old', 'AD_new'],
 ['PL_old', 'dock6_old', 'vina_old', 'AD_old'],
```
    ii. Siempre su tamaño va a ser la cantidad de programas elegidos, solo cambia si queda el new o el old.
g. Después sigue entre ellos calcular las posibilidades de combinación entre 2 programas para las combinaciones elegidas y chequear los RMSD de estos. se almacena en el array "combination_existing_in_df_2p"
h. Luego, en caso de existir esa combinación, se agrega a otro array cada valor de RMS, si no existe se pone 0 de valor.
i. Con estos 2 arrays, se correlacionan los que no son 0, con los nombres de las columnas de RMSD y se pasa a buscar cantidad de poses q coinciden
j. Con eso se calculan result_of_coincidents (el nombre de la columna que coincide con < 2) y numb_of_coincidences (cuantas hay con valor < a 2)
k. Se agregan al .csv.
l. Después sigue "Ahora hay que ver que mols pasan el PRC", que en esencia hace lo mismo que PRC, por lo que no se va a explicar a fondo acá.

2. **SIN merging and shrinking**
    a. Este pipeline es mas sencillo, ya que se fija todo con todo, sin necesidad de hacer MS antes.
    b. Como se calcularon todos los RMSD de todo contra todo, lo primero es seleccionar los que son menor a 2 y almacenar el nombre de la columna.
        i. 
```
all_coincident_poses
```
```
[['RMSD_VINA_1_VINA_2', 'RMSD_PL_1_PL_2', 'RMSD_DOCK6_1_DOCK6_2'],
 ['RMSD_VINA_1_AD_1',
  'RMSD_VINA_1_VINA_2',
  'RMSD_PL_1_DOCK6_1',
  'RMSD_PL_1_PL_2',
  'RMSD_DOCK6_1_PL_2',
  'RMSD_AD_1_VINA_2',
  'RMSD_VINA_2_PL_2',
  'RMSD_PL_1_VINA_2'],
 ['RMSD_VINA_1_PL_1',
```
        ii. Cada sub array tiene la información para cada molecula de los programas que coinciden
        iii. Aca se cambió old y new por 1 y 2.
    c. Después descomponemos este array en los programas que están involucrados, sacando repeticiones.
    d. Almacenamos en el csv las poses coincidentes por programa y la cantidad. (como no hubo un filtrado antes, hay hasta 8 poses coincidentes ahora)
    e. Ahora, lo que busco es si hay varias, quedarme con la de mejor ECR, por lo que hay que calcular los ecr
    f. Después: Elijo dentro de todos las poses coincidentes en una misma molécula, la de mejor ecr de esa combinación
    g. Post este último filtro, calculamos PRC normalmente.
3. **Combinar PRC al final**
    a. En esta opción, lo que se hace es calcular al principio PRC normal:
        i. por estructura se hace consenso de pose
        ii. entre las N estructuras, después vemos cual es la mejor:
            1. ej: para una molécula: E1 tiene 4p coincidentes y E2 tiene 3p, me quedo con la de mejor ECR
        iii. Después de esto, voy a tener más de una columna de PRC final, porque tengo una por estructura. estas 2 columnas las uno siguiendo el mismo concepto de ECR (es decir calculo ECR y filtro tal cual PRC de siempre como si fuesen 2 programas)
        iv. Esta opción es la que mejor performance tuvo