**Supporting Information**

**Comparing a query compound with drug target classes using 3D- chemical similarity**

Sanghyeok Lee, 1 Sangjin Ahn, 2 Mi-hyun Kim1\*

*1Gachon Institute of Pharmaceutical Science and Department of Pharmacy, College of Pharmacy, Gachon University, Yeonsu-gu, Incheon, 21936, Republic of Korea, 2Department of Financial Engineering, College of Business, Ajou University, Suwon, 16499, Republic of Korea*

\*Author for correspondence

E-mail: [kmh0515@gachon.ac.kr](mailto:kmh0515@gachon.ac.kr)

**Table of Contents**

1. Mathematical description
2. Summary of datasets (S.Table1)
3. The common ligands between target classes (S.Table2)
4. 2D Structures of queries (S.Table3)
5. Gaussian mixture models for additional 4 target classes (S.Figures 1 to 4)
6. Frequency distributions of estimates (and) for target classes (S.Figures 5 to 7)
7. Python script for 3D similarity matrix

**The derivation of Kullback–Leibler (K–L) divergence between the two Gaussian distributions.**

The Kullback–Leibler divergence, is defined as either

, (S1.1)

We assume the probability distributions P(x) and Q(x) replace the Gaussian distributions and , where

and , (S1.2)

and the probability density functions, and , are

. (S1.3)

Using (S1.1) and (S1.3), the Kullback–Leibler divergence between the two Gaussian distributions and in (S1.2) are as follows:

=

=

. (S1.4)

By the following relationships,

(S1.5)

we obtain

. (S1.5)

**The mathematical elucidation of expectation-maximization (EM) algorithm to achieve the hyperparameters of Gaussian mixture model (GMM).**

The mixture models, is defined as

(S1.6)

with representing a probabilistic density generated from the unknown compositional data, representing a well-known probability density, and **x** representing a random vector, the functional operator.

Concretely, the EM algorithm for GMM is summarized as follows:

- Start with an initial guess for the GMM parameters.

- E-step: Calculate the conditional expectation of the log likelihood, , for the incomplete data with respect to , , and

, (S1.7)

where and are the vectors of hyperparameters, such that

(S1.8)

and represents a random variable.

* M-step: Determine the parameters, such that

. (S1.9)

In order to find the set of hyperparameters, , the following recursive relationship of elements in is obtained:

, and (S1.10)

,

where

. (S1.11)

- Continue to perform the E- step and the M-step until a positive, infinitesimal number, exists, such that

, (S1.12)

for where N is an actual large number.

**Example calculation of K–L divergence from CHEMBL539392 as a query ligand.**

Using the above equation (2.6) in the main text for Kullback–Leibler divergence between normal distributions,

(3.14)

where

, (3.15)

When *n* = 1, i.e., the class was ESR, the parameters were as follows:

*,*  (S2.1)

and the following K–L divergence was obtained:

. (S2.2)

When *n* = 2, i.e., the class was VDR, the parameters were as follows:

, (S2.3)

and the following K–L divergence was obtained:

(S2.4)

When *n* = 3, i.e., the class was Cyclooxygenase-2, the parameters were as follows:

(S2.5)

and the following K–L divergence was obtained:

(S2.6)

When n = 4, i.e., the class was CTSD, the parameters were as follows:

, (S2.7)

and the following K–L divergence was obtained:

(S2.8)

**S.Table 1.** Summary of datasets

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **Original Conformers**  **(Compounds)** | **Sampled Conformers**  **(Compounds)** | **Resource** |
| Estrogen receptor alpha (ESR) | 278,809  (3460) | 13,957  (3,460) | ChEMBL23 |
| Vitamin D receptor (VDR) | 1,395,788  (11,715) | 13,957  (11,715) | ChEMBL23 |
| Cyclooxygenase-2 (COX2) | 281,176  (3,736) | 13,957  (3,736) | ChEMBL23 |
| Cathepsin D (CTSD) | 154,129  (856) | 13,957  (856) | ChEMBL23 |
| HIV-1 protease (HIV1) | 2,366,517  (603) | 13,957  (603) | ChEMBL25 |
| Heat shock protein 90 (HSP90) | 72,578  (459) | 13,957  (459) | ChEMBL25 |
| Transient receptor potential cation channel subfamily V4 (TRPV4) | 74,591  (468) | 13,957  (468) | ChEMBL25 |
| DNA topoisomerase I (TOP1) | 47,957  (282) | 13,957  (282) | ChEMBL25 |

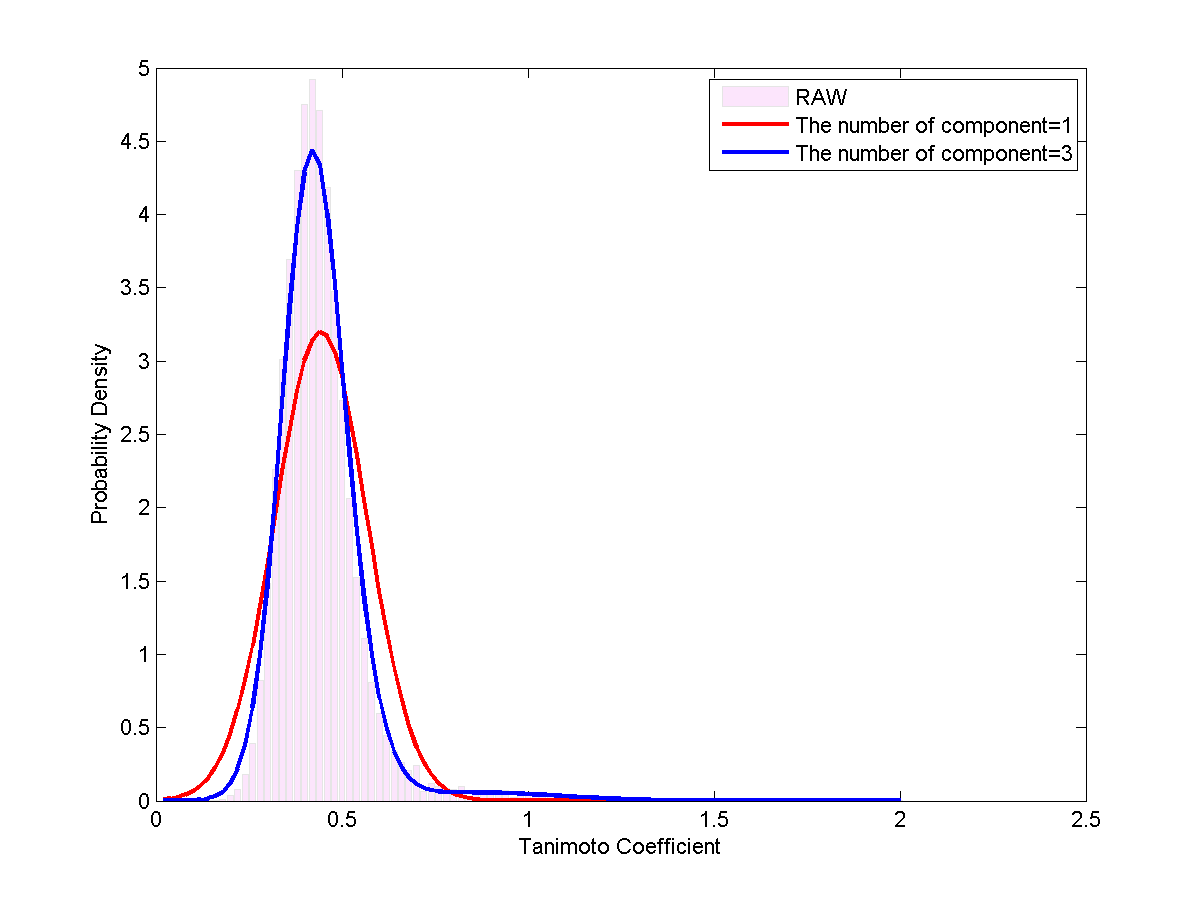
**S.Table2.** The overlapped ligands between target class

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ESR | VDR | COX2 | CTSD | HIV1 | HSP90 | TRPV4 | TOP1 |
| ESR | self | 281 | 634 | 2 | 5 | 0 | 0 | 7 |
| VDR | 281 | self | 130 | 1 | 7 | 4 | 0 | 4 |
| COX2 | 634 | 130 | self | 2 | 7 | 2 | 1 | 8 |
| CTSD | 2 | 1 | 2 | self | 4 | 0 | 0 | 0 |
| HIV1 | 5 | 7 | 7 | 4 | self | 0 | 0 | 0 |
| HSP90 | 0 | 4 | 2 | 0 | 0 | self | 0 | 1 |
| TRPV4 | 0 | 0 | 1 | 0 | 0 | 0 | self | 0 |
| TOP1 | 7 | 4 | 8 | 0 | 0 | 1 | 0 | self |

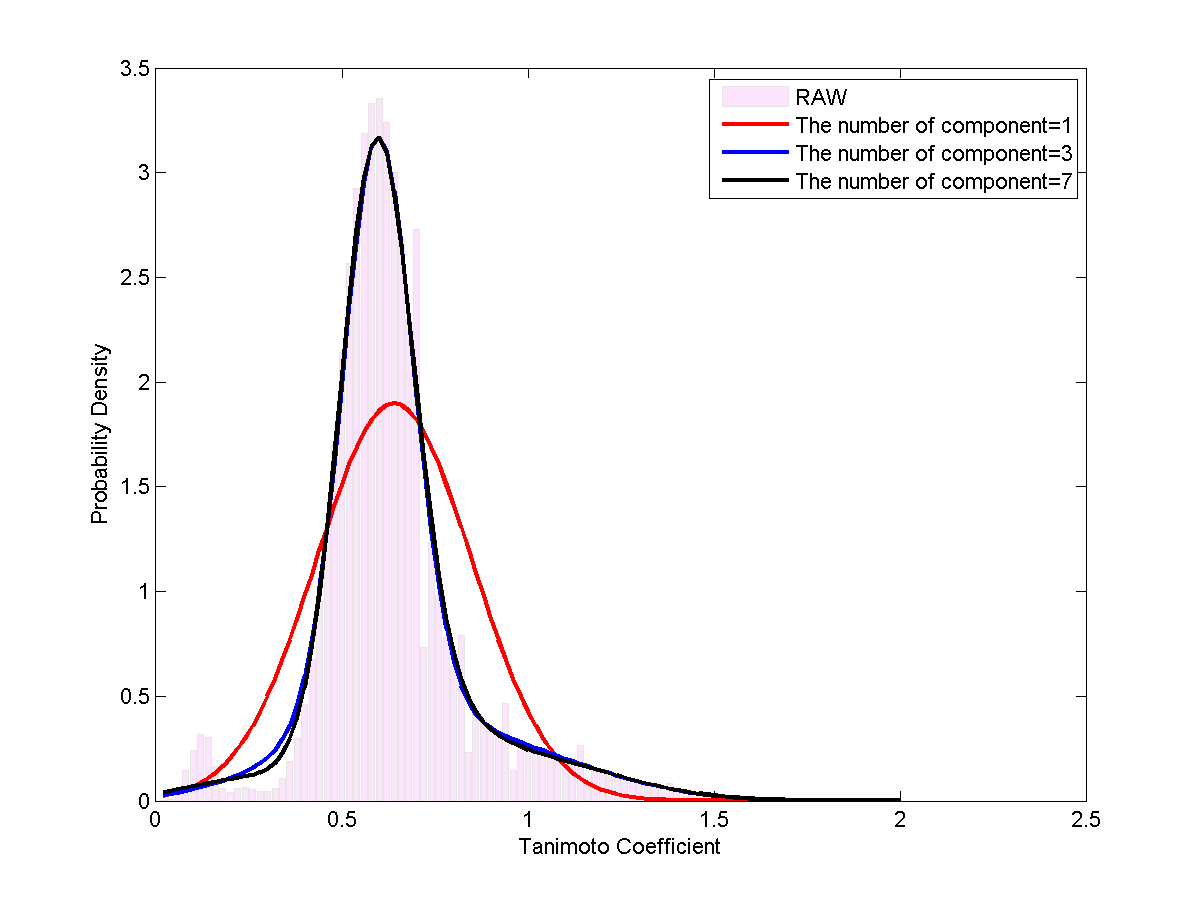
**Table3.** Structure information of randomly chosen queries for K-L divergence calculation from 4 targets (ESR, VDR, COX2, and CTSD).

|  |  |  |  |
| --- | --- | --- | --- |
| ID | CHEMBL539392 | CHEMBL193280 | CHEMBL443605 |
| ESR  Ligand |  |  |  |
| ID | CHEMBL7162 | CHEMBL1322390 | CHEMBL1452735 |
| VDR  Ligand |  |  |  |
| ID | CHEMBL1163237 | CHEMBL127560 | CHEMBL271614 |
| COX2  Ligand |  |  |  |
| ID | CHEMBL263810 | CHEMBL252655 | CHEMBL436438 |
| CTSD  Ligand |  |  |  |

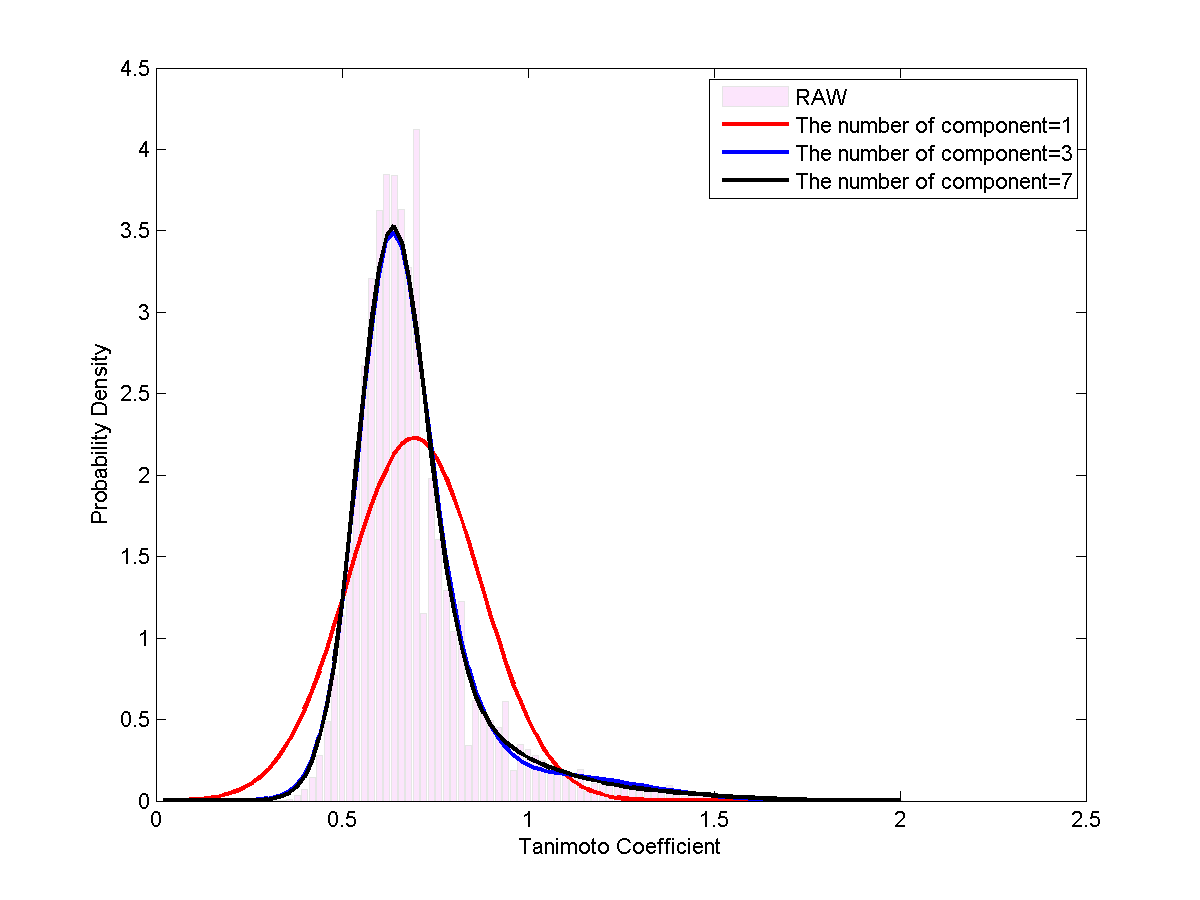
**S.Figure1.** *Q*-distribution of HIV1 based on Gaussian mixture model



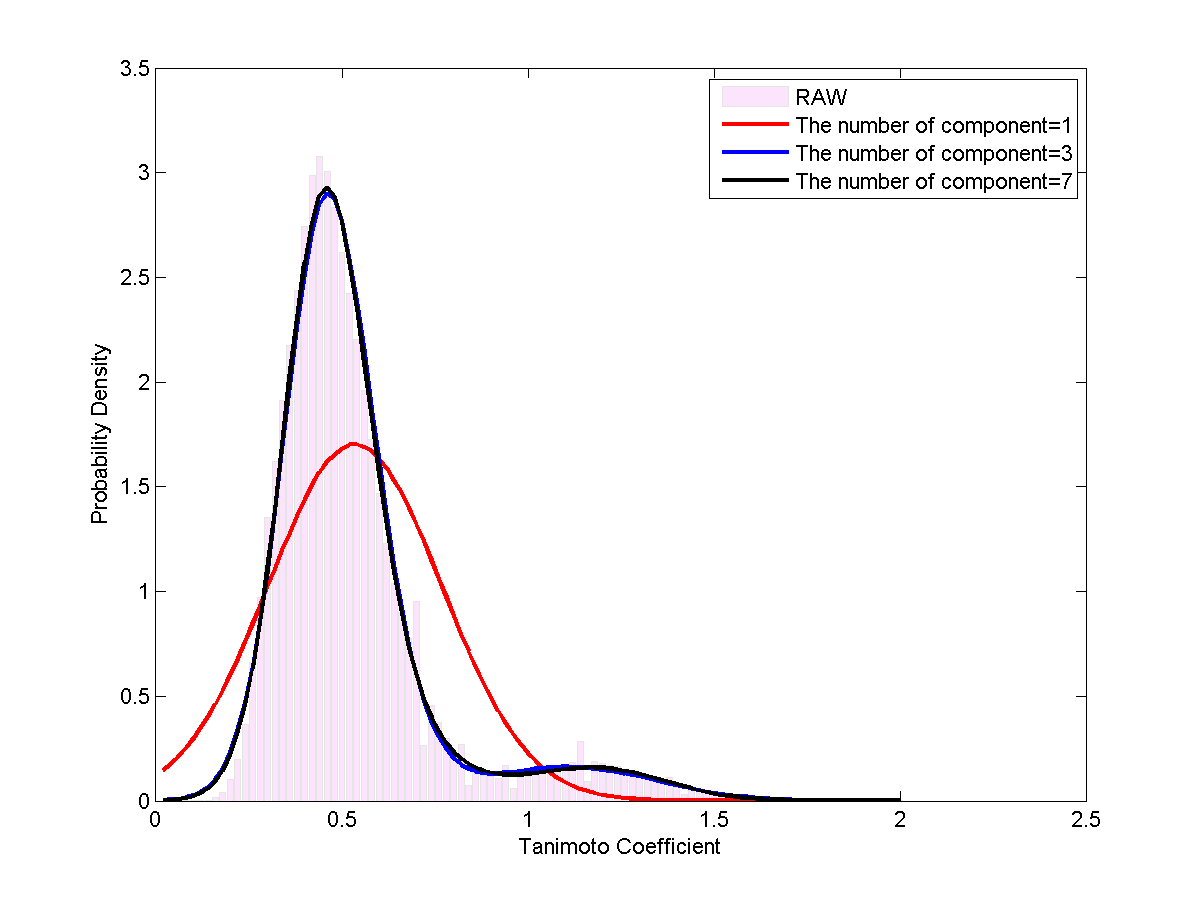
**S.Figure2.** *Q*-distribution of HSP90 based on Gaussian mixture model



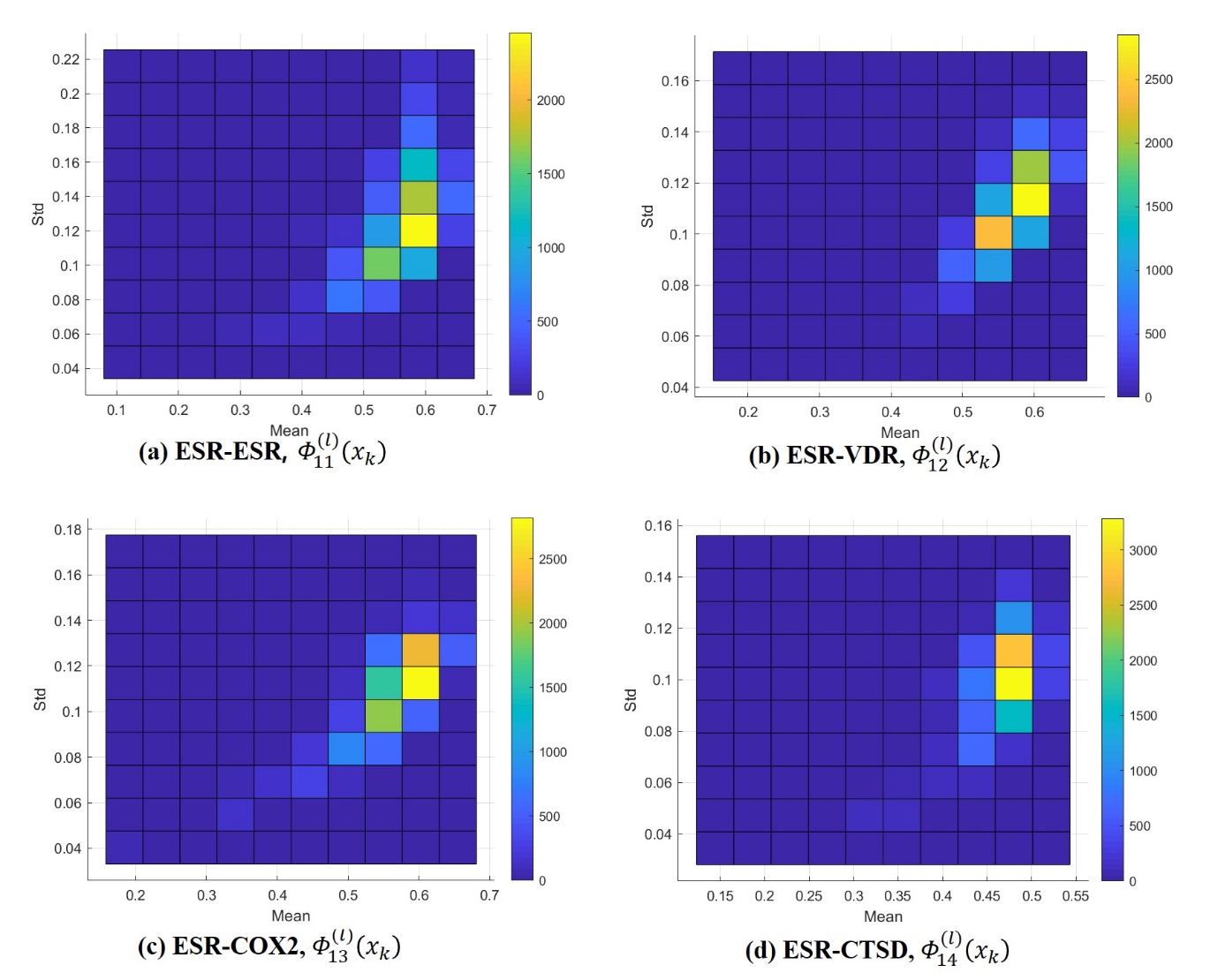
**S.Figure3.** *Q*-distribution of TRPV4 based on Gaussian mixture model



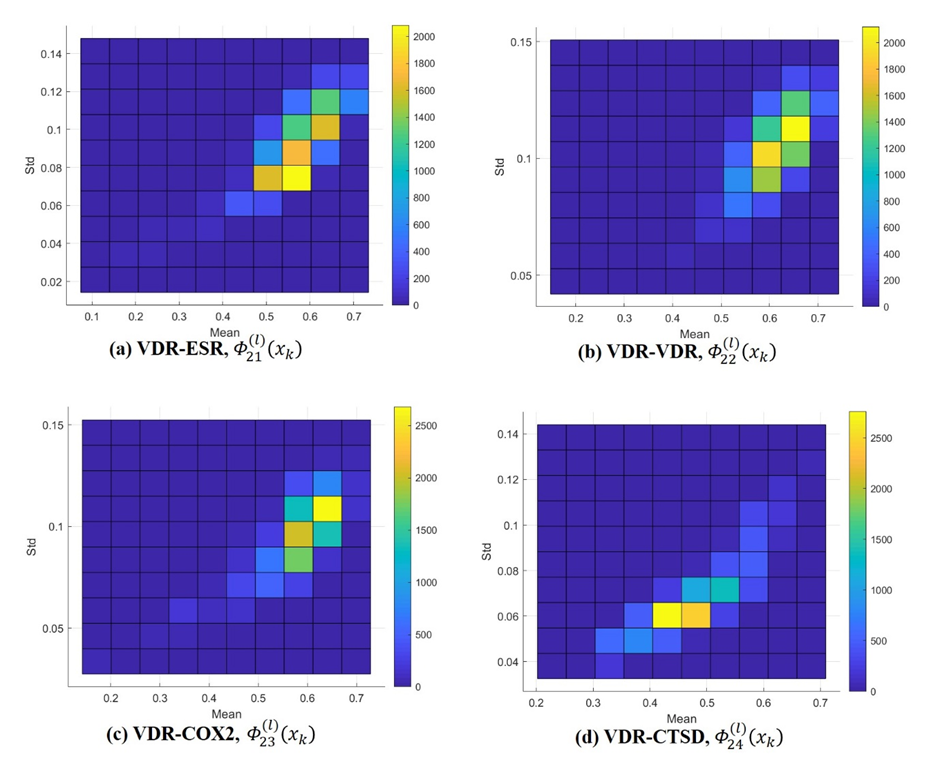
**S.Figure4.** *Q*-distribution of TOP1 based on Gaussian mixture model



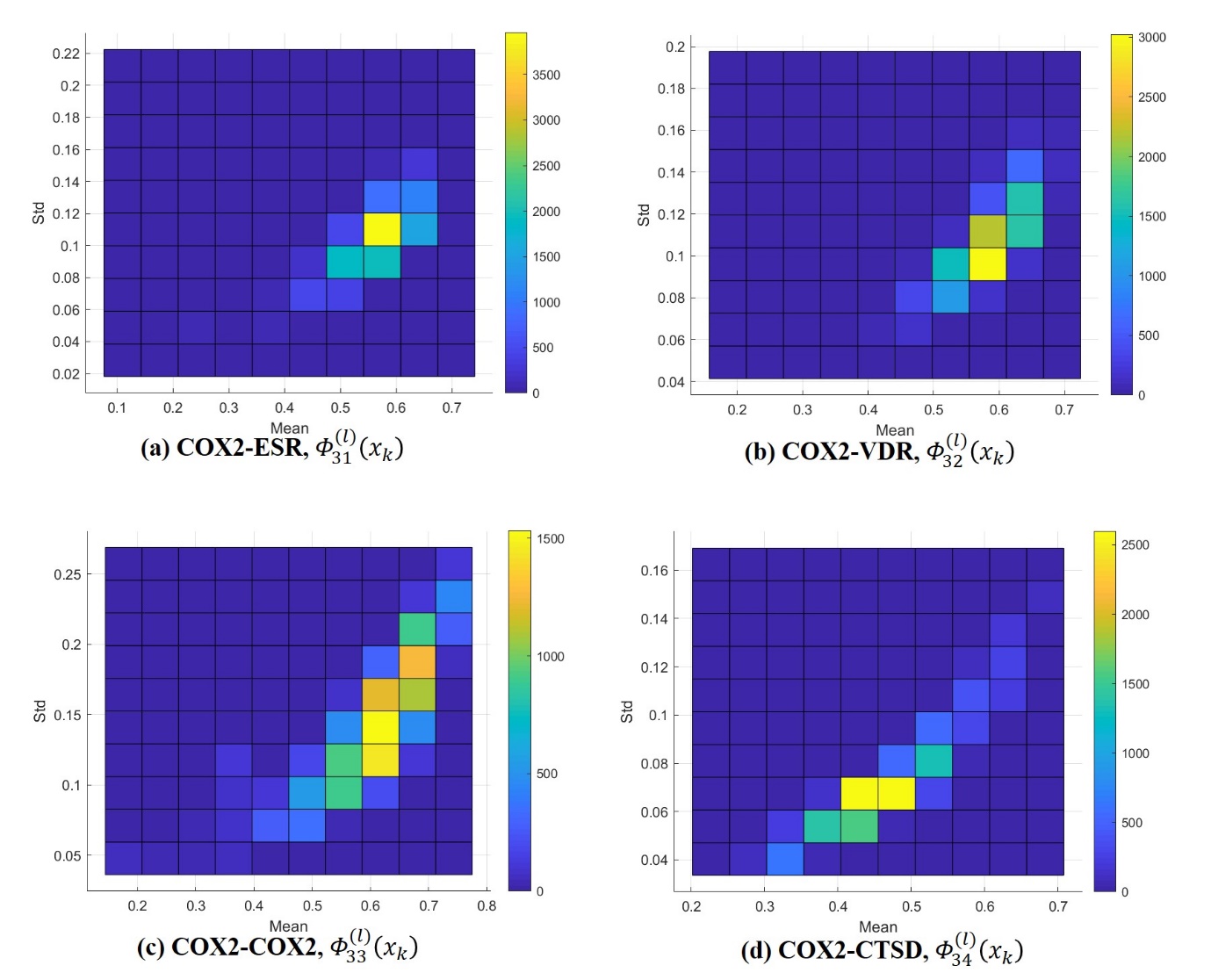
**S. Figure 5.** Frequency distributions of estimates (and). Query (*l*) ESR (class = 1). (a) ESR-ESR, (b) ESR-VDR, (c) ESR-COX2, and (d) ESR-CTSD.\* \*The color bars (right side of the distribution) indicate frequency (eg. Yellow in 3(d) means that for over 3000 queries, the mean of the ML estimates varied from 0.45 to 0.5 and their standard deviation varied from 0.09 to 0.11.).



**S. Figure 6.** Frequency distributions of estimates (and). Query (*l*) VDR (class = 2). (a) VDR-ESR, (b) VDR-VDR, (c) VDR-COX2, and (d) VDR-CTSD.\* \*The color bars (right side of the distribution) indicate frequency (eg. Yellow in 4(d) means that for over 2500 queries, the mean of the ML estimates varied from 0.4 to 0.45 and their standard deviation varied from 0.052 to 0.062.).



**S. Figure 7.** Frequency distributions of estimates (and). Query (*l*) COX2 (class = 3). (a) COX2-ESR, (b) COX2-VDR, (c) COX2-COX2, and (d) COX2-CTSD.\* \*The color bars (right side of the distribution) indicate frequency (eg. Yellow in 5(a) means that for over 3500 queries, the mean of the ML estimates varied from 0.5 to 0.6 and their standard deviation varied from 0.1 to 0.12 in the standard.).



Python script (.py) for 3D similarity matrix

import sys

from openeye import oechem

from openeye import oeshape

from threading import Thread

def BConf(Aconf, prep, molB, csvresult):

csvresult.write("%s\_%d" % (Aconf.GetTitle(), Aconf.GetIdx()))

refmol = oechem.OEGraphMol(Aconf)

options = oeshape.OEOverlayOptions()

options.SetOverlapFunc(oeshape.OEOverlapFunc())

overlay = oeshape.OEOverlay(options)

overlay.SetupRef(refmol)

bfs = oechem.oemolithread(molB)

fitmol = oechem.OEMol()

while oechem.OEReadMolecule(bfs, fitmol):

prep.Prep(fitmol)

scoreiter = oeshape.OEBestOverlayScoreIter()

oeshape.OESortOverlayScores(scoreiter, overlay.Overlay(fitmol), oeshape.OEHighestTanimotoCombo())

for score in scoreiter:

csvresult.write(",%.2f" % score.GetTanimotoCombo())

csvresult.write("\n")

def genHeader(filename, csvresult):

csvresult.write("name")

ifs = oechem.oemolithread(sys.argv[2])

mol = oechem.OEMol()

while oechem.OEReadMolecule(ifs, mol):

for conf in mol.GetConfs():

csvresult.write(",%s\_%d" % (conf.GetTitle(), conf.GetIdx()))

csvresult.write("\n")

def main(argv=[\_\_name\_\_]):

if len(argv) != 4:

oechem.OEThrow.Usage("%s <Afile(Query\_ref)> <Bfile\_Fitmol> <csvresult>" % argv[0])

csvresult = open(argv[3], "w")

genHeader(sys.argv[2], csvresult)

prepA = oeshape.OEOverlapPrep()

afsA = oechem.oemolithread(sys.argv[1])

MA = afsA.GetOEMols()

for molA in MA:

prepA.Prep(molA)

confA= molA.GetConfs()

print(molA.GetTitle())

for conf in confA:

BConf(conf, prepA, sys.argv[2], csvresult)

if \_\_name\_\_ == "\_\_main\_\_":

import sys

sys.exit(main(sys.argv))