Supporting Information for：

Dissecting Olfactory Receptor Binding Pocket Gating Induced by Structurally Similar Odorants Furaneol and Sotolone through Molecular Dynamics Simulations

Penghui Li[1†], Haiyang Wang[2,3†], Yibo Wang[1], Zengyang He[2,3], Shaolin Ge[2,3], Peng Zou[3], Qiong Liu[4,5,6], Yan Feng[7], Wenbin Wang[3]*, Xiubo Du[4]*, Xiaohui Wang[1,8]*

[1]Interdisciplinary Laboratory for Frontier Chemistry, Changchun Institute of Applied Chemistry, Chinese Academy of Sciences, Changchun, Jilin 130022, China

[2]Anhui Provincial Key Laboratory of Aerosol Analysis, Regulation and Biological Effect, Hefei, 230088, China

[3]Technology Center, China Tobacco Anhui Industrial Co., Ltd., Hefei, 230088, China

[4]Shenzhen Key Laboratory of Marine Biotechnology and Ecology, College of Life Sciences & Oceanography, Shenzhen University, Shenzhen, 518055, China

[5]Key Laboratory of Optoelectronic Devices and System of Ministry of Education and Guangdong Province, College Physics and Optoelectronic Engineering, Shenzhen University, Shenzhen, 518060, China

[6]Shenzhen-Hong Kong Institute of Brain Science, Shenzhen Fundamental Research Institutions, Shenzhen, 518055, China

[7]School of Chemistry and Chemical Engineering & Institutes of Physical Science and Information Technology, Anhui Province Key Laboratory of Chemistry for Inorganic/ Organic Hybrid Functionalized Materials & Key Laboratory of Structure and Functional Regulation of Hybrid Materials of Ministry of Education, Anhui University, Hefei, 230601, China

[8]School of Applied Chemistry and Engineering, University of Science and Technology of China, Hefei, 230026, China

* Corresponding authors

Emial: 45643864@qq.com; duxiubo@szu.edu.cn; xiaohui.wang@ciac.ac.cn

† These authors contributed equally to this work.

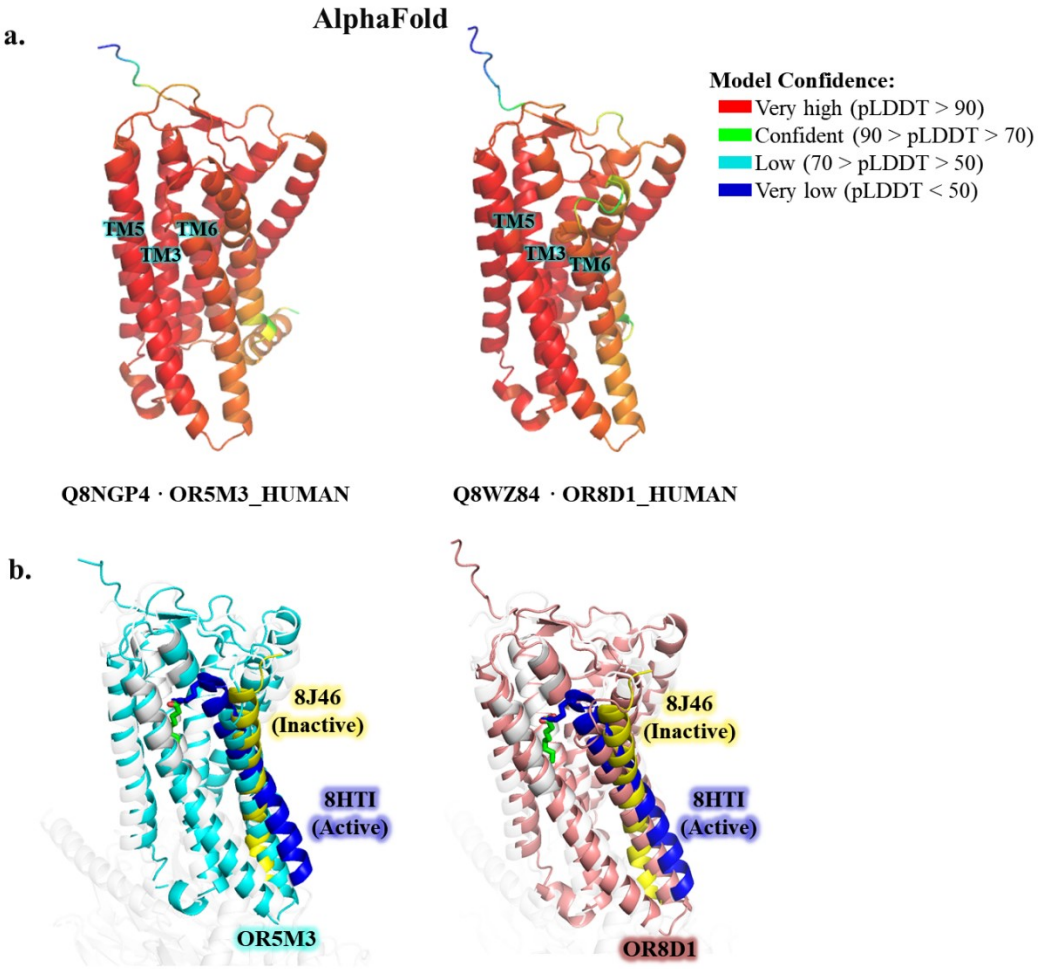

Figure S1. (a) Confidence scores (pLDDT) of the initial coordinates of OR5M3 and OR8D1. (b) Comparison with the corresponding crystal structures.

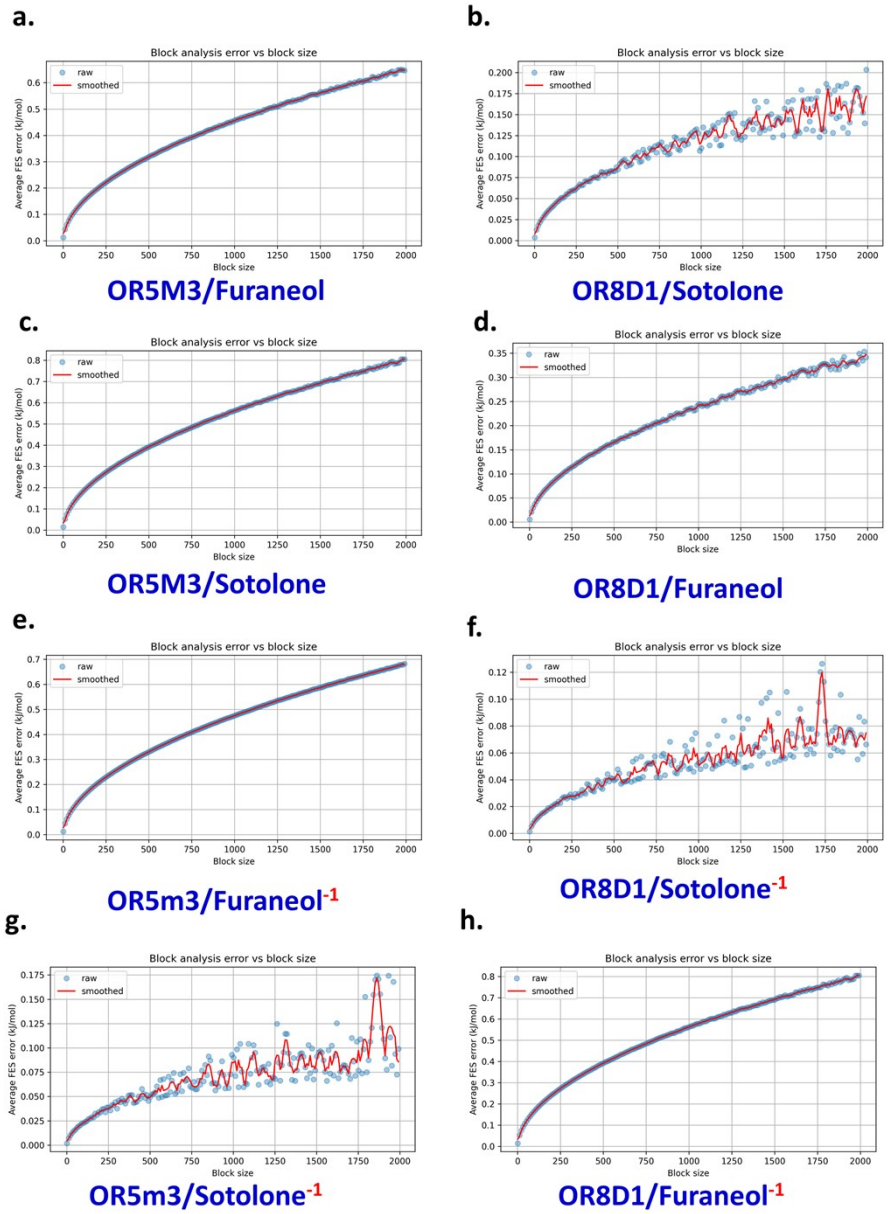

Figure S2. Block analysis was performed on the 500 ns metadynamics simulation to assess the convergence of the simulation. A simple Python script was used to generate each figure directly from the HILLS file produced during the metadynamics simulation.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt
from joblib import Parallel, delayed
import pandas as pd
```

```
# -------------------------
#  参数设置
# -------------------------
hills_file = "HILLS"
cv_min, cv_max = -np.pi, np.pi
n_bins = 51
kT = 2.494339
block_stride = 10            # block size  步长
max_block_size = 2000     #  最大  block size
n_jobs = 8                      #  并行核数
stride_data = 1               # HILLS  取样间隔，1=全用，10=每 10 行取一行

# -------------------------
#  读取  HILLS  文件
# -------------------------
data = []
with open(hills_file, "r") as f:
    for line in f:
        if line.startswith("#!"):
            continue
        parts = line.strip().split()
        if len(parts) < 5:
            continue
        time, cv, sigma, height, biasf = map(float, parts[:5])
        data.append([time, cv, sigma, height, biasf])

data = np.array(data)
cv = data[:,1][::stride_data]
height = data[:,3][::stride_data]

# -------------------------
#  累积  bias -> unbias weights
# -------------------------
V = np.cumsum(height)
Vmax = V.max()
weights = np.exp((V - Vmax)/kT)[::stride_data]

# -------------------------
# CV  网格
# -------------------------
grid = np.linspace(cv_min, cv_max, n_bins)
bin_edges = np.linspace(cv_min, cv_max, n_bins+1)
bin_idx = np.digitize(cv, bin_edges) - 1
```

```
bin_idx = np.clip(bin_idx, 0, n_bins-1)

# -------------------------
# 计算一个 block size 的 FES
# -------------------------
def process_block_size(block_size):
    n_blocks = len(cv) // block_size
    if n_blocks == 0:
        return block_size, np.nan   # 避免 block 太大

    idx_blocks = bin_idx[:n_blocks*block_size].reshape(n_blocks, block_size)
    w_blocks = weights[:n_blocks*block_size].reshape(n_blocks, block_size)

    # 向量化 histogram
    fes_blocks = np.zeros((n_blocks, n_bins))
    for b in range(n_blocks):
        hist_w = np.zeros(n_bins)
        np.add.at(hist_w, idx_blocks[b], w_blocks[b])
        fes_blocks[b] = -kT * np.log(np.maximum(hist_w, 1e-20))
        fes_blocks[b] -= fes_blocks[b].min()

    mean_fes = fes_blocks.mean(axis=0)
    error_fes = fes_blocks.std(axis=0, ddof=1)/np.sqrt(n_blocks)

    np.savetxt(f"fes.{block_size}.dat", np.column_stack([grid, mean_fes, error_fes]))
    avg_err = error_fes.mean()
    return block_size, avg_err

# -------------------------
# 并行计算所有 block size
# -------------------------
block_sizes = list(range(1, max_block_size+1, block_stride))
results = Parallel(n_jobs=n_jobs)(delayed(process_block_size)(bs) for bs in block_sizes)
results = np.array(results)
np.savetxt("err.blocks", results)

# -------------------------
# 绘图：平均误差 vs block size
# -------------------------
# 使用 Pandas 滑动平均，避免最后点拉长
window = 3
smooth_err = pd.Series(results[:,1]).rolling(window, center=True, min_periods=1).mean()
```

```
plt.figure(figsize=(7,4))
plt.plot(results[:,0], results[:,1], 'o', alpha=0.4, label='raw')
plt.plot(results[:,0], smooth_err, '-', color='red', label='smoothed')

plt.xlabel("Block size")
plt.ylabel("Average FES error (kJ/mol)")
plt.title("Block analysis error vs block size")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.savefig("block_error_vs_size.png", dpi=300)
# plt.show()
```

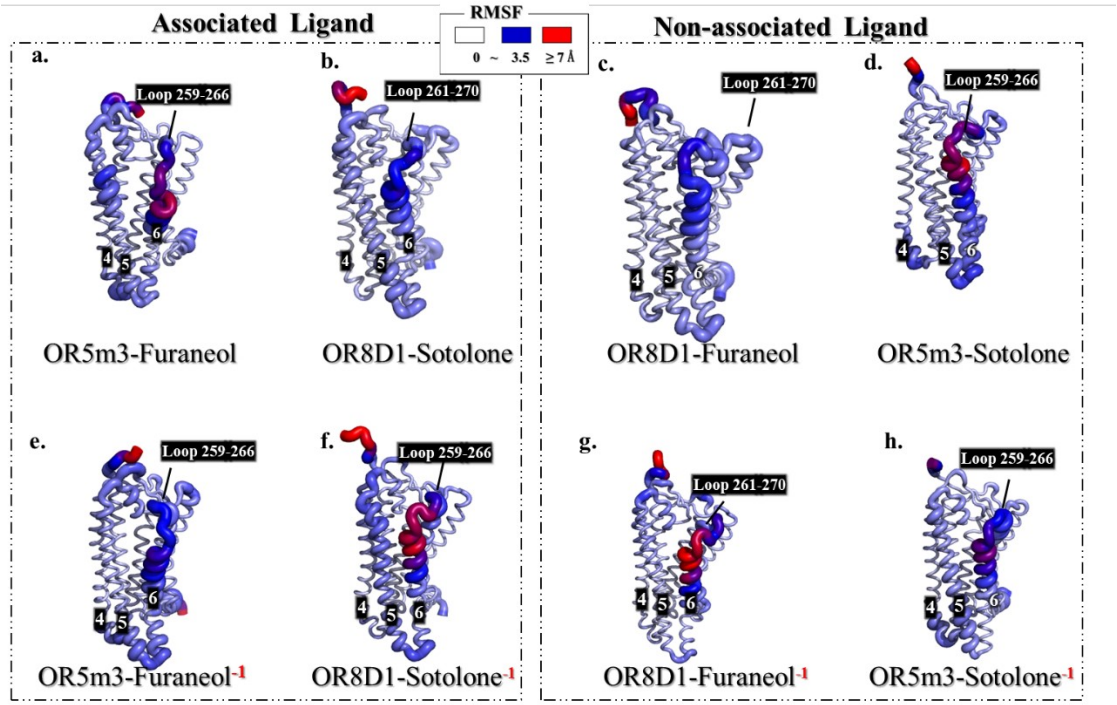

Figure S3. Receptor flexibility was evaluated using RMSF calculated from 500 ns of metadynamics simulation after reweighting. Cartoon-putty representations are shown, where both thickness and color scale with the corresponding RMSF, with thicker regions indicating higher flexibility. Frame weights were applied using PLUMED, and RMSF values, including those for the PDB structure, were calculated using a Python script:

```
import numpy as np
import MDAnalysis as mda

# ================== 用户参数 ==================
traj_file     = "prod-meta.pdb"            # 多帧 PDB 轨迹
weight_file = "COLVAR_REWEIGHT"            # PLUMED 重加权文件
output_pdb    = "weighted_rmsf.pdb"
selection     = "protein and name CA"      # 可改为 "all" 或 "backbone"
align_to_first = False                     # 如果已经对齐，可设为 False
# ============================================

# ---- 读取权重 ----
data = np.loadtxt(weight_file)
# 假设第二列是 rbias
logw = data[:,1]
logw -= np.max(logw)          # 防止指数溢出
```

```
w = np.exp(logw)
w /= np.sum(w)                    # 归一化权重

print("Max/min weight ratio:", np.max(w)/np.min(w))

# ---- 读取轨迹 ----
u = mda.Universe(traj_file, traj_file)
atoms = u.select_atoms(selection)
n_atoms = len(atoms)
n_frames = len(u.trajectory)
print(f"Number of frames: {n_frames}, number of selected atoms: {n_atoms}")

# ---- 提取轨迹坐标 ----
positions = np.zeros((n_frames, n_atoms, 3))

# 如果已经对齐，则直接读取坐标
for i, ts in enumerate(u.trajectory):
    positions[i] = atoms.positions   # 固定原子 selection，保证 shape 一致

# ---- 如果需要，可以对齐到第一帧 ----
if align_to_first:
    ref = positions[0]
    for i in range(n_frames):
        R, t = mda.analysis.align.rotation_matrix(positions[i], ref)
        positions[i] = np.dot(positions[i] - positions[i].mean(axis=0), R.T) + ref.mean(axis=0)

# ---- 计算加权平均坐标 ----
mean_pos = np.average(positions, axis=0, weights=w)

# ---- 计算加权 RMSF ----
diff = positions - mean_pos
sq_diff = np.sum(diff**2, axis=2)
rmsf = np.sqrt(np.average(sq_diff, axis=0, weights=w))
```

```
# ---- 写入 PDB (B-factor 列) ----
with mda.Writer(output_pdb, n_atoms) as W:
    atoms.tempfactors = rmsf
    W.write(atoms)

print(f"Weighted RMSF PDB written to {output_pdb}")
```

PYMOL script:

```
bg_color white
as cartoon
cartoon putty
#alter * and (b > 150), b=150
set cartoon_putty_radius, 1.0
set cartoon_putty_range, 1.0
set cartoon_putty_scale_max, 2
set cartoon_putty_scale_min, 0.6
set cartoon_putty_scale_power, 2
spectrum b, white_blue_red, minimum=0, maximum=7
```

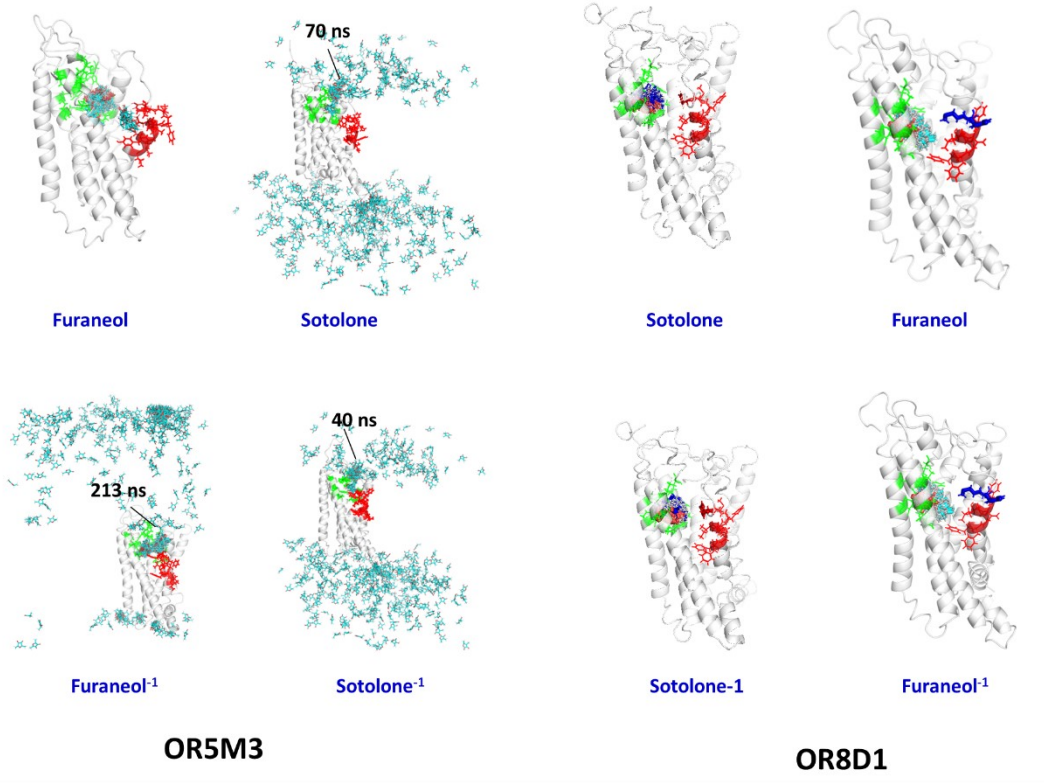

Figure S4. Dynamic behavior of the odorant during the 500 ns metadynamics simulation, illustrated by the superposition of 500 snapshots. The extracellular region of TM6 is shown in red, residues surrounding the ligand-binding pocket are shown in green, and the odorants are represented as sticks.

Table S1. Binding free energies of the odorants furaneol and sotolone toward the olfactory receptors OR5M3 and OR8D1, calculated using the MM/PBSA method. Energetic components include van der Waals, electrostatic, and solvation contributions.[a] All energies are reported in kcal/mol.

	OR5M3		**OR8D1**	
	Furaneol	**Sotolone**	**Furaneol**	**Sotolone**
ΔE_{vdW}	-18.9(2.3)	-16.9(1.9)	-21.2(1.6)	-20.9 (± 1.6)
ΔE_{ele}	-0.8(0.5)	-0.3(0.3)	-0.2(0.3)	-1.2 (± 0.1)
$\Delta G_{sol,GB}$	1.0 (0.3)	0.6(0.2)	0.6(0.1)	0.9 (± 0.1)
$\Delta G_{sol,np}$	-2.1(0.1)	-2.1(0.1)	-2.1(0.1)	-2.0 (± 0.1)
ΔE_{gas}	-19.7(2.2)	-17.3(2)	-21.5(1.6)	-22.1 (± 1.5)
ΔE_{vdW}	-1.1(0.3)	-1.5(0.2)	-1.5(0.2)	-1.1 (± 0.1)
ΔH_{total}	-20.8(2.2)	-18.7(2)	-22.9(1.6)	-23.2 (± 1.5)
	Furaneol^{-1}	**Sotolone^{-1}**	**Furaneol^{-1}**	**Sotolone^{-1}**
ΔE_{vdW}	-16.2(1.7)	-11.7(3)	-15.6(2.6)	-17.6(1.8)
ΔE_{ele}	-6.2(0.4)	-6.0(1.2)	-1.1(1.4)	-1.9(0.9)
$\Delta G_{sol,GB}$	5.1(0.2)	4.9(0.8)	2.3(0.8)	2.6(0.4)
$\Delta G_{sol,np}$	-2.1(0.1)	-1.9(0.2)	-2.0 (0.1)	-2.0 (0.1)
ΔE_{gas}	-22.4(1.7)	-17.7(3.1)	-16.7(3.4)	-19.5(1.9)
ΔE_{vdW}	3.0 (0.2)	3.0 (0.7)	0.2(0.8)	0.6(0.4)
ΔH_{total}	-19.5(1.7)	-14.6(3)	-16.5(2.8)	-19.0(1.8)
$-\Delta TS$	-15.0 (2.8)	-12.9(6.9)	-14.9(3.9)	-14.9(3.9)
ΔG_{total}	-34.5	-27.5	-31.4	-33.9

[a]An implicit membrane model was applied for enthalpy calculations, following the protocol described on pages 95 and 891 of the Amber 2024 Reference Manual. Trajectory data from the 200 ns of each simulation were used, with 4000 frames sampled for enthalpy and 100 frames for entropy calculations.

Script for Figure 6. The hydrogen-bond networks of the eight complex systems are represented using a mosaic-style matrix heat map.

1. cpptraj -i cpptra.in

```
parm ../com-sol.prmtop

trajin ../prod?.mdcrd 1 last 1

trajin ../prod1?.mdcrd 1 last 1

trajin ../prod2?.mdcrd 1 last 1

trajin ../prod3?.mdcrd 1 last 1

trajin ../prod4?.mdcrd 1 last 1

trajin ../prod50.mdcrd 1 last 1

strip :WAT,PA,PC,OL,Na+,Cl-

hbond :* out nhb.dat avgout avghb.dat
```

2. pyhon3 hbond.py

```
import collections


input_file    = "avghb.dat"

output_file = "avghb-liph.txt"


def extract_residue(label):
    """
    'ASP_119@OD1' → 'ASP\t119'
    """
    res = label.split("@")[0]          # ASP_119
    name, num = res.split("_")        # ASP, 119
    return f"{name}\t{num}"             # ASP<TAB>119


occupancy = collections.defaultdict(float)


with open(input_file) as f:
```

```
    for line in f:
        if line.startswith("#") or not line.strip():
            continue

        cols = line.split()
        # cpptraj 格式：
        # Acceptor DonorH Donor Frames Frac AvgDist AvgAng
        # 索引 0        1          2        3          4

        acceptor = cols[0]
        donor    = cols[2]
        frac     = float(cols[4])    # Frac

        res1 = extract_residue(acceptor)
        res2 = extract_residue(donor)

        # 无序残基对（A-B 与 B-A 合并）
        key = tuple(sorted([res1, res2]))

        occupancy[key] += frac

# 输出
with open(output_file, "w") as out:
    out.write("Residue1\tResidue2\tOccupancy\n")
    for (r1, r2), occ in sorted(occupancy.items()):
        out.write(f"{r1}\t{r2}\t{occ:.4f}\n")
```

3. python3 masaike.py

```
#!/usr/bin/env python3
```

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl

# ------------------------------
#  参数
# ------------------------------
N = 309                                  #  残基总数
filename = 'avghb-liph.txt'      #  输入文件
scale = 4                                #  数据格子放大倍数

# ------------------------------
#  读取数据
# ------------------------------
data_list = []    #  存储  (i, j, z)
with open(filename, 'r') as f:
    for line in f:
        parts = line.split()
        if len(parts) != 5:
            continue
        res1, idx1, res2, idx2, count = parts
        i = int(idx1) - 1
        j = int(idx2) - 1
        z = float(count)

        #  统一到上三角
        if i > j:
            i, j = j, i
```

```
            data_list.append((i, j, z))


# ------------------------------
# 颜色映射
# ------------------------------
z_max = max([z for _, _, z in data_list])
cmap = plt.cm.hot                              # 亮色 colormap
norm = mpl.colors.Normalize(vmin=0, vmax=z_max)


# ------------------------------
# 按 z 值排序：小的先画，大的最后画
# ------------------------------
data_list.sort(key=lambda x: x[2])   # 按 z 排序


# ------------------------------
# 绘图
# ------------------------------
fig, ax = plt.subplots(figsize=(12,12), dpi=150)
ax.set_facecolor('black')   # 黑色背景


for i, j, z in data_list:
    color = cmap(norm(z))
    rect = plt.Rectangle(
        (j - 0.5*(scale-1), i - 0.5*(scale-1)),
        scale, scale,
        color=color, ec='gray', lw=0.05
    )
    ax.add_patch(rect)
```

```
# 坐标轴
ax.set_xlim(0, N)
ax.set_ylim(0, N)
ax.set_xticks(np.arange(0, N, 20))
ax.set_yticks(np.arange(0, N, 20))
ax.tick_params(labelsize=20)                    # 坐标刻度字体更大
ax.set_xlabel('Residue index', fontsize=22)
ax.set_ylabel('Residue index', fontsize=22)

# 颜色条
sm = mpl.cm.ScalarMappable(cmap=cmap, norm=norm)
sm.set_array([])
cbar = fig.colorbar(sm, ax=ax, ticks=np.arange(0, z_max+1, 1))
cbar.set_label('H-bond count', fontsize=20)
cbar.ax.tick_params(labelsize=18)

plt.tight_layout()
plt.savefig('hbond_mosaic_blackbg.png', dpi=300)
#plt.show()
```