# 一点背景介绍

## Spot ：

在空间转录组学中，组织切片被放置在一个覆盖有捕获点的芯片上。

每个 spot 是一个微小的物理位置（通常是 55µm 直径），它捕获该位置下方一小块组织区域内的 mRNA 分子。

每个 spot 本质上代表了一个“数据点”，它包含了该位置捕获到的所有 RNA 的测序信息（通常是数百个细胞或一个微环境）。

对于每个 spot，计算在该 spot 中检测到表达（即计数大于 0）的基因数量。这就是该 spot 的基因数。

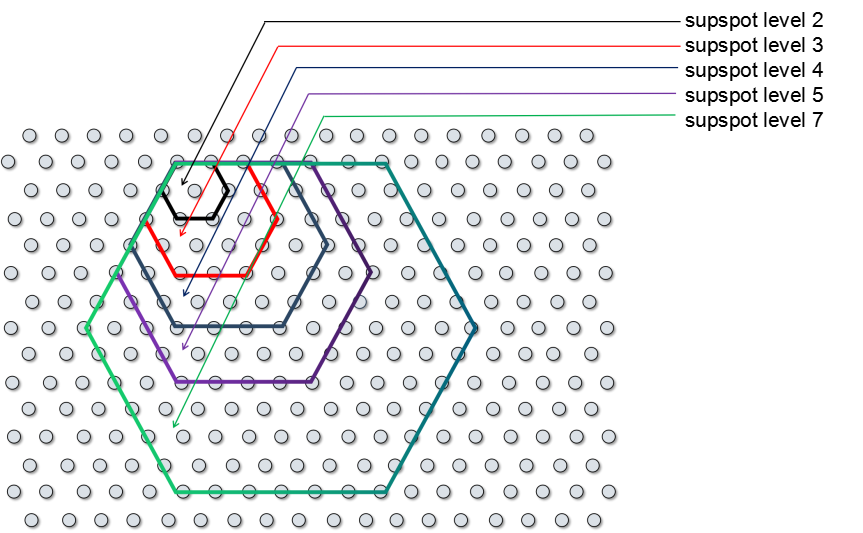
## supspot：

由于单个 Visium spot 可能覆盖多个细胞类型或空间结构，有时为了提高信噪比、减少技术噪音或获得更稳健的表达谱，分析人员会将相邻的多个原始 spot 在计算上合并成一个更大的单元。

这个由多个原始 spot 合并而成的、更大的分析单元就被称为一个 supspot。

## 多级分辨率

那么多级分辨率具体是怎么回事呢？下图是多级分辨率实现的说明。level 1 就是单独的 一个 spot 点，level 2 是把中心点及与其相距为（2-1） 的周围 spot 点合并为 一个 ，同理 level 313 就是合并周围相应的 level – 1 距离内的 spot 为 一个点。



# supspot点的中位基因数查看

输入数据文件格式:adata.h5ad

## 测试：

import scanpy as sc

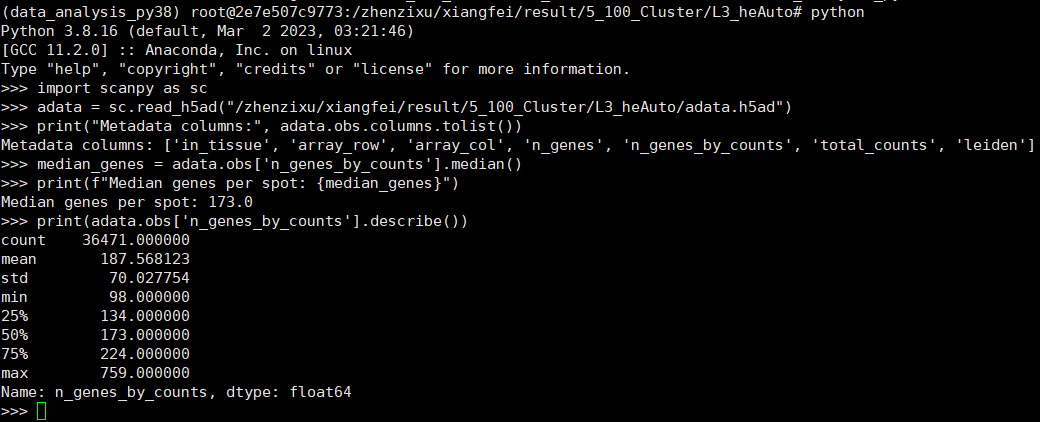
adata = sc.read\_h5ad("path") #例：adata = sc.read\_h5ad("/result/5\_100\_Cluster/L3\_heAuto/adata.h5ad")

print("Metadata columns:", adata.obs.columns.tolist())

median\_genes = adata.obs['n\_genes\_by\_counts'].median()

print(f"Median genes per spot: {median\_genes}")

print(adata.obs['n\_genes\_by\_counts'].describe())



某路径下

import scanpy as sc

# 加载数据

adata = sc.read\_h5ad("path")

# 查看所有spots的基因数分布

print("Metadata columns:", adata.obs.columns.tolist())

median\_genes = adata.obs['n\_genes\_by\_counts'].median()

print(f"Median genes per spot: {median\_genes}")

print(adata.obs['n\_genes\_by\_counts'].describe())

# 检查聚类标签

if 'leiden' in adata.obs.columns:

cluster\_label = 'leiden'

elif 'louvain' in adata.obs.columns:

cluster\_label = 'louvain'

elif 'cluster' in adata.obs.columns:

cluster\_label = 'cluster'

else:

cluster\_label = None

# 计算聚类中位基因数

if cluster\_label:

cluster\_median = adata.obs.groupby(cluster\_label)['n\_genes\_by\_counts'].median()

print(f"\nMedian genes per cluster ({cluster\_label}):")

print(cluster\_median)

else:

print("No clustering information found")

## 同一路径下所有层级结果展示

例如所有层级数据保存在/xiangfei/result/5\_100\_Cluster路径，那么此路径为基础路径，在基础路径下分层文件名命名为Lx\_heAuto，例L3\_heAuto

import scanpy as sc

import pandas as pd

import os

import numpy as np

# 基础路径

base\_path = "/xiangfei/result/5\_100\_Cluster"

# 所有要处理的分层

levels = [f"L{i}" for i in range(1, 10)] + ["L18"]

# 存储结果的列表

results = []

# 遍历所有水平

for level in levels:

# 构建文件路径 - 添加备用路径检查

file\_path1 = os.path.join(base\_path, f"{level}\_heAuto", "adata.h5ad")

file\_path2 = os.path.join(base\_path, level, "adata.h5ad") # 备用路径

# 检查文件是否存在

if os.path.exists(file\_path1):

file\_path = file\_path1

elif os.path.exists(file\_path2):

file\_path = file\_path2

else:

print(f"\n⚠️ File not found for {level}:")

print(f"Tried: {file\_path1}")

print(f"Tried: {file\_path2}")

continue

print(f"\nProcessing {level}...")

print(f"Loading data from: {file\_path}")

try:

# 加载数据

adata = sc.read\_h5ad(file\_path)

# 确保有基因计数数据

if 'n\_genes\_by\_counts' not in adata.obs.columns:

print(f"Calculating QC metrics for {level}...")

sc.pp.calculate\_qc\_metrics(adata, inplace=True)

# 计算整体中位基因数

overall\_median = adata.obs['n\_genes\_by\_counts'].median()

# 查找聚类标签

cluster\_label = None

for possible\_label in ['leiden', 'louvain', 'cluster']:

if possible\_label in adata.obs.columns:

cluster\_label = possible\_label

break

# 计算聚类中位基因数

cluster\_median = None

if cluster\_label:

cluster\_median = adata.obs.groupby(cluster\_label)['n\_genes\_by\_counts'].median()

# 添加到结果列表

results.append({

"Level": level,

"Overall\_Median": overall\_median,

"Cluster\_Median": cluster\_median

})

print(f"✅ Processed {level}: Overall Median = {overall\_median}")

# 释放内存

del adata

except Exception as e:

print(f"❌ Error processing {level}: {str(e)}")

import traceback

traceback.print\_exc()

# 创建结果DataFrame

results\_df = pd.DataFrame(results)

# 输出所有结果

print("\n" + "="\*50)

print("Final Results:")

print("="\*50)

if not results\_df.empty:

# 打印汇总表

print("\nSummary Table:")

print(results\_df[["Level", "Overall\_Median"]])

# 打印每个层级的聚类详细结果

print("\n" + "="\*50)

print("Detailed Cluster Medians:")

print("="\*50)

for \_, row in results\_df.iterrows():

print(f"\nLevel: {row['Level']}")

print(f"Overall Median Genes: {row['Overall\_Median']:.2f}")

if row['Cluster\_Median'] is not None and not row['Cluster\_Median'].empty:

print("\nCluster Median Genes:")

print(row['Cluster\_Median'])

else:

print("No cluster information available")

# 保存结果到CSV

output\_path = os.path.join(base\_path, "median\_genes\_summary.csv")

results\_df.to\_csv(output\_path, index=False)

print(f"\nResults saved to: {output\_path}")

else:

print("❌ No results to display - check file paths and errors")

## result示例

