

生境分析解决方案说明文档 (Kmeans)

1. 背景：K-means 聚类法是一种基于距离最小化准则的无监督非结构化分区聚类技术。

其目的是将数据空间 X 划分为多个簇，使得 X 的每个观测值都属于离质心最近的簇。

医学影像中的 K-means 应用：以体素强度之间的平方欧几里得距离作为相似性度量，

根据每个聚类中的个体素根据它们的相似性和差异性进行分组。

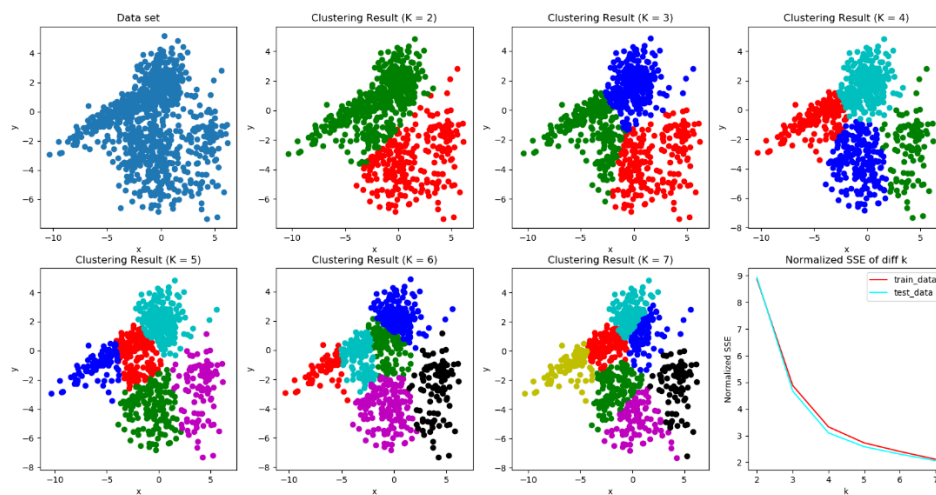


Figure 1: Kmeans 示意图

2. 解决方案：

(1) 首先在原始图像上勾画大区域 ROI，保存为 nifty 格式文件。

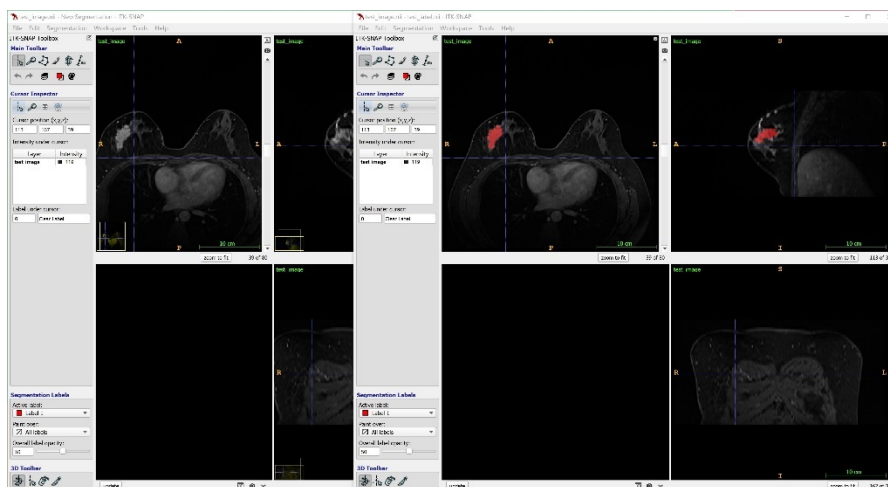


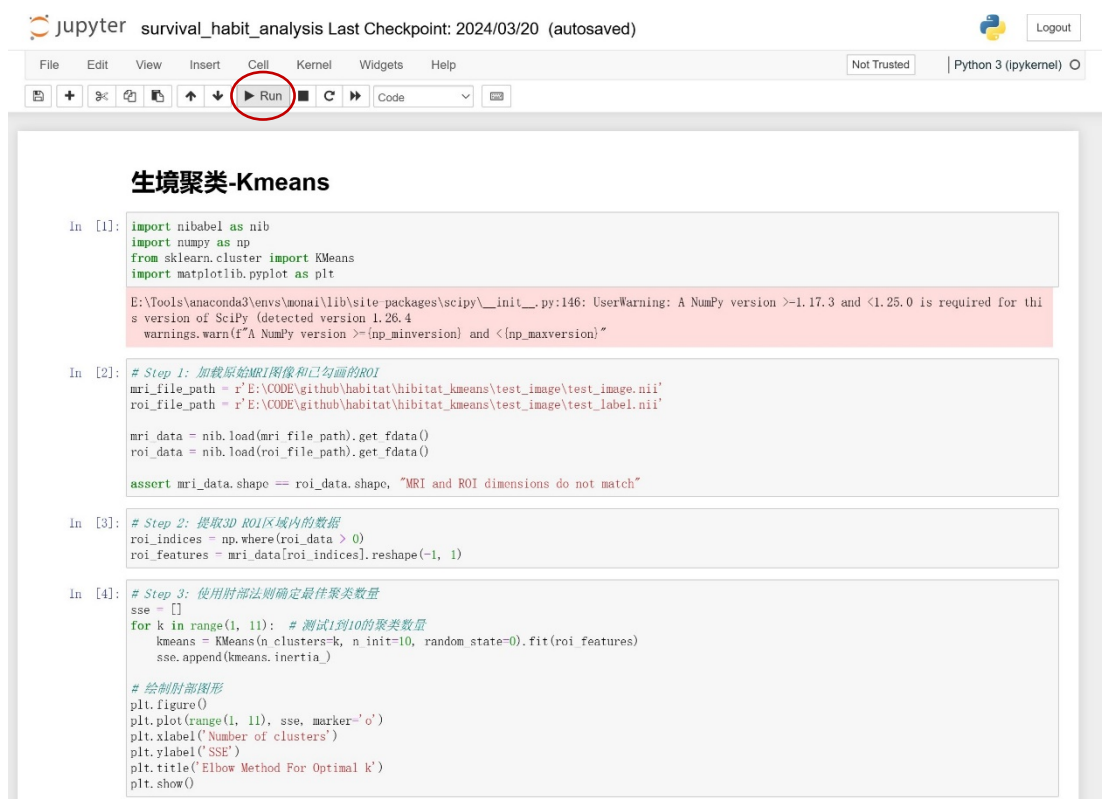
Figure 2: 勾画大区域 ROI

(2) 将 ROI 命名为 “原始图像名_label.nii ”, 分别获取原始图像路径与 ROI 路径例如:

"E:\Code\github\habitat\hibitat_kmeans\test_image\test_image.nii"

"E:\Code\github\habitat\hibitat_kmeans\test_image\test_label.nii"

(3) 使用 anaconda 打开 jupyter notebook, 打开 survival_habit_analysis 代码文件。



```
In [1]: import nibabel as nib
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

E:\Tools\anaconda3\envs\monai\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy version >=1.17.3 and <1.25.0 is required for this version of SciPy (detected version 1.26.4)
warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}")

In [2]: # Step 1: 加载原始MRI图像和已勾画的ROI
mri_file_path = r'E:\Code\github\habitat\hibitat_kmeans\test_image\test_image.nii'
roi_file_path = r'E:\Code\github\habitat\hibitat_kmeans\test_image\test_label.nii'

mri_data = nib.load(mri_file_path).get_fdata()
roi_data = nib.load(roi_file_path).get_fdata()

assert mri_data.shape == roi_data.shape, "MRI and ROI dimensions do not match"

In [3]: # Step 2: 提取3D ROI区域内的数据
roi_indices = np.where(roi_data > 0)
roi_features = mri_data[roi_indices].reshape(-1, 1)

In [4]: # Step 3: 使用肘部法则确定最佳聚类数量
sse = []
for k in range(1, 11): # 测试1到10的聚类数量
    kmeans = KMeans(n_clusters=k, n_init=10, random_state=0).fit(roi_features)
    sse.append(kmeans.inertia_)

# 绘制肘部图形
plt.figure()
plt.plot(range(1, 11), sse, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
plt.title('Elbow Method For Optimal k')
plt.show()
```

Figure 3: 代码界面

(4) 逐个代码块点击 Figure3 中圈红的 Run, 会生成肘部法则图像。

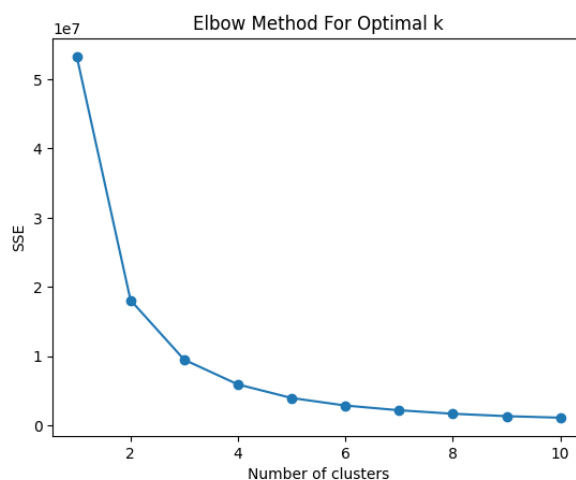


Figure 4: 肘部法则图像

- (5) 根据肘部法则图像选择最佳聚类个数，本例可以看出，横坐标 4 之前的 sse 呈现出陡降态势，4 之后是缓降。因此使用 4 作为最佳聚类个数填入代码块：

```
In [5]: # 根据肘部法则选择最佳聚类数量，这里需要人工观察决定
        optimal_clusters = 4 # 假设最佳聚类数量为4
```

Figure 5: 选择最佳聚类数

- (6) 再次逐步点击 Run。在代码块 8, new_roi_file_path 部分填入想保存的位置与保存的名称，须以 nii 格式结尾，且不与 Label 文件重名。

```
In [6]: ## Step 4: 应用聚类算法进行生境分析
        # n_clusters = 3 # 定义聚类数量
        # kmeans = KMeans(n_clusters=n_clusters, n_init=10, random_state=0).fit(roi_features.reshape(-1, 1))

        ## 获取聚类标签
        # labels = kmeans.labels_

        ## 重建3D ROI以展示聚类结果
        # clustered_roi = np.zeros_like(roi_data)
        # clustered_roi[roi_indices] = labels + 1 # 使用+1以避免标签为0

        # Step 4: 应用聚类算法并保存聚类结果
        kmeans = KMeans(n_clusters=optimal_clusters, n_init=10, random_state=0).fit(roi_features)
        labels = kmeans.labels_

        ## 重建3D ROI以展示聚类结果
        clustered_roi = np.zeros_like(roi_data)
        clustered_roi[roi_indices] = labels + 1

In [8]: # Step 5: 保存分割后的子ROI为NIFTI格式
        new_roi_img = nib.NiftiImage(clustered_roi, affine=nib.load(mri_file_path).affine)
        new_roi_file_path = r'E:\CODE\github\habitat\kmeans\test_image\subroi_1.nii'
        nib.save(new_roi_img, new_roi_file_path)
```

Figure 6: 生境生成部分

- (7) 划分图像查看：

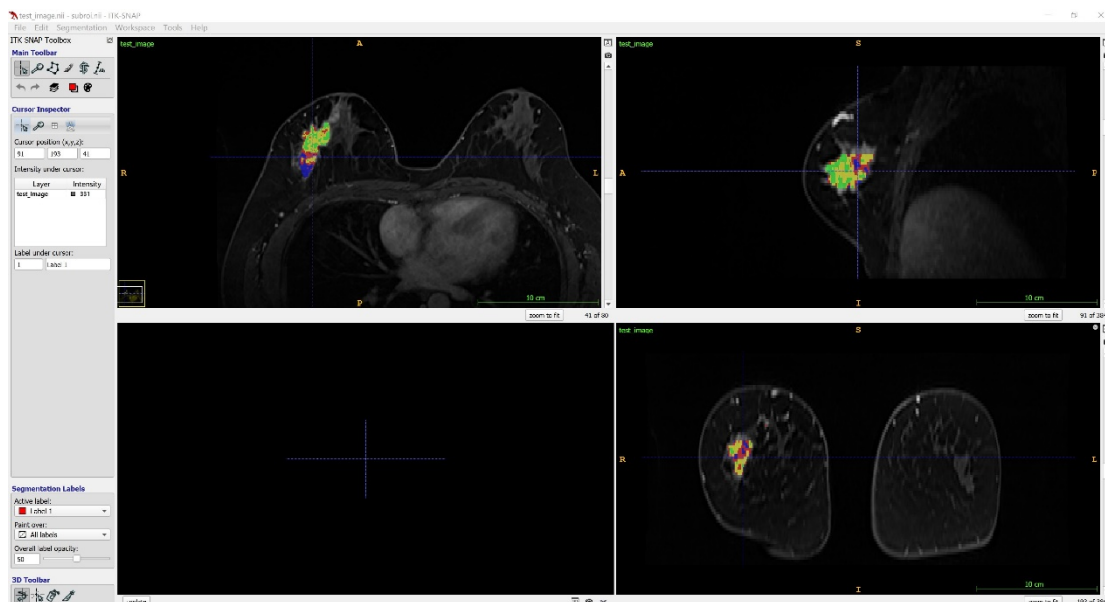


Figure 7: 不同生境区

- (8) 根据不同生境区可进行后续量化特征提取与分析。Kmeans 生境区分割解决方案完成！