Lab 3

Lab 3 — New Python Functions (Hierarchical Clustering)

This lab introduces **hierarchical (agglomerative) clustering** and **dendrograms**. Many functions will look familiar from earlier labs, but they are used in new ways.

1. Distance computation

`scipy.spatial.distance.pdist`

Computes **pairwise distances** between observations.

```
from scipy.spatial.distance import pdist
D = pdist(X, metric="euclidean")
```

Note: Output is a *condensed* distance vector (not a square matrix).

`scipy.spatial.distance.squareform`

Converts between condensed and square distance matrices.

```
from scipy.spatial.distance import squareform
squareform(D)
```

Useful for inspecting distances and matching by-hand calculations.

2. Hierarchical clustering models (sklearn)

`sklearn.cluster.AgglomerativeClustering`

Fits a bottom-up hierarchical clustering model.

```
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(
    distance_threshold=0,
    n_clusters=None,
    linkage="complete"
)
hc.fit(X)
```

Key arguments:

- `linkage`: `"single"`, `"complete"`, `"average"`
- `distance_threshold=0` and `n_clusters=None` → required to build the *full tree*

Note: this does **not** directly give a dendrogram.

3. Converting to a dendrogram (ISLP)

`ISLP.cluster.compute_linkage`

Converts a fitted `AgglomerativeClustering` object into a **SciPy linkage matrix**.

```
from ISLP.cluster import compute_linkage
Z = compute_linkage(hc)
```

This linkage matrix is what `dendrogram()` needs.

Note: This function is specific to ISLP and mirrors the Chapter 12 labs.

4. Drawing dendrograms

`scipy.cluster.hierarchy.dendrogram`

Plots a dendrogram from a linkage matrix.

```
from scipy.cluster.hierarchy import dendrogram
dendrogram(Z, labels=labels)
```

A few options:

- `labels=` — names of observations
- `color_threshold=-np.inf` (for black dendrograms)

5. Cutting the dendrogram into clusters

`scipy.cluster.hierarchy.cut_tree`

Extracts cluster assignments from a dendrogram.

```
from scipy.cluster.hierarchy import cut_tree
clusters = cut_tree(Z, n_clusters=3)
```

6. Scaling variables

`sklearn.preprocessing.StandardScaler`

Standardizes variables before distance-based clustering.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

7. Linkage without sklearn

`scipy.cluster.hierarchy.linkage`

Performs hierarchical clustering directly from distances.

```
from scipy.cluster.hierarchy import linkage
Z = linkage(D, method="complete")
```