统计建模第四次作业：聚类问题解决

闫然 龚昌思宇 柴行

2024-12-10

在第13周的统计建模课上，老师遗留了7个问题让小组解决。我们小组将以iris数据集为例，解决这些聚类问题。

数据集介绍

Iris也称鸢尾花卉数据集，是一类多重变量分析的数据集。通过花萼长度，花萼宽度，花瓣长度，花瓣宽度4个属性预测鸢尾花卉属于（Setosa(山鸢尾)，Versicolour(杂色鸢尾)，Virginica(维吉尼亚鸢尾)）三个种类中的哪一类。

鸢尾花（iris）数据集，它共有4个属性列和一个品种类别列：sepal length（萼片长度）、sepal width（萼片宽度）、petal length（花瓣长度）、petal width （花瓣宽度），单位都是厘米。3个品种类别是Setosa、Versicolour、Virginica，样本数量150个，每类50个。

```
#载入tidyverse包，包中整合了ggplot2、tibble、dplyr等数据处理包。
library(tidyverse)

#载入explore包，包中有iris数据。
library(explore)
data = data("iris")

#展示数据
head(iris,n = 10)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1         3.5          1.4         0.2  setosa
## 2           4.9         3.0          1.4         0.2  setosa
## 3           4.7         3.2          1.3         0.2  setosa
## 4           4.6         3.1          1.5         0.2  setosa
## 5           5.0         3.6          1.4         0.2  setosa
## 6           5.4         3.9          1.7         0.4  setosa
## 7           4.6         3.4          1.4         0.3  setosa
## 8           5.0         3.4          1.5         0.2  setosa
## 9           4.4         2.9          1.4         0.2  setosa
## 10          4.9         3.1          1.5         0.1  setosa
```

k-means聚类问题解决

首先将iris数据标准化后进行K-means聚类，然后依次解决K-means聚类中遗留的三个问题。

```
#移除目标列，仅保留数值变量
iris_data = iris[, -5]

#标准化数据
scaled_iris = scale(iris_data)
set.seed(123)

#应用Kmeans算法
kmeans_result = kmeans(scaled_iris, centers = 3, nstart = 20)
```

1.关于返回结果中totss、withinss、tot.withinss、betweenss的关系

在官方的解释文档中，我们可以看到对对应变量的解释： totss： 总离差平方和，即所有数据点到整个数据集均值的欧几里得距离平方和。
withinss： 一组数据，是每个簇内的平方和，即每个簇中数据点到该簇质心的欧几里得距离平方和。
tot.withinss：是所有簇的withinss之和，反映了聚类结果的紧密性。
betweenss： 簇间平方和，代表簇之间的离差平方和，反映了不同簇的分离程度。

totss	The total sum of squares.
withinss	Vector of within-cluster sum of squares, one component per cluster.
tot.withinss	Total within-cluster sum of squares, i.e. sum(withinss).
betweenss	The between-cluster sum of squares, i.e. totss-tot.withinss.

官方文档对四者的解释

根据定义，很显然有：sum(withinss) = tot.withinss。下面是对剩下三者关系的推导，推导类似于方差分析中的方差分解的推导：

令i代表簇的类别数，k_i代表第i个簇中的样本数，则有：

$$TOTSS = \sum_{i=1}^{n}\sum_{j=1}^{k_i}(Y_{ij} - \bar{Y})^2$$

化简有：

$$TOTSS = \sum_{i=1}^{n}\sum_{j=1}^{k_i}(Y_{ij} - \bar{Y}_i)^2 + \sum_{i=1}^{n}\sum_{j=1}^{k_i}(\bar{Y}_i - \bar{Y})^2$$

根据定义有：

$$BETWEENSS = \sum_{i=1}^{n}\sum_{j=1}^{k_i}(Y_{ij} - \bar{Y}_i)^2, TOT.WITHINSS = \sum_{i=1}^{n}\sum_{j=1}^{k_i}(\bar{Y}_i - \bar{Y})^2$$

故：

$$TOTSS = BETWEENSS + TOT.WITHINSS$$

故最终可以用两个式子描述四者的关系：

$$TOTSS = BETWEENSS + TOT.WITHINSS$$

$$TOT.WITHINSS = sum(WITHINSS)$$

下面利用代码验证：

```
#验证第一个式子
cat("totss为",kmeans_result$totss,
    "。too.withinss和betweenss分别为",kmeans_result$tot.withinss,kmeans_result$betweenss,
    "二者的和为",kmeans_result$tot.withinss+kmeans_result$betweenss,
    "，经验证加和关系成立。")
```

```
## totss为 596 。too.withinss和betweenss分别为 138.8884 457.1116 二者的和为 596 ，经验证加和关系成立。
```

```
cat("\n\n")
```

```
#验证第二个式子
cat("withinss为",kmeans_result$withinss,
    "，tot.withinss为",kmeans_result$tot.withinss,
    "，withinss的和为",sum(kmeans_result$withinss),
    "。\n经验证tot.withinss与withinss的和相等。")
```

```
## withinss为 47.35062 44.08754 47.45019 ，tot.withinss为 138.8884 ，withinss的和为 138.8884 。
## 经验证tot.withinss与withinss的和相等。
```

2. 标准化后真实数据尺度的类中心问题

从结果可以看到打印出的类中心是标准化后的尺度，对于直观理解数字的含义以及解释数字都加大了难度。如果能根据这个结果返回真实数据尺度的类中心，就能解决这个问题。

```
#打印标准化后的类中心
kmeans_result$centers
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1  -1.01119138  0.85041372   -1.3006301  -1.2507035
## 2  -0.05005221 -0.88042696    0.3465767   0.2805873
## 3   1.13217737  0.08812645    0.9928284   1.0141287
```

标准化是将原有数据列的每一个数据进行了线性变化：每个数据减去列均值后除以列方差。因此想要获得真实数据尺度的中心，只需要获得原始数据列的均值、方差，然后用标准化后的中心数据反解真实数据即可。代码如下：

```
#获取原始数据的均值和标准差
means = attr(scale(iris_data), "scaled:center")
sds = attr(scale(iris_data), "scaled:scale")

#获取标准化后的类中心
standardized_centers = kmeans_result$centers
kmeans_resul=kmeans_result#令kmeans_resul存储原本变量
#反标准化
true_centers = sweep(standardized_centers, 2, sds, "*")  # 乘以标准差
true_centers = sweep(true_centers, 2, means, "+")        # 加上均值

#查看反标准化后的类中心
print(true_centers)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     5.006000    3.428000     1.462000    0.246000
## 2     5.801887    2.673585     4.369811    1.413208
## 3     6.780851    3.095745     5.510638    1.972340
```

3. Kmeans聚类结果可视化

Kmeans的聚类结果可以通过下列代码查看：

```
kmeans_result$cluster
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 2 2 2 3 2 2 2 2 2 2 2 3 2 2 2 2 3 2 2 2
##  [75] 2 3 3 3 2 2 2 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3
## [112] 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 3 3 3 3 3 2 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [149] 3 2
```

但是这种结果的表现形式显然不利于我们直观的去观察聚类结果。所以如果可以通过可视化的方式将聚类结果直观的展示出来是最好的。通常被聚类的数据集都是高维度数据集，因此无法直接的用二维散点图或者三维散点图画出，因此需要先将数据降维。常用的聚类结果降维后可视化方法有PCA方法和t-SNE方法。

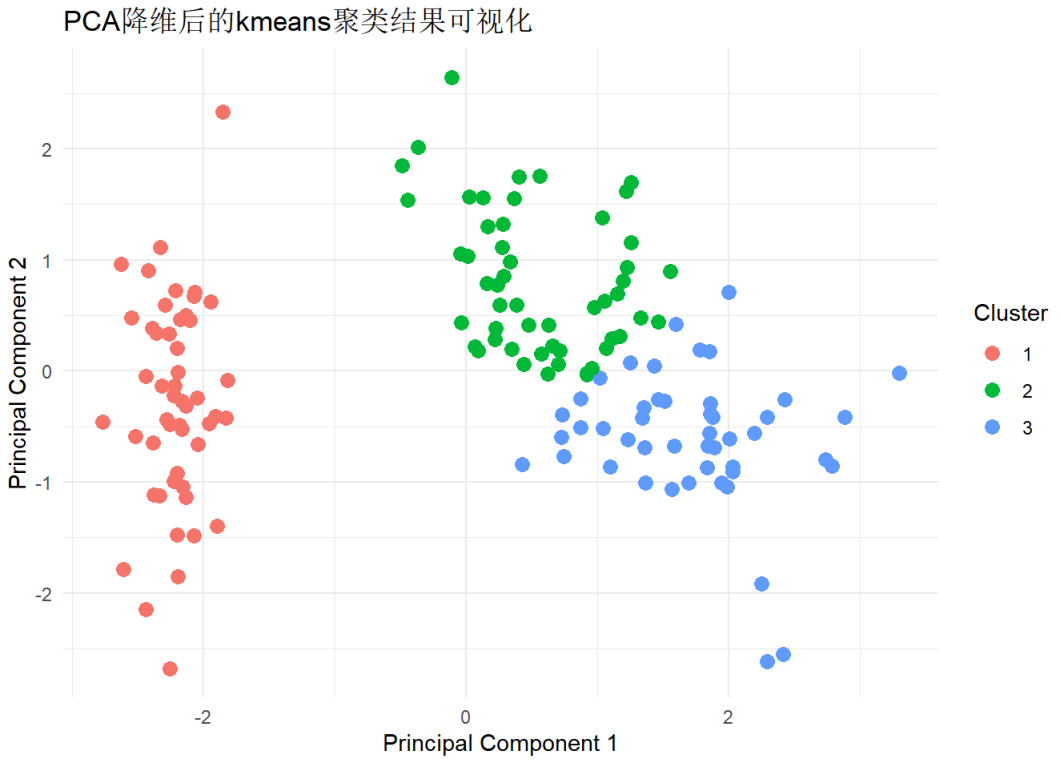
PCA方法降维

以iris数据集为例，可以使用PCA将数据从四维降到二维或者三维。PCA通过提取数据中最大方差的方向，可以有效保留数据的主要特征，并用较少的维度表示数据。这里将数据降到二维后可视化：

```
#执行PCA
pca = prcomp(iris[, 1:4], center = TRUE, scale. = TRUE)

#提取前两个主成分
pca_data = data.frame(pca$x[, 1:2], Cluster = as.factor(kmeans_resul$cluster))

#可视化PCA结果
library(ggplot2)
ggplot(pca_data, aes(x = PC1, y = PC2, color = Cluster)) +
  geom_point(size = 3) +
  labs(title = "PCA降维后的kmeans聚类结果可视化", x = "Principal Component 1", y = "Principal Component 2") +
  theme_minimal()#绘制散点图进行可视化
```



除此之外，还可以进一步查看PCA的方差和方差解释比例。从summary的结果可以看到：PC1 和 PC2 的 Proportion of Variance 分别是0.7396和 0.2285，意味着 PC1 能解释约 73.96% 的方差，而 PC2 能解释约 22.85% 的方差。 Cumulative Proportion表示前几个主成分累积解释的总方差，前两个主成分累积解释了95.81%的方差。说明在这个例子里，降到两维的效果还是比较好的，即降低了维数，也没有损失太多信息。

```
#查看主成分的方差和方差解释比例
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.7084 0.9560 0.38309 0.14393
## Proportion of Variance 0.7296 0.2285 0.03669 0.00518
## Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

t-SNE方法降维

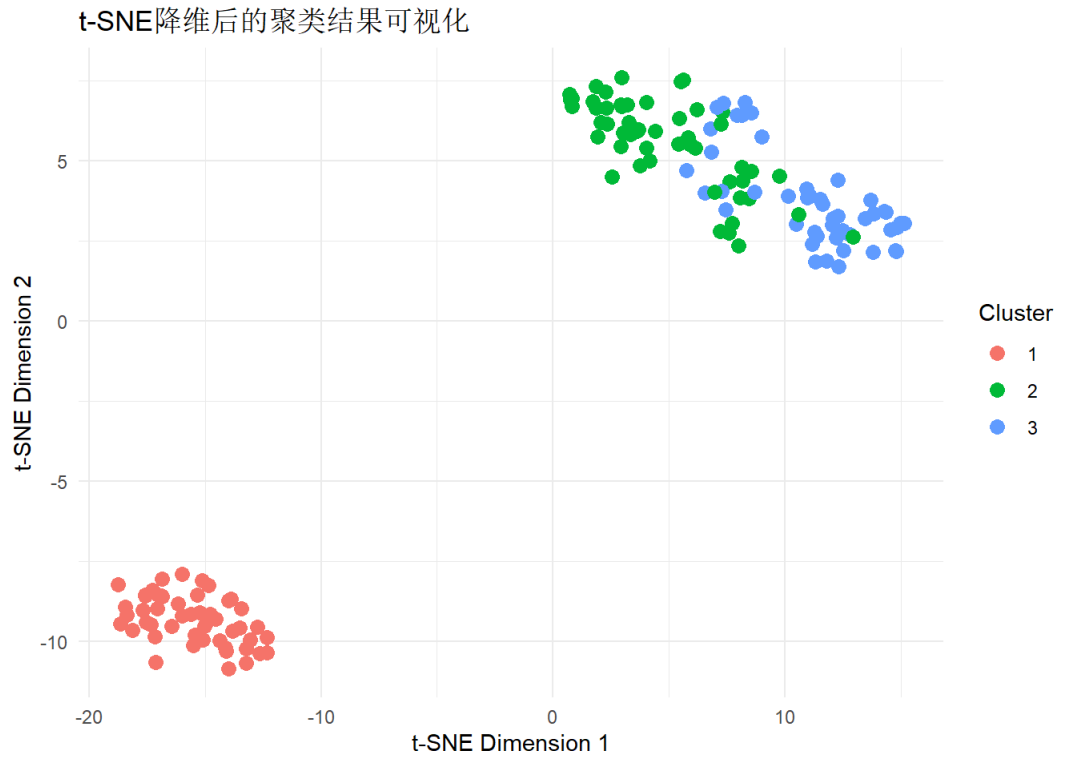
t-SNE法是一种非线性降维算法，主要用于高维数据的可视化。t-SNE法能够较好地保留高维空间中点与点之间的局部结构关系。这里以iris数据集为例将数据用t—SNE法降到二维后可视化：

```
library(Rtsne)#该包主要用于实现 t-SNE算法,特别适合处理高维数据并将其映射到二维或三维
iris_unique = unique(iris_data)#由于Rtsne不能使用含有重复值的数据集，所以要用unique函数去除
#找出被删除的行索引
removed_index = setdiff(1:nrow(iris_data), 1:nrow(iris_unique))

tsne_result = Rtsne(iris_unique, dims = 2)#进行t-sne降维

library(ggplot2)
#将t-SNE结果和K-means聚类标签合并
tsne_df = data.frame(tsne_result$Y)#将降维后的结果Y列生成数据集的形式
tsne_df$Species = iris$Species[!is.element(1:nrow(iris), removed_index)]
#kmeans_result$cluster中删除相应的聚类标签，因为输出结果有150行而去除出后的iris_unique只有149行，所以要删掉多余的一行
kmeans_result$cluster = kmeans_result$cluster[-removed_index]
#因子化聚类结果
tsne_df$Cluster = as.factor(kmeans_result$cluster)

#可视化K-means聚类结果
ggplot(tsne_df, aes(x = X1, y = X2, color = Cluster)) +
  geom_point(size = 3) +
  labs(title = "t-SNE降维后的聚类结果可视化", x = "t-SNE Dimension 1", y = "t-SNE Dimension 2") +
  theme_minimal()
```



总结

对于降维方法的选择，如果希望解释性强，优先选择PCA降维后的散点图。如果注重局部结构的保持，可用t-SNE法。

层次聚类法问题解决

层次聚类遗留了三个问题，首先我们解决层次聚类法是否需要先将数据标准化的问题，然后再解决剩下的两个问题。

1.层次聚类法是否需要先将数据标准化

首先看R中常用的层次聚类包是否内置标准化方法： 在 R 中，常用的层次聚类包包括 cluster、factoextra 和 pvclust。

cluster 包中 agnes 和 diana 函数支持多种层次聚类方法，但不内置数据标准化功能； factoextra 包提供集成的标准化（默认启用）和可视化功能，可便于快速探索数据； pvclust 包可用于评估聚类结果的稳定性，提供自助法（Bootstrap）统计置信度，但需要用户手动标准化。 这样就解决了用R在层次聚类时，如果有标准化需求，需不需要自己手动标准化的问题，也就是需要根据对应的包去决定需不需要在函数外说手动标准化。

数据标准化的意义在于：如果数据集的特征量纲差异较大，则这些特征在计算距离时会主导结果，从而导致聚类偏差。标准化处理能够消除量纲影响，使所有特征对聚类过程的贡献均等。那是不是代表标准化在所有情境下都是必须的呢？为了解答这个问题，我们做了一个聚类实验，将鸢尾花数据集分成标准化的和未标准化的两个数据集，分别聚类后计算聚类的ARI和Purity，具体代码如下：

```
library(datasets) # 加载 R 的内置数据集包
library(cluster) # 加载 cluster 包，提供层次聚类相关函数
library(flexclust) # 加载 flexclust 包，knit Rmd时需要（未使用但可扩展）
library(mclust) # 加载 mclust 包，knit Rmd时需要（未使用但可扩展）
set.seed(123)
#加载数据集&计算标准化与否的结果
data(iris)
#移除目标列 Species，保留数值型特征列
iris_data = iris[, -5]
# 数据标准化
iris_scaled = scale(iris_data)
# 计算欧几里得距离矩阵
distance_matrix_raw = dist(iris_data, method = "euclidean")
distance_matrix_scaled = dist(iris_scaled, method = "euclidean")

# 进行层次聚类（Ward法）Ward.D2 方法通过最小化类内方差进行合并，是常用的层次聚类方法之一
hc_raw = hclust(distance_matrix_raw, method = "ward.D2")
hc_scaled = hclust(distance_matrix_scaled, method = "ward.D2")
# 切割聚类树，得到3个簇
cluster_raw = cutree(hc_raw, k = 3)
cluster_scaled = cutree(hc_scaled, k = 3)

#对比标准化与否的聚类结果
#使用 `factor` 函数将字符标签转换为因子，再使用 `as.numeric` 提取对应的数字编码
true_labels = as.numeric(factor(iris$Species))
#计算调整兰德指数（Adjusted Rand Index, ARI），用于评估聚类结果与真实标签的相似性
#ARI 的取值范围为 [-1, 1]，值越高表示聚类结果与真实分类越吻合
ari_raw = adjustedRandIndex(cluster_raw, true_labels)
ari_scaled = adjustedRandIndex(cluster_scaled, true_labels)
#计算聚类纯度（Purity），用于评估聚类的准确性
#通过比较每个簇中占比最多的真实类别，并计算其比例
purity_raw <- mean(apply(table(cluster_raw, true_labels), 1, max)) / length(true_labels)
purity_scaled <- mean(apply(table(cluster_scaled, true_labels), 1, max)) / length(true_labels)

#输出结果
cat(" 未标准化的结果:\n","ARI =",
    ari_raw, "\n","Purity =",
    purity_raw, "\n","\n","标准化的结果:\n",
    "ARI =", ari_scaled, "\n",
    "Purity =", purity_scaled, "\n")
```

```
##  未标准化的结果:
##  ARI = 0.7311986
##  Purity = 0.2977778
##
##  标准化的结果:
##  ARI = 0.615323
##  Purity = 0.2755556
```

从打印结果我们可以看到：未标准化的结果，不管是ARI还是Purity，表现都比标准化后的数据集要好。从实用主义的角度来说，不进行标准化反而是最正确的。这也就是说，并不是所有数据在聚类前都必须标准化，或许不标准化的数据能在分类上取得更好的表现。为了进一步探究这个情况出现的原因，我们计算了各个鸢尾花类在各个特征上的均值：

```
#计算每个类在各特征上的均值
agg_summary = aggregate(cbind(Sepal.Length, Petal.Length, Sepal.Width, Petal.Width) ~ Species, data = iris, FUN = mean)
print(agg_summary)
```

```
##      Species Sepal.Length Petal.Length Sepal.Width Petal.Width
## 1     setosa        5.006        1.462       3.428       0.246
## 2 versicolor        5.936        4.260       2.770       1.326
## 3  virginica        6.588        5.552       2.974       2.026
```

究其原因发现鸢尾花数据集中setosa 类的Petal.Length和Petal.Width的均值明显低于versicolor和virginica类。这种数据不平衡性的显著差异为聚类提供了有力的区分依据，标准化后，这些特征的差异被压缩，使得setosa类的独特性减弱，聚类效果下降。

总的来说，实验结果表明：标准化的必要性取决于数据的特征分布：对于大多数数据集，特别是当特征的数值范围差异较大时，标准化通常有助于提高聚类效果，避免某些特征主导聚类过程。但对于像鸢尾花这样的数据集，特征之间的差异本身就足够明显，标准化可能会削弱这种差异，因此在这种情况下，未标准化的聚类效果反而更好。

2.dist矩阵在计算什么的距离

首先介绍dist函数： dist函数结构： **dist(x, method = “euclidean”, diag = FALSE, upper =FALSE, p = 2)。** *x*是所选数据集，*method*是计算距离类型，*diag*是逻辑值，指示*print.dist*函数是否应打印距离矩阵的对角线，*upper*是逻辑值，指示*print.dist*函数是否应打印距离矩阵的上三角部分，*p*是*Minkowski*距离的幂次。*p*=2即为欧氏距离。

在层次聚类中，dist函数用于计算数据集中各样本点之间的成对距离，这些距离用于构建聚类树（即层次聚类树）。 `dist()` 默认计算的是数据集中所有点之间的成对欧氏距离。欧氏距离是两个点之间的直线距离，计算公式为：

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2}$$

其中 x_i 和 x_j 是两个数据点，x_{ik} 和 x_{jk} 分别是这两个数据点在第 k 个维度上的值，n 是维度数（在这个例子中，$n = 4$，因为使用的是鸢尾花数据集的前四个特征）。 除了欧氏距离外， `dist` 函数还支持其他几种距离度量方法，包括：

曼哈顿距离（Manhattan distance）：也称为城市街区距离，是两个点在标准坐标系上的绝对轴距总和。公式为：

$$\sum_{i=1}^{n} |x_i - y_i|$$

*适用场景：*适用于维度间存在顺序关系或权重不同的情况。

切比雪夫距离（Chebyshev distance）：是两个点在标准坐标系上的坐标差的绝对值的最大值。公式为：

$$\max_{i=1}^{n} |x_i - y_i|$$

*适用场景：*在某些图像处理算法中，用于衡量像素点之间的最大差异。在某些优化问题中，当目标是最小化最大误差时，可以使用切比雪夫距离。

明可夫斯基距离（Minkowski distance）：是欧氏距离和曼哈顿距离的广义形式，公式为：

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

其中 p 是一个正整数参数。

*适用场景:*当数据集中的维度具有不同的重要性或权重时，通过调整p值来适应不同的权重分布。

下面进行举例说明dist函数的默认用法：

```
#生成3行5列的随机数矩阵（三个样本点）
set.seed(123)
aa=matrix(rnorm(15,0,1),c(3,5))

#计算距离
dist(aa,p=2)
```

```
##          1        2
## 2 2.442078
## 3 3.174394 2.690788
```

dist返回一个“dist”类的变量。从结果我们可以看到，第一个样本与第二个样本的距离为2.442078，第二个样本与第三个样本的距离为3.174394，第二个样本与第三个样本的距离为2.690788。

3.层次聚类法的聚类中心的确定

首先，类中心是指一个簇内所有数据点的几何平均值（即各维度的均值），它代表了这个簇的“中心位置”。在层次聚类中，类中心的计算方法和 K-means有很大不同，因为层次聚类的核心机制不依赖类中心，故层次聚类过程中并不计算类中心。

如果想计算类中心，在层次聚类完成后，可以利用K-means聚类的类中心确定思想计算层次聚类的类中心。具体计算思路是：对于聚类完成后形成的每个簇，取簇内所有数据点在各维度上的均值，作为该簇的类中心。具体代码如下：

```
#计算类中心（不标准化的聚类结果）
cluster_data_raw = cbind(iris_data, cluster_raw = cluster_raw)
#aggregate函数按簇（cluster_raw）分组，并对每个特征列计算均值
cluster_centers_raw = aggregate(. ~ cluster_raw, data = cluster_data_raw, FUN = mean)
print(cluster_centers_raw)
```

```
##   cluster_raw Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1           1     5.006000    3.428000     1.462000    0.246000
## 2           2     5.920312    2.751562     4.420312    1.434375
## 3           3     6.869444    3.086111     5.769444    2.105556
```

拓展性问题

除了和聚类有关的一些技术性问题外，还有一个拓展性的问题：怎样才能在使用大语言模型工具时，能够不谄媚提问人，获得更加理性、准确的回答?

这个问题也非常重要，大模型如今已经成为我们重要的生产力工具，因此更好的使用大模型也是一种重要的能力。作为用户，我们无法直接控制模型的参数，因此只能从提问的方式入手，让模型尽可能得回答准确。我们整理了以下可以提高模型准确性的方式：

1.可以通过提示引导模型进行推测，并要求它指出回答的不确定性和推理深度。例如，“你能提供这条信息的来源吗？”或者“你对这条回答的信心有多高？”。这样可以让模型明确其不确定性，避免误导。
2.在问题中加入引导语句，要求模型提供推理逻辑、理由。例如，在提出问题时，可以明确要求模型提供理由或思路：“根据你所知的市场趋势，如何分析某只股票未来的价格变化？”
3.可以通过多次交互细化回答，或者使用不同的模型交叉验证回答。例如，可以开多个不同窗口提问，比较回答的一致性。还可以使用不同的模型提问，比如豆包、文心一言、千问等，比较获得答案。