mathmodels 包使用手册

化繁为简的数学建模利器

张敬信

2025-07-27

目录

前言 **1**

mathmodels 包与本配套使用手册的核心特色 1

关于 mathmodels 包（当前 0.0.5 版本） 2

更多学习资源 4

第一部分　评价类算法 **5**

第一章　指标数据预处理 **7**

1.1　方向一致性处理 7

1.1.1　负向指标正向化 7

1.1.2　居中型指标正向化 8

1.1.3　区间型指标正向化 8

1.2　无量纲化处理 8

1.2.1　标准化 8

1.2.2　归一化 9

1.2.3　规范化 9

1.2.4　用于 GRA 的无量纲化 9

1.3　综合案例 10

第二章　层次分析法 **13**

2.1　算法步骤 13

2.1.1　建立层次结构 13

2.1.2　构造判断矩阵 14

2.1.3　计算权向量及一致性检验 15

2.1.4　计算组合权向量与组合一致性检验 16

2.2　案例：旅游地选择 16

2.2.1　建立层次结构 17

2.2.2　确定判断矩阵 17

2.2.3　基于 mathmodels 包求解 18

第三章 熵权法 **21**
3.1 算法步骤 . . . 21
3.2 案例：河流水质数据 . . . 23
3.2.1 准备数据 . . . 23
3.2.2 数据预处理 . . . 24
3.2.3 熵权法 . . . 24

第四章 CRITIC 法 **27**
4.1 算法步骤 . . . 27
4.2 改进：熵-CRITIC 法与不同相关系数 . . . 29
4.3 案例：河流水质数据 . . . 29
4.3.1 准备数据 . . . 29
4.3.2 数据预处理 . . . 30
4.3.3 CRITIC 及其改进权重 . . . 31

第五章 主成分赋权法 **34**
5.1 算法步骤 . . . 34
5.2 案例：河流水质数据 . . . 36
5.2.1 准备数据 . . . 36
5.2.2 数据预处理 . . . 37
5.2.3 主成分赋权 . . . 38

第六章 主客观组合赋权 **41**
6.1 四种主客观组合赋权法 . . . 41
6.2 R 实现 . . . 42

第七章 TOPSIS **44**
7.1 算法步骤 . . . 44
7.2 案例：河流水质评价 . . . 45
7.2.1 数据预处理 . . . 46
7.2.2 指标权重：熵权法 . . . 47
7.2.3 TOPSIS 评价 . . . 47

第八章 灰色关联分析 **49**
8.1 灰色关联度 . . . 49
8.1.1 数据无量纲处理 . . . 49
8.1.2 计算关联系数 . . . 49
8.1.3 计算灰色关联度 . . . 50
8.2 案例 1：运动员训练与成绩 . . . 50
8.2.1 创建数据 . . . 50
8.2.2 无量纲化处理 . . . 51

8.2.3 计算灰色关联度 . . . 51
8.3 案例 2：投资产出优势分析 . . . 52
8.3.1 创建数据 . . . 52
8.3.2 无量纲化处理 . . . 52
8.3.3 优势分析：批量计算灰色关联度 . . . 53
8.4 灰色关联评价 . . . 53
8.4.1 数据预处理 . . . 53
8.4.2 指标权重：熵权法 . . . 54
8.4.3 灰色关联评价 . . . 55

第九章 秩和比法 57
9.1 算法步骤 . . . 57
9.1.1 编秩 . . . 57
9.1.2 计算秩和比并排序 . . . 58
9.1.3 确定 RSR 的分布（转化为概率单位） . . . 58
9.1.4 拟合线性回归模型，计算模型估计值 . . . 59
9.1.5 进行分档排序 . . . 59
9.2 案例：孕产妇保健评价 . . . 59
9.2.1 准备数据 . . . 59
9.2.2 秩和比法评价 . . . 60
9.2.3 重现经典案例结果 . . . 60
9.2.4 更一般做法 . . . 62

第十章 模糊综合评价 64
10.1 模糊理论 . . . 64
10.1.1 模糊集与隶属度 . . . 64
10.1.2 隶属函数 . . . 65
10.1.3 `compute_mf()` 函数 . . . 73
10.1.4 模糊运算 . . . 74
10.2 模糊综合评价算法步骤 . . . 77
10.3 案例：耕作方案的模糊综合评价 . . . 79
10.3.1 准备评价对象数据（表 2） . . . 80
10.3.2 准备阈值数据（从表 1 提炼，用于计算隶属向量） . . . 80
10.3.3 批量模糊综合评价 . . . 81

第十一章 数据包络分析 84
11.1 DEA 相关概念 . . . 85
11.2 常用 DEA 模型 . . . 86
11.2.1 CCR 模型（规模收益不变假设下的径向 DEA 模型） . . . 86
11.2.2 BCC 模型（规模收益可变假设下的径向 DEA 模型） . . . 87
11.2.3 带非期望产出的 SBM 模型 . . . 90

11.2.4 超效率模型 91
11.2.5 Malmquist 指数 92
11.3 DEA 案例 95
11.4 DEA-malmquist 案例 98

第十二章 不平等度量 **100**
12.1 个体数据与平均数据 100
12.2 基尼系数 101
12.2.1 洛伦兹曲线与基尼系数 101
12.2.2 基尼系数计算方法 102
12.2.3 案例：批量计算基尼系数 102
12.2.4 绘制洛伦兹曲线 105
12.3 泰尔指数 107
12.3.1 总体泰尔指数 107
12.3.2 单分组变量的泰尔指数及其分解 107
12.3.3 两分组变量的泰尔指数及其分解 108
12.3.4 案例：批量计算泰尔指数及其分解 110

第十三章 系统评价 **117**
13.1 耦合协调度 117
13.1.1 耦合度 117
13.1.2 协调指数 118
13.1.3 耦合协调度 118
13.2 障碍度 119
13.3 案例：耦合协调度、障碍度 120
13.3.1 分别综合评价每个子系统 121
13.3.2 批量计算耦合协调度 122
13.3.3 批量计算障碍度 122

参考文献 **125**

图索引

1 配套书：《数学建模：算法与编程实现》 3
2 R 编程入门书：《R 语言编程：基于 tidyverse》 4

2.1 AHP 层次结构示意图 14
2.2 旅游地选择的层次结构图 17
2.3 方案层对准则层的各因素的贡献度结构图 18

10.1 严格周末与模糊周末 65
10.2 三角隶属函数 70
10.3 梯形隶属函数 71
10.4 高斯隶属函数 71
10.5 布尔运算示意图 75
10.6 布尔运算换个写法 75
10.7 从布尔运算到模糊运算 76

11.1 CRS 假设下的生产可能集 88
11.2 VRS 假设下生产可能集 89

12.1 洛伦兹曲线与基尼系数 101

表索引

2.1 Saaty 发明的 1-9 标度含义 15
2.2 不同阶数矩阵的随机一致性指标 16

10.2 耕作方案评价指标数据的等级划分 79
10.3 三种耕作方案的指标数据 79

前言

这是与 `mathmodels` 包配套的中文使用手册[1]，内容整合自该包的 vignettes，目前只完整包含**评价类算法**。

`mathmodels` 包与本配套使用手册的核心特色

1. 细节考究，算法严谨

- **全面覆盖主流算法**：涵盖综合评价、数据包络分析、不平等度量、灰色预测、模糊评价等 13 大类核心算法
- **算法实现精益求精**：每个函数都经过精心设计，充分考虑数据预处理、指标方向性、权重计算等易忽略的细节
- **理论与实践并重**：所有算法均基于《数学建模：算法与编程实现》教材，理论扎实，应用可靠

2. 接口优雅，极致易用

- **统一的函数设计哲学**：采用一致的参数命名和接口规范，学习成本极低
- **智能内置处理**：自动处理数据标准化、归一化、正向化等预处理步骤，让用户专注核心分析
- **丰富的可视化支持**：提供隶属函数可视化、结果排序展示等直观功能

3. tidyverse 生态无缝集成

- **数据管道式编程**：完美支持 `|>` 管道操作，代码简洁流畅
- **批量处理能力强大**：结合 `mutate()`、`across()` 等 tidyverse 函数，轻松实现批量算法应用
- **现代化数据操作**：与 tidyverse 生态系统深度融合，提供符合 R 用户习惯的操作体验

4. 教学科研双重价值

- **完整配套资源**：包内 vignettes 与在线书籍形成完整学习体系
- **案例驱动学习**：丰富的真实案例帮助用户快速掌握算法应用
- **学术研究利器**：为研究人员提供可靠、高效的算法工具箱

[1] 使用黄湘云提供的 `quarto` 中文书籍模板编写.

5. 创新算法组合

- **主客观赋权融合**：提供线性组合、乘法综合、博弈均衡等多种权重合成方法
- **多层次分解分析**：支持泰尔指数多维度分解，深入挖掘数据内在规律
- **扩展性强**：模块化设计便于功能扩展和定制开发

这套完整的解决方案将复杂的数学建模算法转化为简单易用的 R 工具，让研究者和实践者能够专注于解决实际问题，而非算法实现的细节。

关于 mathmodels 包（当前 0.0.5 版本）

mathmodels 包是一个功能丰富的 R 语言数学建模包，专为《数学建模：算法与编程实现》[1]（机械工业出版社）配套开发。该包涵盖了微分方程与差分方程、统计分析、优化、评价和预测等多个领域的算法实现。当前版本重点关注评价算法模块，主要包括：指标数据预处理（标准化、归一化等）、主客观赋权方法（层次分析法、熵权法、CRITIC 法、主成分法）及主客观组合赋权、综合评价方法（TOPSIS 法、模糊综合评价法、秩和比法、数据包络分析（DEA, SBM, Malmquist）、不平等测度（基尼系数、泰尔指数）、系统评价（耦合协调度、障碍度）以及灰色预测模型（GM(1,1)、GM(1,N)、Verhulst 模型）等。本包面向数学建模领域的研究人员和实践者而设计，提供便捷高效的算法工具支持。

可从 Github 安装：

```
remotes::install_github("zhjx19/mathmodels")
```

或下载到本地解压到当前路径安装（需先安装依赖包：deaR 包）：

```
install.packages("mathmodels-master", repos=NULL, type="source")
```

图 1: 配套书：《数学建模：算法与编程实现》

图 2: R 编程入门书：《R 语言编程：基于 tidyverse》

更多学习资源

欢迎关注我的知乎，公众号：R 语言与数学建模，B 站：张敬信老师

欢迎加入我的个人知识库，里面有所有整理好的我出品的基于 tidyverse/mlr3verse 的全网最新的 R 语言编程、R 机器学习、数学建模相关学习、案例资料，可以下载，可以 DeepSeek 问答：

- **ima 知识库**：数学建模，tidy-R 语言，R 机器学习，大学数学学习
- **知乎直答知识库**：数学建模，tidy-R 语言，R 机器学习，大学数学学习

第一部分

评价类算法

评价类算法包含 13 个章节：

- 指标数据预处理
- 层次分析法
- 熵权法
- CRITIC 法
- 主成分赋权法
- 主客观组合赋权
- TOPSIS
- 灰色关联分析
- 秩和比法
- 模糊综合评价
- 数据包络分析
- 不平等度量
- 系统评价

第一章　指标数据预处理

多指标综合评价中，指标数据的预处理是确保评价结果科学、合理和具有可比性的关键。由于原始数据常存在量纲不同、数量级差异大以及指标方向不一致等问题，直接计算会导致量纲大的指标主导结果，或方向相反的指标相互抵消。因此，通常采用无量纲化、标准化、归一化等方法消除量纲影响；对负向指标进行正向化处理；必要时还可使用对数变换减少极端值的影响。

这些预处理使得不同方向、量纲和数量级的指标能够在同一方向、统一尺度上进行比较与合成，从而更准确地体现各自的实际贡献，为后续的权重赋值和综合评价提供可靠基础。

假设有 n 个评价对象，m 个评价指标，形成原始指标数据矩阵 $\mathbf{X}$，其中，x_{ij} 表示第 i 个评价对象第 j 个指标的值。

R 语言实现的话，指标数据 $\mathbf{X}$ 通常用数据框存放，一个指标的数据是 $\mathbf{X}$ 的 1 列（向量 $\mathbf{x}$）。所以，指标数据预处理，就是用 `mutate()` 修改列，若同时对多列做某一种预处理，可以结合 `across()`。

以下所有预处理函数，都设计为接受一个向量 `x`，返回同样长度的结果向量。

1.1　方向一致性处理

1.1.1　负向指标正向化

正向指标，值越大越好，如 GDP；负向指标，值越小越好，如失业率。

正向指标不用处理，负向指标可用以下方式转化为正向指标：

- 倒数变换：$\mathbf{x}' = 1/\mathbf{x}$
- 极小极大变换：$\mathbf{x}' = \max \mathbf{x} - \mathbf{x}$

注：负向指标正向化，也可以在做归一化/标准化时同时完成（见下文）。

`mathmodels` 包提供了 `to_positive()` 函数实现负向指标正向化，基本语法为：

```
to_positive(x, type = "minmax")
```

- `type` 指定处理方法：`"minmax"`（极小极大变换，默认）、`"reciprocal"`（倒数变换）。

1.1.2 居中型指标正向化

居中型指标，是指值越接近某个中间值 x_{best} 越好，如 PH 值，通常按如下公式变换：

$$\mathbf{x}' = 1 - \frac{|\mathbf{x} - x_{best}|}{\max\{|\mathbf{x} - x_{best}|\}}$$

`mathmodels` 包提供了 `rescale_middle()` 函数实现居中型指标正向化，基本语法为：

```
rescale_middle(x, m)
```

- `m` 为最佳的中间值。

1.1.3 区间型指标正向化

区间型指标，是指值在某个确定的区间范围 $[a, b]$ 内为最好，如体温，通常做如下变换：

$$\mathbf{x}' = \begin{cases} 1 - \dfrac{a - \mathbf{x}}{M}, & \mathbf{x} < a \\ 1, & a \le \mathbf{x} \le b \\ 1 - \dfrac{\mathbf{x} - b}{M}, & \mathbf{x} > b \end{cases}$$

其中，$M = \max\{a - \min \mathbf{x}, \max \mathbf{x} - b\}$。

`mathmodels` 包提供了 `rescale_interval()` 函数实现区间型指标正向化，基本语法为：

```
rescale_interval(x, a, b)
```

- `a`，`b` 为最佳区间的左右端点值

注意，上述居中型、区间型指标正向化已含归一化，结果落在 $[0, 1]$，不必再做归一化。

1.2 无量纲化处理

各指标通常具有不同的量纲（单位）和数量级，若直接进行综合计算，容易使量纲或数值范围较大的指标主导结果，而量纲小或范围窄的指标被弱化，影响评价的客观性与准确性。因此，需对原始数据进行无量纲化处理，消除单位和量级差异，使得指标能够在统一尺度上进行比较与合成。

1.2.1 标准化

标准化，也称为 Z 标准化，将数据变成均值为 0，标准差为 1：

$$\mathbf{x}' = \frac{\mathbf{x} - \mu}{\sigma}$$

其中，μ 和 σ 分别为 $\mathbf{x}$ 的均值和标准差。

若是负向指标，可以标准化后再乘以 -1 进行正向化处理。

注：若只减去均值，不除以标准差，叫做**中心化**。

`mathmodels` 包提供了 `standardize()` 函数实现标准化，基本语法为：

```
standardize(x, center = TRUE, scale = TRUE)
```

该函数是将自带的 `scale()` 函数封装为只处理向量，默认是做标准化，设置 `scale = FALSE` 则做中心化。

1.2.2 归一化

归一化，是指对数据进行线性变换，等比例缩放，并平移到目标区间 $[a, b]$（通常取 $[0, 1]$），保持原始数据的相对大小关系不变。

对于正向指标，

$$\mathbf{x}' = (b - a) \cdot \frac{\mathbf{x} - \min \mathbf{x}}{\max \mathbf{x} - \min \mathbf{x}} + a$$

对于负向指标，

$$\mathbf{x}' = (b - a) \cdot \frac{\max \mathbf{x} - \mathbf{x}}{\max \mathbf{x} - \min \mathbf{x}} + a$$

`mathmodels` 包提供了 `rescale()` 函数实现归一化，基本语法为：

```
rescale(x, type = "+", a = 0, b = 1)
```

- `type` 指定指标方向
- `a`，`b` 确定目标区间，默认是 0 和 1

注：标准化更适合应用到近似服从正态分布的数据；归一化更适合应用到近似服从均匀分布的数据。

1.2.3 规范化

规范化，是指将向量除以其范数，变成单位长度（即长度为 1）：

$$\mathbf{x}' = \frac{\mathbf{x}}{\|\mathbf{x}\|_2}$$

这里使用 L_2 范数（欧几里德范数）：$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$。

规范化后，向量在欧氏空间中的长度为 1，以便用于 TOPSIS 法。

`mathmodels` 包提供了 `normalize()` 函数实现 L_2 规范化：

```
normalize(x)
```

1.2.4 用于 GRA 的无量纲化

灰色关联分析（GRA）中，原始数据通常被视为具有时间序列特性的指标。通常优先使用初值化法，因其能保留序列的增长特性，与灰色系统理论强调的“少数据建模”特点契合。

初值化：适合分析增长趋势（如经济指标、产量变化）

$$\mathbf{x}' = \frac{\mathbf{x}}{\mathbf{x}[1]},\ \text{（正向指标）} \quad \mathbf{x}' = \frac{\mathbf{x}[1]}{\mathbf{x}},\ \text{（负向指标）}$$

还有另外两种方法：

均值化：适合稳定波动的数据（如温度、pH 值）

$$\mathbf{x}' = \frac{\mathbf{x}}{\text{mean}(\mathbf{x})}$$

最大值化：适合极差较大的数据

$$\mathbf{x}' = \frac{\mathbf{x}}{\max(\mathbf{x})},\ \text{（正向指标）} \quad \mathbf{x}' = \frac{\min(\mathbf{x})}{\mathbf{x}},\ \text{（负向指标）}$$

`mathmodels` 包提供了如下三个函数实现这三种无量纲化方法：

```
rescale_initial(x, type = "+")
rescale_mean(x)
rescale_extreme(x, type = "+")
```

- `type` 设置指标方向。

1.3 综合案例

加载包：

```
library(tidyverse)
library(mathmodels)
```

20 条河流的水质数据，已内置到 `mathmodels` 包，直接访问：

```
water_quality
```

```
# A tibble: 20 x 5
      ID    O2    PH  germ nutrient
   <dbl> <dbl> <dbl> <dbl>    <dbl>
 1     1  4.69  6.59    51    11.9
 2     2  2.03  7.86    19     6.46
 3     3  9.11  6.31    46     8.91
 4     4  8.61  7.05    46    26.4
 5     5  7.13  6.5     50    23.6
 6     6  2.39  6.77    38    24.6
 7     7  7.69  6.79    38     6.01
 8     8  9.3   6.81    27    31.6
 9     9  5.45  7.62     5    18.5
```

```
10    10  6.19  7.27    17     7.51
11    11  7.93  7.53     9     6.52
12    12  4.4   7.28    17    25.3
13    13  7.46  8.24    23    14.4
14    14  2.01  5.55    47    26.3
15    15  2.04  6.4     23    17.9
16    16  7.73  6.14    52    15.7
17    17  6.35  7.58    25    29.5
18    18  8.29  8.41    39    12.0
19    19  3.54  7.27    54     3.16
20    20  7.44  6.26     8    28.4
```

指标包括：含氧量越高越好（正向指标）；PH 值越接近 7 越好（居中型指标）；细菌总数越少越好（负向指标）；植物性营养物量介于 10 ~ 20 之间最佳（区间型指标）。

做如下预处理：

- 居中型、区间型指标做正向化（归一化）处理
- 负向指标做极小极大正向化
- （两个）正向指标做归一化

前文说了，指标数据预处理就是 `mutate()` 修改列问题。

```
water_quality |>
  mutate(PH = rescale_middle(PH, 7),
         nutrient = rescale_interval(nutrient, 10, 20),
         germ = to_positive(germ),
         across(c(O2, germ), rescale))
```

```
# A tibble: 20 x 5
      ID      O2     PH   germ nutrient
   <dbl>   <dbl>  <dbl>  <dbl>    <dbl>
 1     1 0.368   0.717  0.0612    1
 2     2 0.00274 0.407  0.714     0.694
 3     3 0.974   0.524  0.163     0.906
 4     4 0.905   0.966  0.163     0.444
 5     5 0.702   0.655  0.0816    0.691
 6     6 0.0521  0.841  0.327     0.601
 7     7 0.779   0.855  0.327     0.655
 8     8 1       0.869  0.551     0
 9     9 0.472   0.572  1         1
10    10 0.573   0.814  0.755     0.785
11    11 0.812   0.634  0.918     0.699
```

12	12	0.328	0.807	0.755	0.542
13	13	0.748	0.145	0.633	1
14	14	0	0	0.143	0.455
15	15	0.00412	0.586	0.633	1
16	16	0.785	0.407	0.0408	1
17	17	0.595	0.6	0.592	0.182
18	18	0.861	0.0276	0.306	1
19	19	0.210	0.814	0	0.409
20	20	0.745	0.490	0.939	0.273

第二章　层次分析法

层次分析法（AHP），是一种将复杂决策问题结构化并进行定量分析的系统方法，由美国运筹学家 T. L. Saaty 于 20 世纪 70 年代提出。该方法通过构建层次结构模型，并对主观判断进行量化处理，从而有效解决多目标、多准则下的复杂决策问题，尤其适用于定性因素与定量因素交织的实际场景。

AHP 的核心优势在于能够合理地结合定性判断与定量分析，依据人类思维和心理规律，将决策过程层次化、数量化，使原本模糊的判断变得清晰可操作。因此，它被广泛应用于各类复杂的决策分析中。由于其决策依据主要来源于计算所得的权重，AHP 也常被用于确定评价指标的相对重要性权重。

与传统依赖大量定量数据的方法不同，AHP 只需通过对指标之间的定性比较，即可构造判断矩阵并进一步合成得出各指标的权重，也因此是一种主观赋权法。在实际应用中，建议采用问卷形式，邀请多位专家对各项指标进行两两比较或直接打分，从而构建出更具代表性和科学性的判断矩阵。

2.1　算法步骤

2.1.1　建立层次结构

在深入分析实际问题的基础上，将有关的各个因素按照不同属性自上而下地分解成若干层次，同一层的诸因素从属于上一层的因素或对上层因素有影响，同时又支配下一层的因素或受到下层因素的作用。

最上层为目标层，通常只有一个因素，最下层通常为方案或对象层，中间可以有一个或多个层次，通常为准则或指标层。准则层一般 $5 \sim 7$ 个因素为好，当准则过多时（比如多于 9 个）应进一步分解出子准则层。

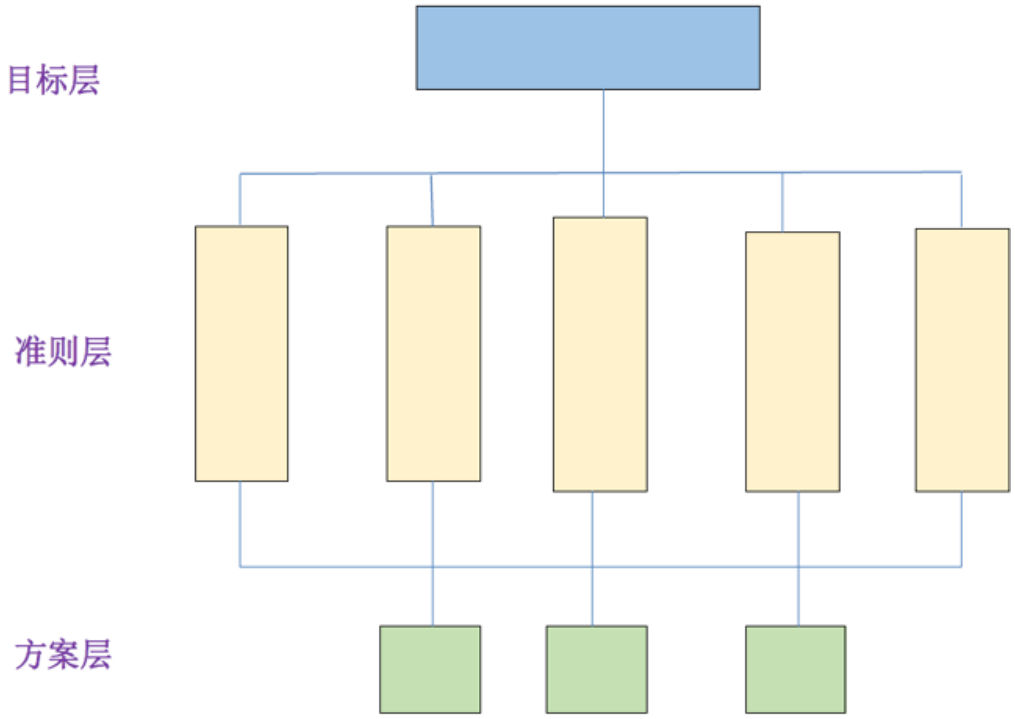


图 2.1: AHP 层次结构示意图

2.1.2 构造判断矩阵

构造判断矩阵，这一步是要比较层次结构模型的第二层各个因素对上一层因素影响，从而确定它们对上层因素的影响作用中所占的权重。有时需要理解为下层因素对上层因素的贡献。

设有 n 个因素 $x_1, x_2, \cdots, x_n$ 对上一层目标有影响，直接确定它们对目标的影响程度不是很容易，所以每次取两个因素 x_i 与 x_j 比较，用 a_{ij} 表示 x_i 与 x_j 对上层目标的影响之比，则 $A = (a_{ij})_{m\times n}$ 称为**判断矩阵**或成对比较矩阵。

$$A = \begin{bmatrix} \frac{x_1}{x_1} & \frac{x_1}{x_2} & \cdots & \frac{x_1}{x_n} \\ \frac{x_2}{x_1} & \frac{x_2}{x_2} & \cdots & \frac{x_2}{x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{x_n}{x_1} & \frac{x_n}{x_2} & \cdots & \frac{x_n}{x_n} \end{bmatrix}$$

判断矩阵主对角线都为 1，关于主对角线对称的元素互为倒数，所以是正互反矩阵。

Saaty 根据绝大多数人认知事物的心理习惯，建议用 $1 \sim 9$ 及其倒数作为标度来确定 a_{ij} 的值。

表 2.1: Saaty 发明的 1-9 标度含义

i 比 j 强的重要程度	相等	稍强	强	很强	绝对强
a_{ij}	1	3	5	7	9

其中，2, 4, 6, 8 分别介于 1, 3, 5, 7, 9 对应的重要程度之间。

2.1.3 计算权向量及一致性检验

对于每一个判断矩阵，计算其最大特征根 λ_{max} 及对应特征向量。

再进行**一致性检验**：利用一致性指标、平均随机一致性指标计算一致性比率。若检验通过，那么归一化特征向量即为权向量；若不通过，需重新构造判断矩阵.

判断矩阵涉及到的一个关键问题：**一致性**，这涉及到两两比较的传递性。比如 a 的重要性是 b 的 2 倍，b 的重要性是 c 的 3 倍，则传递过来的 a 的重要性得是 c 的 6 倍，而你对 a 和 c 两两比较的重要性不一定是 6 倍，这就是不一致。

由于判断矩阵构造过程只是在做两两比较，看不到这些传递过来的关系，故不可能做到完全的一致性，所以需要对判断矩阵进行一致性检验：即保证一致性的偏差不能太大。

定义 1 若正互反矩阵 A 满足：

$$a_{ij}a_{jk} = a_{ik}, \quad i, j, k = 1, \cdots, n$$

则称 A 为一致矩阵。

n 阶一致阵具有如下性质：

i) 一致阵的唯一非零特征根为 n

ii) 一致阵的任一列（行）向量都是对应于特征根 n 的特征向量

因此，判别一个 n 阶矩阵 A 是否为一致矩阵，只要计算 A 的最大特征根 λ_{max} 即可。如果 A 不是一致矩阵，则可以证明 $\lambda_{max} > n$，并且 λ_{max} 越大，不一致程度越严重。

定义 2 设 A 为 n 阶判断矩阵，定义一致性指标为：

$$CI = \frac{\lambda_{max} - n}{n - 1}$$

当 $CI = 0$ 时为一致矩阵；CI 越大，A 的不一致程度越严重。

前面说了要保证判断矩阵一致性的偏差不能太大，那就需要有个基准，为此，Saaty 采用随机模拟取平均的方法，得到了各阶判断矩阵的一致性的基准：一致性指标 RI。

表 2.2: 不同阶数矩阵的随机一致性指标

矩阵阶数	3	4	5	6	7	8	9	10	11	12	13
RI	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.54	1.56

有了 RI，只要一致性指标偏离它的相对偏差不超过一定程度，就可以认为是满足一致性要求。于是，

定义 3 定义一致性比率为：

$$CR = \frac{CI}{RI}$$

规定当 $CR < 0.1$ 时，A 的不一致程度在容许范围内，可用其归一化的特征向量作为权重向量，否则需要重新调整判断矩阵 A。

2.1.4 计算组合权向量与组合一致性检验

为了实现层次分析法的最终目的，需要从上而下逐层进行各层元素对目标层合成权重的计算。

对应单个层次结构的单个判断矩阵必须要满足一致性要求。同样地，各层元素对目标层的合成权重向量是否可以接受，就需要进行综合一致性检验。

注：实际应用中，通常不做整体一致性检验，只保证每个层次结构单独满足一致性检验即可。

2.2 案例：旅游地选择

问题描述：某人要出去旅游，有 3 个备选旅游地供参考，他准备从景色、费用、居住条件、饮食、旅途共 5 个因素来考量，最终选择最优的旅游地。

2.2.1 建立层次结构

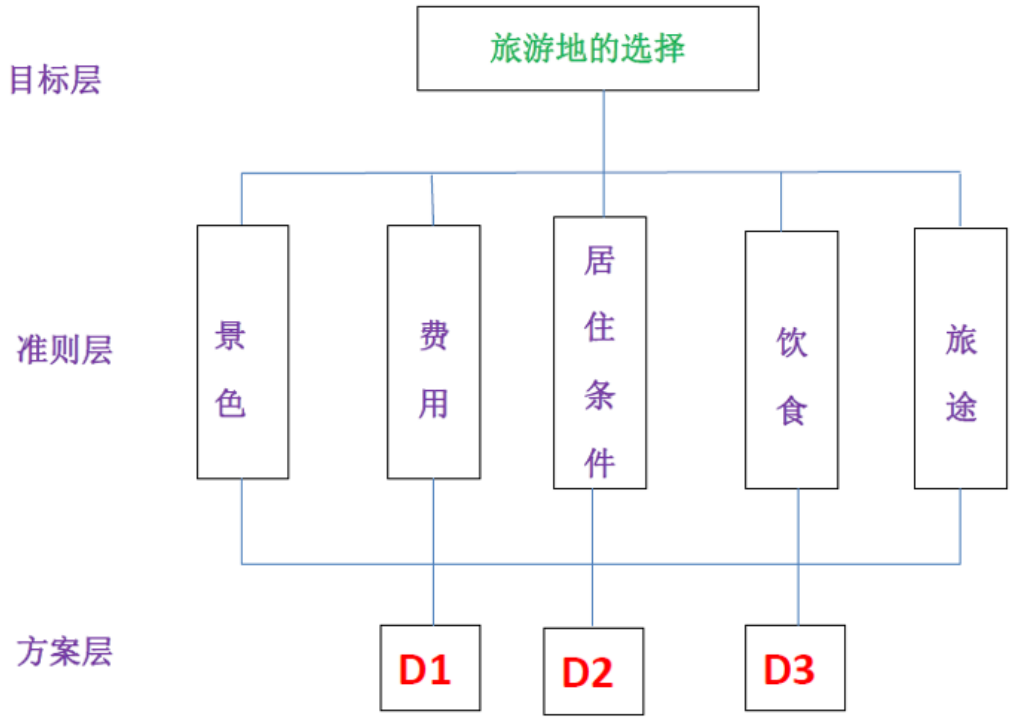


图 2.2: 旅游地选择的层次结构图

2.2.2 确定判断矩阵

1. 准则层对目标层的判断矩阵

准则层 5 个因素景色、费用、居住条件、饮食、交通，对目标层旅游地选择的相对重要度，构成 1 个子结构，需要 1 个判断矩阵：

$$\begin{bmatrix} 1 & 1/2 & 4 & 3 & 3 \\ 2 & 1 & 7 & 5 & 5 \\ 1/4 & 1/7 & 1 & 1/2 & 1/3 \\ 1/3 & 1/5 & 2 & 1 & 1 \\ 1/3 & 1/5 & 3 & 1 & 1 \end{bmatrix}$$

其中，a_{ij} 表示 x_i 与 x_j 对选择旅游地的相对重要性之比。比如，$a_{21} = 2$ 表示在该人看来，费用 x_2 是景色 x_1 的 2 倍重要。

2. 方案层对准则层的判断矩阵

方案层（3 个旅游地）对准则层（5 个因素），构成了 5 个单独的子结构，需要 5 个判断矩阵。此时可以理解为 3 个旅游地分别对每个因素的贡献度（重要度）。

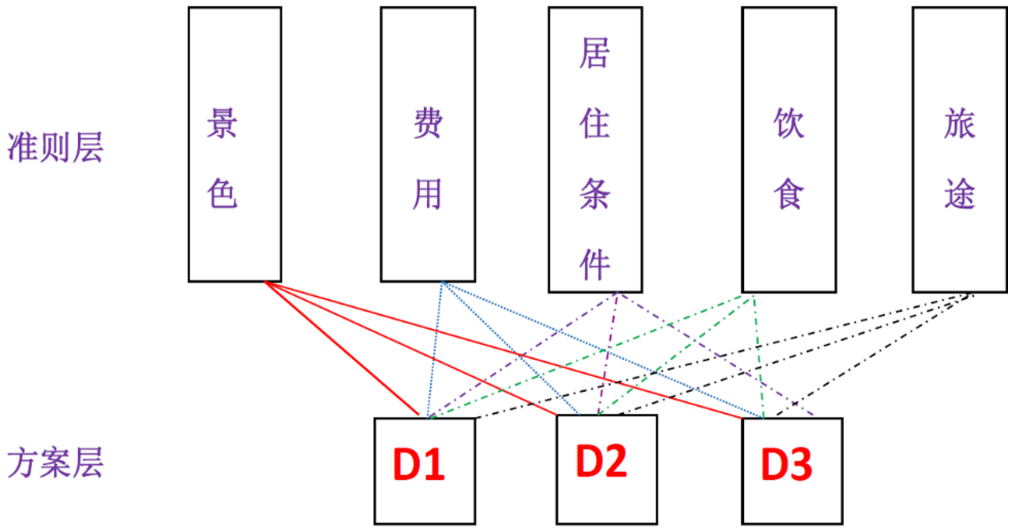


图 2.3: 方案层对准则层的各因素的贡献度结构图

$$B_1=\begin{bmatrix}1 & 2 & 5\\ 1/2 & 1 & 2\\ 1/5 & 1/2 & 1\end{bmatrix},\quad B_2=\begin{bmatrix}1 & 1/3 & 1/8\\ 3 & 1 & 1/3\\ 8 & 3 & 1\end{bmatrix}$$

$$B_3=\begin{bmatrix}1 & 1 & 3\\ 1 & 1 & 3\\ 1/3 & 1/3 & 1\end{bmatrix},\quad B_4=\begin{bmatrix}1 & 3 & 4\\ 1/3 & 1 & 1\\ 1/4 & 1 & 1\end{bmatrix},\quad B_5=\begin{bmatrix}1 & 1 & 1/4\\ 1 & 1 & 1/4\\ 4 & 4 & 1\end{bmatrix}$$

2.2.3 基于 mathmodels 包求解

加载包：

```
library(tidyverse)
library(mathmodels)
```

mathmodels 包提供了 AHP() 函数实现一个子结构的层次分析法，只需要提供判断矩阵，就可以计算权重、一致性比率、最大特征值、一致性指标。

本例是层次分析法典型的嵌套层次结构（三个层级，两层判断矩阵），需要对判断矩阵逐层计算，再向上合成。

(1) 第 2 层（下层）

- 准备判断矩阵：

```
B1 = matrix(c(1,  2, 5,
              1/2,1, 2,
              1/5,1/2,1), nrow = 3, byrow = TRUE)
```

```
B2 = matrix(c(1, 1/3, 1/8,
              3, 1, 1/3,
              8, 3, 1), nrow = 3, byrow = TRUE)
B3 = matrix(c(1, 1, 3,
              1, 1, 3,
              1/3,1/3,1), nrow = 3, byrow = TRUE)
B4 = matrix(c(1, 3, 4,
              1/3,1, 1,
              1/4,1, 1), nrow = 3, byrow = TRUE)
B5 = matrix(c(1, 1, 1/4,
              1, 1, 1/4,
              4, 4, 1), nrow = 3, byrow = TRUE)
```

- 批量层次分析法：

```
res2 = list(B1,B2,B3,B4,B5) |>
  map(AHP)
```

- 提取 5 个一致性比率：

```
map_dbl(res2, "CR")
```

```
[1]  4.771648e-03  1.328987e-03 -7.656711e-16  7.933373e-03  0.000000e+00
```

- 提取 5 组权重，分别是方案层对准则层每个因素的重要度（贡献度）。按列合并，转化为矩阵：

```
W2 = map_dfc(res2, "w") |> as.matrix()
W2
```

```
          ...1       ...2      ...3      ...4      ...5
[1,] 0.5953790 0.08193475 0.4285714 0.6337079 0.1666667
[2,] 0.2763505 0.23634070 0.4285714 0.1919206 0.1666667
[3,] 0.1282705 0.68172455 0.1428571 0.1743715 0.6666667
```

该权重矩阵，行代表方案，列代表父级因素，将用于向上合成时的左乘矩阵。注意，若某方案不受某个父级因素支配（无连线），相应位置补 0。

(2) 第 1 层（上层）

```
A = matrix(c(1,   1/2, 4, 3,   3,
             2,   1,   7, 5,   5,
             1/4, 1/7, 1, 1/2, 1/3,
             1/3, 1/5, 2, 1,   1,
             1/3, 1/5, 3, 1,   1),
```

```
           byrow = TRUE, nrow = 5)
res = AHP(A)
```

- 提取一致性比率：

```
res$CR
```

```
[1] 0.01609027
```

- 提取权重向量：

```
W1 = res$w
W1
```

```
[1] 0.26360349 0.47583538 0.05381460 0.09806829 0.10867824
```

(3) 向上合成

- 即做矩阵乘法：

```
W2 %*% W1 |> as.vector()
```

```
[1] 0.2992545 0.2453040 0.4554415
```

注：若有更多的层级，继续将新的权重矩阵左乘即可，比如 `W3 %*% W2 %*% W1`。

该权重向量就是方案层（3 个旅游地）对目标层（选择旅游地）的权重。其中，第 3 个权重最大为 0.4554，所以应该选择第 3 个旅游地，作为最佳旅游地。

第三章　熵权法

在信息论中，**熵**是对不确定性的一种度量。不确定性越大，熵值越高，所包含的信息量也就越大；反之，不确定性越小，熵值越低，信息量也相应减少。

根据熵的这一特性，可以通过计算熵值来评估一个事件的随机性与无序程度。同时，熵值也可用于衡量某个指标的离散程度。指标的离散程度越大，其在综合评价体系中的影响（即权重）也就越大。例如，如果某一指标在所有样本中的取值完全相同，则该指标对综合评价没有任何区分作用，其提供的信息量为零，因此对应的权重也应为零。

熵权法，是一种基于数据本身分布特征的客观赋权方法，其核心思想是依据各指标的信息熵来确定权重，从而避免主观因素的干扰，具有较强的科学性与合理性。

3.1　算法步骤

设有 n 个评价对象，m 个评价指标，形成原始指标数据矩阵，其中 x_{ij} 表示第 i 个评价对象第 j 个指标的值。

(1) 数据归一化

由于不同指标通常具有不同的量纲和数量级，为了消除其影响，需要进行归一化处理。

- 对于正向指标（越大越好）：

$$x'_{ij} = \frac{x_{ij} - \min\limits_{i} x_{ij}}{\max\limits_{i} x_{ij} - \min\limits_{i} x_{ij}}$$

- 对于负向指标（越小越好）：

$$x'_{ij} = \frac{\max\limits_{i} x_{ij} - x_{ij}}{\max\limits_{i} x_{ij} - \min\limits_{i} x_{ij}}$$

  其中，$\min_i x_{ij}$ 和 $\max_i x_{ij}$ 分别是第 j 个指标所有样本值中的最小值和最大值。

注意，熵权法中涉及 $\ln(\cdot)$ 运算，所以归一化不能出现 0 和 1，为此，归一化时把 $[0,1]$ 稍微收缩一下，比如到 $[0.002, 0.998]$。

为了方便起见，归一化后的数据 x'_{ij} 仍记为 x_{ij}。

注：归一化更适合应用到近似服从均匀分布的数据；也可以使用标准化，标准化更适合应用到近似服从正态分布的数据，负向指标标准化后乘以 -1 即可正向化。

(2) 计算第 j 个指标下第 i 个样本值占该指标的比重

将标准化后的值 x_{ij} 视为某种" 概率贡献"，计算其在第 j 个指标总 "贡献" 中的比重：

$$p_{ij} = \frac{x_{ij}}{\sum_{i=1}^{n} x_{ij}}, \quad i = 1, \cdots, n,\ j = 1, \cdots, m$$

(3) 计算第 j 个指标的熵值

利用信息熵公式计算每个指标的信息熵：

$$e_j = -k \sum_{i=1}^{n} p_{ij} \ln(p_{ij}), \quad j = 1, \cdots, m$$

其中，$k = 1/\ln(n) > 0$，满足 $e_j \geq 0$。

熵值 e_j 反映了第 j 个指标下数据的离散程度，熵值 e_j 越大，表明该指标提供的有用信息量越少。

(4) 计算信息熵冗余度（差异系数）

$$d_j = 1 - e_j, \quad j = 1, \cdots, m$$

差异系数反映了指标信息量的大小，d_j 越大，表示第 j 个指标值的离散程度越大（不确定性越小，信息量越大），该指标在综合评价中应赋予较大的权重。

(5) 计算各个指标的权重

将差异系数进行归一化处理，得到各指标的最终权重：

$$w_j = \frac{d_j}{\sum_{j=1}^{m} d_j}, \quad j = 1, \cdots, m$$

(6)（可选）计算各个评价对象的综合得分

$$s_i = 100» \sum_{j=1}^{m} w_j x_{ij}, \quad i = 1, \cdots, n$$

其中，x_{ij} 为标准化后的数据。

3.2 案例：河流水质数据

加载包：

```
library(tidyverse)
library(mathmodels)
```

3.2.1 准备数据

使用内置的 water_quality 数据来演示：

```
water_quality
```

```
# A tibble: 20 x 5
      ID    O2    PH  germ nutrient
   <dbl> <dbl> <dbl> <dbl>    <dbl>
 1     1  4.69  6.59    51    11.9
 2     2  2.03  7.86    19     6.46
 3     3  9.11  6.31    46     8.91
 4     4  8.61  7.05    46    26.4
 5     5  7.13  6.5     50    23.6
 6     6  2.39  6.77    38    24.6
 7     7  7.69  6.79    38     6.01
 8     8  9.3   6.81    27    31.6
 9     9  5.45  7.62     5    18.5
10    10  6.19  7.27    17     7.51
11    11  7.93  7.53     9     6.52
12    12  4.4   7.28    17    25.3
13    13  7.46  8.24    23    14.4
14    14  2.01  5.55    47    26.3
15    15  2.04  6.4     23    17.9
16    16  7.73  6.14    52    15.7
17    17  6.35  7.58    25    29.5
18    18  8.29  8.41    39    12.0
19    19  3.54  7.27    54     3.16
20    20  7.44  6.26     8    28.4
```

这是 20 条河流的水质数据，含氧量越高越好（正向指标）；PH 值越接近 7 越好（居中型指标）；细菌总数越少越好（负向指标）；植物性营养物量介于 $10 \sim 20$ 之间最佳（区间型指标）。

3.2.2 数据预处理

借助 mathmodels 包提供的预处理函数，使用 mutate() 修改列即可。只需要处理居中型和区间型指标，因为指标归一化已内置到熵权法函数。

```
df = water_quality |>
  mutate(PH = rescale_middle(PH, 7),
         nutrient = rescale_interval(nutrient, 10, 20))
df
```

```
# A tibble: 20 x 5
      ID    O2     PH  germ nutrient
   <dbl> <dbl>  <dbl> <dbl>    <dbl>
 1     1  4.69 0.717     51    1
 2     2  2.03 0.407     19    0.694
 3     3  9.11 0.524     46    0.906
 4     4  8.61 0.966     46    0.444
 5     5  7.13 0.655     50    0.691
 6     6  2.39 0.841     38    0.601
 7     7  7.69 0.855     38    0.655
 8     8  9.3  0.869     27    0
 9     9  5.45 0.572      5    1
10    10  6.19 0.814     17    0.785
11    11  7.93 0.634      9    0.699
12    12  4.4  0.807     17    0.542
13    13  7.46 0.145     23    1
14    14  2.01 0         47    0.455
15    15  2.04 0.586     23    1
16    16  7.73 0.407     52    1
17    17  6.35 0.6       25    0.182
18    18  8.29 0.0276    39    1
19    19  3.54 0.814     54    0.409
20    20  7.44 0.490      8    0.273
```

3.2.3 熵权法

用 entropy_weight() 函数实现用**熵权法**计算权重，该函数已内置归一化处理，基本语法：

```
entropy_weight(X, index = NULL, epsilon = 0.002)
```

- index 可以指定对哪些列做正向（"+"）、负向（"-"）、不做（NA）归一化，默认 index = NULL 表示，所有列都不做归一化。

- epsilon 是熵权法中避免出现 0 和 1 值的微调量，默认为 0.002。

本例已对指标列 PH 和 nutrient 事先做了归一化，故将指标方向设置为 NA，以不再重复做归一化，只是微调 0 和 1 值，避免熵权法计算时出现 $\ln(0)$。

```
idx = c("+", NA, "-", NA)
res = entropy_weight(df[-1], idx)
```

- 查看指标权重：

```
res$w
```

```
       O2        PH      germ  nutrient
0.3153202 0.1813149 0.3506929 0.1526721
```

- 查看评价对象得分：

```
res$s
```

```
 [1] 42.05869 43.14229 59.75555 58.55797 47.47757 37.60200 61.51622 66.57186
 [9] 75.49739 71.25274 79.89397 59.70830 63.57278 12.10016 48.22587 48.81539
[17] 53.16568 53.61734 27.72132 69.36453
```

- 对应到河流，增加排名列：

```
tibble(ID = df$ID, score = res$s, rank = min_rank(-score)) |>
  arrange(rank)
```

```
# A tibble: 20 x 3
      ID score  rank
   <dbl> <dbl> <int>
 1    11  79.9     1
 2     9  75.5     2
 3    10  71.3     3
 4    20  69.4     4
 5     8  66.6     5
 6    13  63.6     6
 7     7  61.5     7
 8     3  59.8     8
 9    12  59.7     9
10     4  58.6    10
11    18  53.6    11
12    17  53.2    12
13    16  48.8    13
14    15  48.2    14
15     5  47.5    15
16     2  43.1    16
```

17	1	42.1	17
18	6	37.6	18
19	19	27.7	19
20	14	12.1	20

第四章　CRITIC 法

CRITIC 法，是一种相较于熵权法和标准离差法更为先进的客观赋权方法。

它基于评价指标的对比强度和指标之间的冲突性来综合确定各指标的权重，不仅考虑了指标自身的变异程度（即数据波动性），还引入了指标间的相关性信息，从而更全面地反映指标在评价体系中的重要性。

与“数值越大越重要”的主观判断不同，CRITIC 法完全依赖于数据本身的客观特征进行科学赋权。

变异性（对比强度），是指同一指标在不同评价方案间的取值差异大小，通常以标准差的形式表示。标准差越大，说明该指标在不同方案间的差异越显著，其权重也相应越高。

冲突性，则通过指标间的相关系数来衡量。若两个指标之间存在较强的正相关关系（相关系数接近 1），说明它们在评价中提供的是相似的信息，彼此之间冲突性小，因此在权重分配上应适当降低；反之，若指标间相关性较弱或呈负相关，则表明其冲突性较强，权重相应提高。

从 CRITIC 法的角度来看，在标准差保持不变的前提下：

- 指标间的冲突性越小，其权重越低；
- 冲突性越大，其权重越高；
- 当两个指标高度正相关时（相关系数接近 1），其冲突性趋近于零，意味着二者在评价中传递的信息重叠度高，其中一个指标即可代表另一个的作用，因而权重会更低。

4.1　算法步骤

设有 n 个评价对象，m 个评价指标，形成原始指标数据矩阵，其中 x_{ij} 表示第 i 个评价对象第 j 个指标的值。

(1) 无量纲化处理

为消除因量纲不同对评价结果的影响，需要对各指标进行无量纲化处理处理。

CRITIC 法一般使用正向或负向归一化处理，不建议使用标准化处理，否则标准差全部都变成数值 1，即所有指标的标准差完全一致，这就导致变异性指标没有意义。

- 对于正向指标（越大越好）：

$$x'_{ij} = \frac{x_{ij} - \min\limits_i x_{ij}}{\max\limits_i x_{ij} - \min\limits_i x_{ij}}$$

- 对于负向指标（越小越好）：

$$x'_{ij} = \frac{\max\limits_i x_{ij} - x_{ij}}{\max\limits_i x_{ij} - \min\limits_i x_{ij}}$$

其中, $\min_i x_{ij}$ 和 $\max_i x_{ij}$ 分别是第 j 个指标所有样本值中的最小值和最大值。

为了方便起见, 归一化后的数据 x'_{ij} 仍记为 x_{ij}。

(2) 指标变异性

在 CRITIC 法中使用标准差来表示各指标的内取值的差异波动情况：

$$S_j = \sqrt{\frac{\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2}{n-1}}$$

标准差越大表示该指标的数值差异越大，越能放映出更多的信息，该指标本身的评价强度也就越强，应该给该指标分配更多的权重。

(3) 指标冲突性

根据相关系数计算：

$$R_j = \sum_{i=1}^{p}(1 - r_{ij})$$

其中，r_{ij} 表示指标 i 和 j 之间的相关系数。

使用相关系数来表示指标间的相关性，与其他指标的相关性越强，则该指标就与其他指标的冲突性越小，反映出相同的信息越多，所能体现的评价内容就越有重复之处，一定程度上也就削弱了该指标的评价强度，应该减少对该指标分配的权重。

(4) 信息量

$$C_j = S_j \times R_j$$

其中，C_j 越大，第 j 个指标在整个评价指标体系中的作用越大，就应该给其分配更多的权重。

(5) 将信息量归一化，得到权重

$$w_j = \frac{C_j}{\sum_{j=1}^{p} C_j}$$

4.2 改进：熵-CRITIC 法与不同相关系数

标准 CRITIC 法中，指标变异性是用标准差刻画的，完全可以借鉴熵权法，换成是基于**熵**的刻画：

计算每个指标的熵，再计算信息量：

$$e_j = -\frac{1}{\ln(n)} \sum_{i=1}^{n} p_{ij} \ln(p_{ij})$$

$$S_j = 1 - e_j$$

其中，$p_{ij} = \frac{x_{ij}}{\sum_{i=1}^{n} x_{ij}}$（归一化后的概率）。熵越小，信息量越大（熵权法逻辑）。

相比熵权法，熵-CRITIC 法的优点：

- 结合信息熵（数据不确定性）和指标冲突性（相关性），比单纯熵权法更稳健。
- 适用于数据分布不均匀但指标间存在相关性的情况

另外，标准 CRITIC 法中，冲突性是用 `Pearson` 相关系数刻画的，也可以根据数据的特点，相应地换成 `Spearman` 和 `Kendall` 相关系数。

4.3 案例：河流水质数据

加载包：

```
library(tidyverse)
library(mathmodels)
```

4.3.1 准备数据

使用内置的 `water_quality` 数据来演示：

```
water_quality
```

```
# A tibble: 20 x 5
      ID    O2    PH  germ nutrient
   <dbl> <dbl> <dbl> <dbl>    <dbl>
 1     1  4.69  6.59    51    11.9
 2     2  2.03  7.86    19     6.46
 3     3  9.11  6.31    46     8.91
 4     4  8.61  7.05    46    26.4
 5     5  7.13  6.5     50    23.6
 6     6  2.39  6.77    38    24.6
 7     7  7.69  6.79    38     6.01
 8     8  9.3   6.81    27    31.6
 9     9  5.45  7.62     5    18.5
```

```
10    10  6.19  7.27    17     7.51
11    11  7.93  7.53     9     6.52
12    12  4.4   7.28    17    25.3
13    13  7.46  8.24    23    14.4
14    14  2.01  5.55    47    26.3
15    15  2.04  6.4     23    17.9
16    16  7.73  6.14    52    15.7
17    17  6.35  7.58    25    29.5
18    18  8.29  8.41    39    12.0
19    19  3.54  7.27    54     3.16
20    20  7.44  6.26     8    28.4
```

这是 20 条河流的水质数据，含氧量越高越好（正向指标）；PH 值越接近 7 越好（居中型指标）；细菌总数越少越好（负向指标）；植物性营养物量介于 10 ~ 20 之间最佳（区间型指标）。

4.3.2 数据预处理

借助 `mathmodels` 包提供的预处理函数，使用 `mutate()` 修改列即可。只需要处理居中型和区间型指标，因为指标归一化已内置到 CRITIC 赋权法函数。

```
df = water_quality |>
  mutate(PH = rescale_middle(PH, 7),
         nutrient = rescale_interval(nutrient, 10, 20))
df
```

```
# A tibble: 20 x 5
      ID    O2     PH  germ nutrient
   <dbl> <dbl>  <dbl> <dbl>    <dbl>
 1     1  4.69 0.717     51    1
 2     2  2.03 0.407     19    0.694
 3     3  9.11 0.524     46    0.906
 4     4  8.61 0.966     46    0.444
 5     5  7.13 0.655     50    0.691
 6     6  2.39 0.841     38    0.601
 7     7  7.69 0.855     38    0.655
 8     8  9.3  0.869     27    0
 9     9  5.45 0.572      5    1
10    10  6.19 0.814     17    0.785
11    11  7.93 0.634      9    0.699
12    12  4.4  0.807     17    0.542
13    13  7.46 0.145     23    1
```

```
14    14  2.01 0         47    0.455
15    15  2.04 0.586     23    1
16    16  7.73 0.407     52    1
17    17  6.35 0.6       25    0.182
18    18  8.29 0.0276    39    1
19    19  3.54 0.814     54    0.409
20    20  7.44 0.490      8    0.273
```

4.3.3 CRITIC 及其改进权重

用 `critic_weight()` 函数实现用 CRITIC 法及其改进计算权重，该函数已内置归一化处理，基本语法：

```
critic_weight(
  X,
  index = NULL,
  method = "std",
  cor_method = "pearson",
  epsilon = 0.002)
```

- `index` 可以指定对哪些列做正向（`"+"`）、负向（`"-"`）、不做（`NA`）归一化，，默认 `index = NULL` 表示，所有列都不做归一化。
- `method` 可以选择计算变异性的方法，是基于标准差（`"std"`）还是熵（`"entropy"`）；
- `cor_method` 可以选择计算哪种相关系数：`"pearson"`、`"spearman"`、`"kendall"`；
- `epsilon` 是选择基于熵的话，对 0 和 1 值的微调量。

本例已对指标列 `PH` 和 `nutrient` 事先做了归一化，故将指标方向设置为 `NA`，以不再重复做归一化，若使用基于熵的方法，则微调 0 和 1 值，避免计算熵时出现 $\ln(0)$。

(1) 标准 CRITIC 权重

```
idx = c("+", NA, "-", NA)
res = critic_weight(df[-1], idx)      # 默认基于标准差
```

- 查看各指标权重

```
res$w
```

```
       O2        PH      germ  nutrient
0.2561818 0.2257389 0.2539392 0.2641401
```

- 查看各评价对象的综合得分：

```
res$s
```

```
 [1] 53.57760 45.72631 64.85384 60.86933 53.11902 44.48714 64.86180 59.22668
```

```
 [9] 76.81822 72.96404 76.91674 60.10287 64.90101 15.63614 55.81794 56.73669
[17] 48.64189 56.87927 34.54560 61.18867
```

- 对应到河流，增加排名列：

```
tibble(ID = df$ID, score = res$s, rank = min_rank(-score)) |>
  arrange(rank)
```

```
# A tibble: 20 x 3
      ID score  rank
   <dbl> <dbl> <int>
 1    11  76.9     1
 2     9  76.8     2
 3    10  73.0     3
 4    13  64.9     4
 5     7  64.9     5
 6     3  64.9     6
 7    20  61.2     7
 8     4  60.9     8
 9    12  60.1     9
10     8  59.2    10
11    18  56.9    11
12    16  56.7    12
13    15  55.8    13
14     1  53.6    14
15     5  53.1    15
16    17  48.6    16
17     2  45.7    17
18     6  44.5    18
19    19  34.5    19
20    14  15.6    20
```

(2) 熵-CRITIC 权重

```
res = critic_weight(df[-1], idx, method = "entropy")    # 基于熵
res$w
```

```
       O2        PH      germ  nutrient
0.3013685 0.1866990 0.3438015 0.1681310
```

- 改用 "spearman" 相关系数：

```
# 基于熵、spearman 相关系数
```

```
res = critic_weight(df[-1], idx, method = "entropy",
                    cor_method = "spearman")
res$w
```

```
       O2        PH      germ  nutrient
0.2932186 0.1924354 0.3486399 0.1657060
```

第五章　主成分赋权法

主成分分析（PCA），是一种多元统计分析方法，其核心思想是在损失很少信息的前提下，将多个原始指标转化为少数几个综合指标。这些转化后的综合指标称为主成分，每个主成分都是原始变量的线性组合，且彼此之间互不相关。这使得主成分在解释数据变异时具有更强的独立性和代表性。因此，主成分比原始变量在某些方面具有更为优越的性能。

主成分法赋权，是一种完全基于数据自身特征（如相关性、方差）来确定权重的客观赋权法。其核心思想是：利用主成分的方差贡献率和载荷（特征向量）来综合确定各原始指标的权重。方差贡献率反映了主成分包含原始数据信息量的多少，载荷则反映了原始指标与主成分之间的相关程度。

当用于综合评价时，若选取的指标较多，且存在一定的相关性，那么不同指标所反映的信息可能具有一定程度的重叠。此时使用主成分法赋权，能够有效避免主观判断带来的随机干扰，同时减少指标间信息重复的问题，使评价体系更加科学、客观。

对于主成分分析，在所有线性组合中，第一主成分 F_1 是方差最大的组合，称为第一主成分；如果第一主成分不足以代表原始数据的全部信息，则引入第二主成分 F_2，并要求 F_2 在包含尽可能多的新信息的同时，不再重复 F_1 的信息。用数学语言表示就是：$COV(F_1, F_2) = 0$。依此类推，可以构造出第 3 个、第 4 个……第 p 个主成分，直至累计方差贡献率达到满意水平。

关于主成分个数的选择：

- 使用全部主成分：主成分赋权的目的不是降维，如果数据质量较高，应该保留全部主成分，因为这样不丢失信息
- 按照 PCA 原则，选择部分主成分：好处是降噪和提高权重的解释性。

5.1　算法步骤

设有 n 个评价对象，m 个评价指标，形成原始指标数据矩阵，其中 x_{ij} 表示第 i 个评价对象第 j 个指标的值。

(1) 无量纲处理：标准化

由于指标通常具有不同的量纲和数量级，必须对原始数据进行预处理以消除量纲影响。主成分分析通常使用标准化：

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

其中，μ_j 和 σ_j 分别为第 j 个指标列的均值和标准差。

对于负向指标，通常建议在标准化后乘以 -1 进行正向化处理。

注意，这样做的目的是确保在后续的综合得分计算中，所有指标的方向一致（越大越好），避免因指标方向不一致导致综合得分逻辑错误。

这样就得到标准化数据矩阵 $Z = (z_{ij})_{n\times m}$，每列均值为 0，标准差为 1。

(2) 计算协方差矩阵

计算标准化数据矩阵 Z 的协方差矩阵，也即原始数据矩阵的相关系数矩阵

$$R = \frac{1}{n-1} Z^T Z$$

(3) 特征分解

对矩阵 R 进行特征分解：

$$R = U \Lambda U^T$$

- $\Lambda = \mathrm{diag}(\lambda_1, \cdots, \lambda_m)$ 是特征值矩阵，满足 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0$。
- $U = [u_1, \cdots, u_m]$ 是正交化单位特征向量矩阵，满足 $U^T U = I$。

(4) 载荷矩阵

$$A = [a_{ji}]_{m\times m} = \left[u_{ji}\sqrt{\lambda_i}\right]_{m\times m}, \qquad i, j = 1, \cdots, m$$

其中，a_{ji} 表示指标 j 在主成分 i 上的载荷（相关系数）。

旋转后载荷矩阵（可选）：$A_{\mathrm{rot}} = \mathrm{Varimax}(A_m \times p)$

- 对前 $p \leq m$ 个主成分的载荷矩阵 $A_{m\times p}$ 应用 Varimax 旋转（若 $p > 1$ 且启用旋转）。
- a_{ji}^{rot} 是旋转后载荷，增强解释性。
- 若无旋转或 $p = 1$，则 $A_{\mathrm{rot}} = A_{m\times p}$。

注：PCA 通常取累计贡献率达到 85% 以上，或特征值 > 1，或基于碎石图、平行分析，选择前 p 个主成分。

(5) 方差贡献率与累积方差贡献率

计算每个主成分的方差贡献率和前 p 个主成分的累计方差贡献率。

$$\eta_j = \frac{\lambda_j}{\sum_{k=1}^{m} \lambda_k}, \quad j = 1, \cdots, m$$

$$\alpha_p = \sum_{j=1}^{p} \eta_j$$

(6) 指标权重

依据指标载荷值（反映该指标与该主成分之间的相关性）和对应的方差贡献（衡量其在总方差中的占比），计算指标在所选主成分中的重要性，通常有两种做法：

绝对值法（更常用，平衡各指标贡献）：$\beta_j = \sum_{i=1}^{p} \eta_i |a_{ji}^{\text{rot}}|$

平方法（突出主导指标）：$\beta_j = \sum_{i=1}^{p} \eta_i [a_{ji}^{\text{rot}}]^2$

再归一化得到指标权重：

$$w_j = \frac{\beta_j}{\sum_{k=1}^{m} \beta_k}, \quad j = 1, \cdots, m$$

(7) 综合得分

基于指标权重和标准化后的数据，计算各评价对象的综合得分：

$$s_i = \sum_{j=1}^{m} w_j z_{ij}, \quad i = 1, \cdots, n$$

5.2 案例：河流水质数据

加载包：

```
library(tidyverse)
library(mathmodels)
```

5.2.1 准备数据

使用内置的 `water_quality` 数据来演示：

```
water_quality
```

```
# A tibble: 20 x 5
      ID    O2    PH  germ nutrient
   <dbl> <dbl> <dbl> <dbl>    <dbl>
```

```
 1     1  4.69  6.59    51    11.9
 2     2  2.03  7.86    19     6.46
 3     3  9.11  6.31    46     8.91
 4     4  8.61  7.05    46    26.4
 5     5  7.13  6.5     50    23.6
 6     6  2.39  6.77    38    24.6
 7     7  7.69  6.79    38     6.01
 8     8  9.3   6.81    27    31.6
 9     9  5.45  7.62     5    18.5
10    10  6.19  7.27    17     7.51
11    11  7.93  7.53     9     6.52
12    12  4.4   7.28    17    25.3
13    13  7.46  8.24    23    14.4
14    14  2.01  5.55    47    26.3
15    15  2.04  6.4     23    17.9
16    16  7.73  6.14    52    15.7
17    17  6.35  7.58    25    29.5
18    18  8.29  8.41    39    12.0
19    19  3.54  7.27    54     3.16
20    20  7.44  6.26     8    28.4
```

这是 20 条河流的水质数据，含氧量越高越好（正向指标）；PH 值越接近 7 越好（居中型指标）；细菌总数越少越好（负向指标）；植物性营养物量介于 $10 \sim 20$ 之间最佳（区间型指标）。

5.2.2 数据预处理

借助 `mathmodels` 包提供的预处理函数，使用 `mutate()` 修改列即可。只需要处理居中型和区间型指标，因为指标标准化已内置到主成分赋权法函数。

```
df = water_quality |>
  mutate(PH = rescale_middle(PH, 7),
         nutrient = rescale_interval(nutrient, 10, 20))
df
```

```
# A tibble: 20 x 5
      ID    O2     PH  germ nutrient
   <dbl> <dbl>  <dbl> <dbl>    <dbl>
 1     1  4.69 0.717     51    1
 2     2  2.03 0.407     19    0.694
 3     3  9.11 0.524     46    0.906
 4     4  8.61 0.966     46    0.444
```

```
 5     5  7.13 0.655     50    0.691
 6     6  2.39 0.841     38    0.601
 7     7  7.69 0.855     38    0.655
 8     8  9.3  0.869     27    0
 9     9  5.45 0.572      5    1
10    10  6.19 0.814     17    0.785
11    11  7.93 0.634      9    0.699
12    12  4.4  0.807     17    0.542
13    13  7.46 0.145     23    1
14    14  2.01 0         47    0.455
15    15  2.04 0.586     23    1
16    16  7.73 0.407     52    1
17    17  6.35 0.6       25    0.182
18    18  8.29 0.0276    39    1
19    19  3.54 0.814     54    0.409
20    20  7.44 0.490      8    0.273
```

5.2.3 主成分赋权

用 pca_weight() 函数实现用**主成分法**计算权重，该函数已内置标准化处理，基本语法：

```
pca_weight(X, index = NULL, nfs = NULL, varimax = TRUE, method = "abs")
```

- index 可以指定对哪些列做正向（"+"）、负向（"-"）、不做（NA）标准化；
- nfs 选择主成分数，默认为全部主成分（即 X 的列数）；
- varimax 设置是否对载荷矩阵做正交最大旋转，默认为 TRUE；
- method 设置计算指标权重使用载荷的方法："abs"（默认，绝对值法）、"squared"（平方法）。

两个已经归一化的指标仍做标准化，germ 为负向指标，选择 3 个主成分，做主成分赋权：

```
idx = c("+", "+", "-", "+")
res = pca_weight(df[-1], idx, nfs = 3)
```

- 查看指标权重：

```
res$w
```

```
[1] 0.2295767 0.2649918 0.2441633 0.2612682
```

- 查看各个评价对象得分：

```
res$s
```

```
 [1]  0.001198524 -0.320107088  0.217018493  0.195405945 -0.084231291
 [6] -0.238983664  0.307469102  0.070879636  0.625563844  0.558020048
[11]  0.593610342  0.178356317  0.131099010 -1.337577994  0.054638084
```

```
[16] -0.030864174 -0.269580866 -0.145513855 -0.565644280  0.059243867
```

- 对应到河流，增加排名列：

```
tibble(ID = df$ID, score = res$s, rank = min_rank(-score)) |>
  arrange(rank)
```

```
# A tibble: 20 x 3
      ID    score  rank
   <dbl>    <dbl> <int>
 1     9  0.626       1
 2    11  0.594       2
 3    10  0.558       3
 4     7  0.307       4
 5     3  0.217       5
 6     4  0.195       6
 7    12  0.178       7
 8    13  0.131       8
 9     8  0.0709      9
10    20  0.0592     10
11    15  0.0546     11
12     1  0.00120    12
13    16 -0.0309     13
14     5 -0.0842     14
15    18 -0.146      15
16     6 -0.239      16
17    17 -0.270      17
18     2 -0.320      18
19    19 -0.566      19
20    14 -1.34       20
```

- 查看 3 个主成分的累积贡献：

```
res$cum_contrib
```

```
[1] 0.8400963
```

- 查看旋转的载荷矩阵：

```
res$loading
```

```
                   PC1          PC2          PC3
O2       -0.0008082562 -0.001248033 -0.999822686
PH        0.7123628135  0.095775451 -0.013746300
germ      0.0009564714 -0.990774991 -0.001308225
```

```
nutrient -0.7018102691  0.095866694 -0.012803306
```

可以据此，解释这 3 个主成分（略）。

第六章　主客观组合赋权

综合评价中，权重的确定往往直接影响最终结果的科学性与合理性。

主观权重，是由专家经验或决策者偏好人为赋予，不依赖于具体数据；而客观权重，则是基于实际数据通过数学方法计算得出，能够较好地反映指标的实际差异与规律。单一使用主观或客观权重均存在局限：主观权重易受人为偏见影响，客观权重则可能忽视实际情境中的重要因素。

主客观组合赋权法，能够将主观判断与客观数据有机结合，在兼顾实际背景的同时提升结果的科学性，从而为多指标综合评价提供更加合理、稳健的权重分配方案。

6.1　四种主客观组合赋权法

(1) 线性组合法

根据领域知识或主客观侧重程度，人为指定一个组合系数 α，将主观权重、客观权重做线性组合：

$$w = \alpha * w^s + (1-\alpha) * w^o$$

比如，认为主客观权重同样重要，可取 $\alpha = 0.5$。

(2) 乘法综合法（乘积归一化）

只有主客观权重同时较高时，综合权重才会高；若任意一个权重很低，综合权重就会被大幅拉低。类似于 “一票否决”——主观和客观都认可，这个指标才真正重要。

具体是先计算主客观权重的乘积，再归一化结果：

$$w_i = \frac{w_i^s \cdot w_i^o}{\sum_{j=1}^{m} w_j^s \cdot w_j^o}$$

(3) 博弈均衡几何平均法（几何平均归一化）

博弈论，是研究多方决策主体在交互环境中如何达成最优策略的重要理论工具。在综合评价中，主观权重与客观权重常存在差异，二者可视为博弈中的 “双方”。通过引入博弈论思想，可寻求主客观权重的均衡组合，即在冲突与妥协中达成最优权重分配，从而提升评价结果的科学性与合理性。

优化目标：最小化组合权重 w 与主客观权重的联合相对熵（KL 散度）：

$$\min_{w} \sum_{i=1}^{n} w_i \Big(\ln \frac{w_i}{w_i^s} + \ln \frac{w_i}{w_i^o} \Big), \quad \text{s.t.} \quad \sum_{i=1}^{n} w_i = 1,\ w_i > 0$$

求解优化问题：通过拉格朗日乘子法，得到最优解：

$$w_i = \frac{\sqrt{w_i^s \cdot w_i^o}}{\sum_{j=1}^{n} \sqrt{w_j^s \cdot w_j^o}}$$

该组合权重是主客观权重的几何平均的归一化，更温和地平衡了双方的信息差异。

(4) 博弈最优线性组合法

传统线性组合法虽能调和主观权重与客观权重的冲突，但固定系数（如 $\alpha = 0.5$）缺乏理论支撑，难以反映权重的动态博弈关系。

博弈最优线性组合法，基于博弈论均衡思想，将主客观权重视为博弈双方，通过最小二乘优化求解最优组合系数，相比几何平均法，该方法更适用于需平滑调和主客观权重冲突的场景。

设组合权重表示为主客观权重的线性组合：

$$w = \alpha w^s + \beta w^o$$

其中，α, β 为待定系数，满足 $\alpha + \beta = 1$。

优化目标：最小化组合权重与原始权重的差异：

$$\min \left(\|w - w^s\|_2 + \|w - w^o\|_2 \right)$$

求解优化问题：根据矩阵微分性质，可得最优解满足：

$$\begin{bmatrix} (w^s)^T w^s & (w^s)^T w^o \\ (w^o)^T w^s & (w^o)^T w^o \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} (w^s)^T w^s \\ (w^o)^T w^o \end{bmatrix}$$

求解得到的 α 和 β，再进行归一化：

$$\alpha^* = \frac{\alpha}{\alpha + \beta}, \quad \beta^* = \frac{\beta}{\alpha + \beta}$$

从而，最终权重为 $w^* = \alpha^* w^s + \beta^* w^o$。

6.2 R 实现

`mathmodels` 包提供了 `combine_weights()` 函数实现上述四种主客观组合赋权法，基本语法：

```
combine_weights(w_subj, w_obj, type = "linear", alpha = 0.5)
```

- `w_subj`, `w_obj` 分别为已归一化的主客观权重向量；

- type 设置组合方法："linear"（线性组合法），"multiplicative"（乘法综合法）、"game"（博弈均衡几何平均法）、"game_linear"（博弈最优线性组合法）
- alpha 与 type = "linear" 连用，设置组合系数，默认为 0.5

演示示例：

```
library(mathmodels)
w_subj = c(0.4, 0.3, 0.2, 0.1)          # 主观权重
w_obj = c(0.25, 0.2, 0.3, 0.25)         # 客观权重
```

(1) 线性组合法，取

```
combine_weights(w_subj, w_obj, type = "linear", alpha = 0.6)
```

```
[1] 0.34 0.26 0.24 0.16
```

(2) 乘法综合法

```
combine_weights(w_subj, w_obj, type = "multiplicative")
```

```
[1] 0.4081633 0.2448980 0.2448980 0.1020408
```

(3) 博弈均衡几何平均法

```
combine_weights(w_subj, w_obj, type = "game")
```

```
[1] 0.3279556 0.2540333 0.2540333 0.1639778
```

(4) 博弈最优线性组合法

```
combine_weights(w_subj, w_obj, type = "game_linear")
```

```
[1] 0.3735683 0.2823789 0.2176211 0.1264317
```

第七章　TOPSIS

TOPSIS 法，也称为理想解法，是一种多目标决策方法，能充分利用原始数据的信息，结果能精确地反映各评价对象之间的差距。其基本原理是：

- 将 m 个评价指标看成 m 条坐标轴，由此可以构造出一个 m 维空间，则每个评价对象依照其各项指标值就对应 m 维空间中一个坐标点；
- 针对各项指标从所有评价对象中选出该指标的最优值（正理想解，对应最优坐标点）和最差值（负理想解，对应最差坐标点），依次求出各个待评价对象的坐标点分别到最优坐标点和最差坐标点的距离；
- 距离正理想解越近越好，距离负理想解越远越好，为此构造综合评价指标：每个评价对象的相对接近度，据此确定评价对象的优劣。

7.1　算法步骤

设有 n 个评价对象，m 个评价指标，形成原始指标数据矩阵 A，称为**决策矩阵**，其中 a_{ij} 表示第 i 个评价对象第 j 个指标的值。

若存在居中型、区间型指标，先做归一化处理，得到的数据仍记为 A。

(1) 数据无量纲化处理：规范化

$$b_{ij}=\frac{a_{ij}}{\sqrt{\sum_{i=1}^{n}{a_{ij}}^2}},\quad i=1,\cdots n;\ j=1,\cdots,m$$

记 $B_{n\times m}=(b_{ij})_{n\times m}$ 称为**规范决策矩阵**，规范化法处理后，同一评价指标的各样本值的平方和为 1，适合 TOPSIS 法中计算欧氏距离。

(2) 计算加权规范决策矩阵

根据赋权法赋以合理的权重 $w=[w_1,w_2,\cdots,w_m]$，将 B 的第 j 列乘以其权重 w_j 得到**加权规范决策矩阵**：

$$C_{n\times m}=B_{n\times m}\cdot *\begin{pmatrix}w\\ \vdots\\ w\end{pmatrix}_{n\times m}$$

(3) 确定正理想解 C 和负理想解 C^0

$$C^* = (c_1^*, c_2^*, \cdots, c_m^*), \quad C^0 = (c_1^0, c_2^0, \cdots, c_m^0)$$

其中，

$$c_j^* = \begin{cases} \max\limits_i c_{ij}, & \text{若指标 } j \text{ 为正向指标} \\ \min\limits_i c_{ij}, & \text{若指标 } j \text{ 为负向指标} \end{cases} \quad c_j^0 = \begin{cases} \min\limits_i c_{ij}, & \text{若指标 } j \text{ 为正向指标} \\ \max\limits_i c_{ij}, & \text{若指标 } j \text{ 为负向指标} \end{cases}$$

(4) 计算各个样本到正理想解和负理想解的欧氏距离

$$d_i^* = \sqrt{\sum_{j=1}^m (c_{ij} - c_j^*)^2}, \quad d_i^0 = \sqrt{\sum_{j=1}^m (c_{ij} - c_j^0)^2}, \qquad i = 1, \cdots n$$

(5) 计算每个样本到理想解的相对接近度

定义相对接近度如下，来评判各个评价对象的优劣：

$$f_i = \frac{d_i^0}{d_i^0 + d_i^*}, \quad i = 1, \cdots, n$$

7.2 案例：河流水质评价

加载包：

```
library(tidyverse)
library(mathmodels)
```

使用内置的 `water_quality` 数据来演示：

```
water_quality
```

```
# A tibble: 20 x 5
      ID    O2    PH  germ nutrient
   <dbl> <dbl> <dbl> <dbl>    <dbl>
 1     1  4.69  6.59    51    11.9
 2     2  2.03  7.86    19     6.46
 3     3  9.11  6.31    46     8.91
 4     4  8.61  7.05    46    26.4
 5     5  7.13  6.5     50    23.6
 6     6  2.39  6.77    38    24.6
 7     7  7.69  6.79    38     6.01
 8     8  9.3   6.81    27    31.6
```

```
 9     9  5.45  7.62     5    18.5
10    10  6.19  7.27    17     7.51
11    11  7.93  7.53     9     6.52
12    12  4.4   7.28    17    25.3
13    13  7.46  8.24    23    14.4
14    14  2.01  5.55    47    26.3
15    15  2.04  6.4     23    17.9
16    16  7.73  6.14    52    15.7
17    17  6.35  7.58    25    29.5
18    18  8.29  8.41    39    12.0
19    19  3.54  7.27    54     3.16
20    20  7.44  6.26     8    28.4
```

这是 20 条河流的水质数据，含氧量越高越好（正向指标）；PH 值越接近 7 越好（居中型指标）；细菌总数越少越好（负向指标）；植物性营养物量介于 10 ~ 20 之间最佳（区间型指标）。

7.2.1 数据预处理

每个指标数据是一列，借助 `mathmodels` 包提供的预处理函数修改列即可。

中间型、区间型指标必须要先做归一化处理。正向、负向指标不做处理，因为熵权法和 TOPSIS 都会自己对正向、负向指标做归一化/规范化处理。

```
df = water_quality |>
  mutate(PH = rescale_middle(PH, 7),
         nutrient = rescale_interval(nutrient, 10, 20))
df
```

```
# A tibble: 20 x 5
      ID    O2     PH  germ nutrient
   <dbl> <dbl>  <dbl> <dbl>    <dbl>
 1     1  4.69 0.717     51    1
 2     2  2.03 0.407     19    0.694
 3     3  9.11 0.524     46    0.906
 4     4  8.61 0.966     46    0.444
 5     5  7.13 0.655     50    0.691
 6     6  2.39 0.841     38    0.601
 7     7  7.69 0.855     38    0.655
 8     8  9.3  0.869     27    0
 9     9  5.45 0.572      5    1
10    10  6.19 0.814     17    0.785
11    11  7.93 0.634      9    0.699
```

```
12    12  4.4  0.807     17    0.542
13    13  7.46 0.145     23    1
14    14  2.01 0         47    0.455
15    15  2.04 0.586     23    1
16    16  7.73 0.407     52    1
17    17  6.35 0.6       25    0.182
18    18  8.29 0.0276    39    1
19    19  3.54 0.814     54    0.409
20    20  7.44 0.490      8    0.273
```

7.2.2 指标权重：熵权法

使用 entropy_weight()，PH 和 nutrient 已归一化，不再重复做：

```
res = entropy_weight(df[2:5], c("+",NA,"-",NA))
res$w
```

```
       O2        PH      germ  nutrient
0.3153202 0.1813149 0.3506929 0.1526721
```

7.2.3 TOPSIS 评价

mathmodels 包提供了 topsis() 函数实现 TOPSIS 法，该函数已内置规范化处理，基本语法：

```
topsis(X, w = NULL, index = NULL)
```

- X 为决策矩阵，w 为指标权重，index 指定指标正负向："+" 表示正向指标，"-" 表示负向指标，默认为 NULL 表示都作为正向指标。

```
idx = c("+","+","-","+")
RC = topsis(df[2:5], res$w, idx)
RC                # 相对接近度
```

```
 [1] 0.3841556 0.4852199 0.5021310 0.5084128 0.4226200 0.3945085 0.5502261
 [8] 0.6258551 0.7260529 0.7138447 0.8091923 0.6159619 0.6007565 0.1593096
[15] 0.4974685 0.4250799 0.5565555 0.4815844 0.3070290 0.7098355
```

转化成 0 ∼ 100 分数，增加排名列：

```
tibble(ID = df$ID, closeness = RC, score = rescale(closeness, b = 100),
       rank = min_rank(-score)) |>
  arrange(rank)
```

```
# A tibble: 20 x 4
      ID closeness score  rank
```

	<dbl>	<dbl>	<dbl>	<int>
1	11	0.809	100	1
2	9	0.726	87.2	2
3	10	0.714	85.3	3
4	20	0.710	84.7	4
5	8	0.626	71.8	5
6	12	0.616	70.3	6
7	13	0.601	67.9	7
8	17	0.557	61.1	8
9	7	0.550	60.2	9
10	4	0.508	53.7	10
11	3	0.502	52.8	11
12	15	0.497	52.0	12
13	2	0.485	50.1	13
14	18	0.482	49.6	14
15	16	0.425	40.9	15
16	5	0.423	40.5	16
17	6	0.395	36.2	17
18	1	0.384	34.6	18
19	19	0.307	22.7	19
20	14	0.159	0	20

第八章　灰色关联分析

灰色关联分析源于几何直观，实质上是一种曲线间几何形状的比较：几何形状越接近，则发展变化趋势越接近，关联程度就越大。

8.1　灰色关联度

8.1.1　数据无量纲处理

对单位不一，初值不同的序列，应首先进行初值化，即将该序列的所有数据分别除以首项数据，将变量化为无单位的相对数值。负向数据需要做正向化，若用取倒数变换，可以和初值化一起做，即用首项数据除以该所有数据；

或者，

- 做数据均值化，所有数据都除以均值
- 做数据百分比化，所有数据都除以最大值
- 做数据归一化

`mathmodels` 包提供了初值化、均值化、最大值化函数 `rescale_initial()`, `rescale_mean()`, `rescale_extreme()` 专用于灰色关联预处理。

8.1.2　计算关联系数

设参考序列为

$$X_0 = \{x_0(1), x_0(2), \cdots, x_0(m)\}$$

比较序列为

$$X_i = \{x_i(1), x_i(2), \cdots, x_i(m)\}, \quad i = 1, \cdots, n$$

比较序列 X_i 对参考序列 X_0 在 k 处的关联系数定义为：

$$\eta_i(k) = \frac{\min_s \min_t |x_0(t) - x_s(t)| + \rho \max_s \max_t |x_0(t) - x_s(t)|}{|x_0(k) - x_i(k)| + \rho \max_s \max_t |x_0(t) - x_s(t)|}$$

其中，$\min_s \min_t |x_0(t) - x_s(t)|$ 和 $\max_s \max_t |x_0(t) - x_s(t)|$ 分别称为两级最小差、两级最大差；ρ 称为分辨系数，越大分辨率越大，一般采用 $\rho = 0.5$。

8.1.3 计算灰色关联度

关联系数只表示了各个位置参考序列和比较序列之间的关联程度，为了从总体上了解序列之间的关联程度，必须求出它们的平均值，即**灰色关联度**：

$$r_i = \frac{1}{n}\sum_{k=1}^{n}\eta_i(k)$$

若各指标有不同的权重，可以对进行加权平均，得到**灰色加权关联度**。

结果解读：

- $r \in [0, 0.35)$，称为弱关联
- $r \in (0.35, 0.65)$，称为中度关联
- $r \in [0.65, 1]$，称为强关联

加载包：

```
library(tidyverse)
library(mathmodels)
```

8.2 案例 1：运动员训练与成绩

对某健将级女子铅球运动员的跟踪调查，获得其 1982 年至 1986 年每年最好成绩及 16 项专项素质和身体素质的数据，做灰色关联分析，看哪些指标与铅球成绩关联度更高？

8.2.1 创建数据

```
sports = readxl::read_excel("data/sports.xlsx")
sports
```

```
# A tibble: 5 x 18
   年份  铅球 `4kg前抛` `4kg后抛` `4kg原地` 立定跳远  高翻  抓举  卧推 `3kg前抛`
  <dbl> <dbl>     <dbl>     <dbl>     <dbl>    <dbl> <dbl> <dbl> <dbl>     <dbl>
1  1982  13.6      11.5      13.8      12.4     2.48    85    55    65      12.8
2  1983  14.0      13        16.4      12.7     2.49    85    65    70      15.3
3  1984  14.5      15.2      16.9      14.0     2.56    90    75    75      16.2
4  1985  15.6      15.3      16.6      14.0     2.64   100    80    85      16.4
5  1986  15.7      15.0      17.3      13.5     2.59   105    80    90      17.0
# i 8 more variables: `3kg后抛` <dbl>, `3kg原地` <dbl>, `3kg滑步` <dbl>,
#   三级跳 <dbl>, 全蹲 <dbl>, 挺举 <dbl>, `30米跑` <dbl>, `100米跑` <dbl>
```

8.2.2 无量纲化处理

```
sports = sports |>
  mutate(across(2:16, rescale_initial),
         across(17:18, \(x) rescale_initial(x, "-")))
sports
```

```
# A tibble: 5 x 18
   年份  铅球 `4kg前抛` `4kg后抛` `4kg原地` 立定跳远  高翻  抓举  卧推 `3kg前抛`
  <dbl> <dbl>     <dbl>     <dbl>     <dbl>    <dbl> <dbl> <dbl> <dbl>     <dbl>
1  1982  1         1         1         1        1     1     1     1         1
2  1983  1.03      1.13      1.19      1.02     1.00  1     1.18  1.08      1.20
3  1984  1.07      1.32      1.23      1.12     1.03  1.06  1.36  1.15      1.27
4  1985  1.15      1.33      1.20      1.13     1.06  1.18  1.45  1.31      1.28
5  1986  1.15      1.31      1.26      1.08     1.04  1.24  1.45  1.38      1.33
# i 8 more variables: `3kg后抛` <dbl>, `3kg原地` <dbl>, `3kg滑步` <dbl>,
#   三级跳 <dbl>, 全蹲 <dbl>, 挺举 <dbl>, `30米跑` <dbl>, `100米跑` <dbl>
```

8.2.3 计算灰色关联度

mathmodel 包提供了 grey_corr() 函数实现计算灰色关联度，基本语法：

```
grey_corr(ref, cmp, rho = 0.5, w = NULL)
```

- ref 为参考序列
- cmp 为比较序列
- rho 为分辨系数
- w 为指标权重向量

采用默认分辨系数，不提供指标权重：

```
res = grey_corr(sports[[2]], sports[,3:18])
```

- 将结果转化为数据框，并排序：

```
enframe(res, value = " 灰色关联度") |>
  arrange(-灰色关联度)
```

```
# A tibble: 16 x 2
   name      灰色关联度
   <chr>          <dbl>
 1 全蹲           0.933
 2 3kg滑步        0.895
 3 高翻           0.855
 4 4kg原地        0.854
```

```
 5 挺举          0.847
 6 立定跳远      0.776
 7 30米跑        0.745
 8 100米跑       0.726
 9 3kg原地       0.708
10 三级跳        0.705
11 3kg后抛       0.683
12 4kg后抛       0.663
13 卧推          0.659
14 4kg前抛       0.588
15 3kg前抛       0.582
16 抓举          0.502
```

8.3 案例 2：投资产出优势分析

灰色关联分析的参考序列只有 1 个，当参考序列不止 1 个时，让比较序列和各个参考序列都做一遍灰色关联分析，得到灰色关联度矩阵，叫作**优势分析**。

设有 m 个参考序列，l 个比较序列，则这 l 个比较序列对每一个参考序列都有 1 个关联度，记 r_ij 表示第 j 个比较序列对第 i 个参考序列的关联度，可得到关联度矩阵 $R = (r_{ij})_m \times l$。

根据矩阵 R 的各元素的大小，可分析判断出哪些因素起主要影响（优势因素），哪些因素起次要影响。

8.3.1 创建数据

```
invest = read_csv("data/invest.csv")
invest
```

```
# A tibble: 5 x 12
   year    x1    x2    x3    x4    x5    y1    y2    y3    y4    y5    y6
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  1979  309.  195.  24.6  20    19.0  170   57.6  88.6  11.2  4.03  13.7
2  1980  310   190.  21    25.6  19    174   70.7  70    13.3  4.26  15.6
3  1981  295   187.  12.2  23.3  22.3  197   76.8  85.4  16.8  4.34  13.8
4  1982  346   205   15.1  29.2  23.5  216.  80.7  99.8  18.9  5.06  12.0
5  1983  367   223.  14.6  30    27.7  236.  89.8 103.   22.8  5.78  14.0
```

8.3.2 无量纲化处理

```
invest = invest |>
  mutate(across(-1, rescale_initial))
```

```
invest
```

```
# A tibble: 5 x 12
   year    x1    x2    x3    x4    x5    y1    y2    y3    y4    y5    y6
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  1979 1     1     1     1     1     1     1     1     1     1     1
2  1980 1.00  0.972 0.854 1.28  1.00  1.02  1.23 0.790 1.19  1.06 1.14
3  1981 0.956 0.958 0.496 1.16  1.17  1.16  1.33 0.964 1.50  1.08 1.01
4  1982 1.12  1.05  0.614 1.46  1.24  1.27  1.40 1.13  1.69  1.26 0.874
5  1983 1.19  1.14  0.592 1.5   1.46  1.39  1.56 1.17  2.04  1.43 1.02
```

8.3.3 优势分析：批量计算灰色关联度

```
map_dfc(invest[7:12], \(x) grey_corr(x, invest[2:6])) |>
  mutate(`投资/产出` = names(invest)[2:6], .before = 1)
```

```
# A tibble: 5 x 7
  `投资/产出`    y1    y2    y3    y4    y5    y6
  <chr>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 x1          0.802 0.689 0.891 0.678 0.811 0.743
2 x2          0.761 0.666 0.858 0.663 0.774 0.766
3 x3          0.557 0.529 0.579 0.568 0.565 0.562
4 x4          0.810 0.885 0.577 0.780 0.804 0.607
5 x5          0.935 0.800 0.675 0.731 0.921 0.632
```

8.4 灰色关联评价

灰色关联评价，类似于理想解法，算法步骤如下：

- 对指标数据预处理：一致化、规范化后，得到规范矩阵；
- 由于各个指标对综合评价的权重不同，需要根据指标权重对规范矩阵做加权，得到加权规范矩阵；
- 构造参考样本，为“正理想样本”，即各个指标取最大值，构成的最佳样本；
- 将每个样本（评价对象）看作是比较序列，将参考样本作为参考序列，代入灰色关联分析算法，计算灰色关联度。
- 根据该灰色关联度，就可以排序或评价样本的优劣。

以评价河流水质为例。

8.4.1 数据预处理

- 只将中间型、区间型指标做归一化：

```
df = water_quality |>
  mutate(PH = rescale_middle(PH, 7),
         nutrient = rescale_interval(nutrient, 10, 20))
df
```

```
# A tibble: 20 x 5
      ID    O2     PH  germ nutrient
   <dbl> <dbl>  <dbl> <dbl>    <dbl>
 1     1  4.69 0.717     51    1
 2     2  2.03 0.407     19    0.694
 3     3  9.11 0.524     46    0.906
 4     4  8.61 0.966     46    0.444
 5     5  7.13 0.655     50    0.691
 6     6  2.39 0.841     38    0.601
 7     7  7.69 0.855     38    0.655
 8     8  9.3  0.869     27    0
 9     9  5.45 0.572      5    1
10    10  6.19 0.814     17    0.785
11    11  7.93 0.634      9    0.699
12    12  4.4  0.807     17    0.542
13    13  7.46 0.145     23    1
14    14  2.01 0         47    0.455
15    15  2.04 0.586     23    1
16    16  7.73 0.407     52    1
17    17  6.35 0.6       25    0.182
18    18  8.29 0.0276    39    1
19    19  3.54 0.814     54    0.409
20    20  7.44 0.490      8    0.273
```

8.4.2 指标权重：熵权法

已经归一化的指标，不必重复做归一化：

```
idx = c("+", NA, "-", NA)
res = entropy_weight(df[2:5], idx)
res$w
```

```
       O2        PH      germ  nutrient
0.3153202 0.1813149 0.3506929 0.1526721
```

8.4.3 灰色关联评价

mathmodels 包提供了 grey_corr_topsis() 函数实现灰色关联 TOPSIS 法，基本语法：

```
grey_corr_topsis(X, w, index = NULL, rho = 0.5)
```

- X 为决策矩阵
- w 为指标权重
- index 可以指定对哪些列做正向（"+"）、负向（"-"）、不做（NA）归一化，默认 index = NULL 表示，所有列都不做归一化。
- rho 为分辨系数

执行灰色关联 TOPSIS 评价，已经归一化的指标，不必重复做归一化：

```
gcd = grey_corr_topsis(df[2:5], res$w, idx)
```

转化成 0-100 分数，增加排名列：

```
tibble(ID = df$ID, 灰色关联度 = gcd,
       score = rescale(灰色关联度, b = 100),
       rank = min_rank(-score)) |>
  arrange(-score)
```

```
# A tibble: 20 x 4
      ID 灰色关联度 score  rank
   <dbl>      <dbl> <dbl> <int>
 1    11      0.613 100       1
 2     9      0.613  99.9     2
 3    10      0.592  91.9     3
 4     7      0.562  80.4     4
 5    20      0.560  79.6     5
 6     3      0.558  79.0     6
 7     8      0.557  78.6     7
 8    13      0.552  76.7     8
 9     4      0.552  76.5     9
10    12      0.545  73.9    10
11    18      0.520  64.2    11
12    16      0.511  60.9    12
13    15      0.506  58.8    13
14     5      0.503  57.6    14
15    17      0.502  57.3    15
16     1      0.501  56.9    16
17     6      0.472  45.8    17
18     2      0.469  44.8    18
```

19	19	0.430	29.8	19
20	14	0.353	0	20

第九章　秩和比法

秩和比法（Rank-sum ratio，RSR）由田凤调教授于 1988 年提出，集古典参数统计与近代非参数统计各自优点于一体的统计分析方法。RSR 法现在广泛地应用于医疗卫生、科技、经济等邻域的多指标综合评价、统计预测预报、鉴别分类、统计质量控制等方面。

RSR 一般过程是将正向指标从小到大排序进行排名、负向指标从大到小排序进行排名，再计算秩和比，最后统计回归、分档排序。通过秩转换，获得无量纲统计量 RSR；在此基础上，运用参数统计分析的概念与方法，研究 RSR 的分布；以 RSR 值对评价对象的优劣直接排序或分档排序，从而对评价对象做出综合评价。

优点：以非参数法为基础，对指标的选择无特殊要求，适用于各种评价对象，由于计算时使用的数值是秩次，可以消除异常值的干扰

缺点：排序的主要依据是利用原始数据的秩次，最终算得的 RSR 值反映的是综合秩次的差距，而与原始数据的顺位间的差距程度大小无关，这样在指标转化为秩次是会失去一些原始数据的信息，如原始数据的大小差别等。当 RSR 值实际上不满足正态分布时，分档归类的结果与实际情况会有偏差，且只能回答分级程度是否有差别，不能进一步回答具体的差别情况。

RSR 法本质

RSR 只使用了数据的相对大小关系，而不真正运用数值本身，也能用于处理 “好”、“较好”、“一般” 这类模糊指标问题。只要选择恰当的权重，RSR 法也能转化为模糊综合评价。

实际上，只要确定了权重，编好了秩，原始样本的排序就结束了，但是 RSR 综合评价法再进行了统计回归、分档排序，这一步实际上是通过将样本的秩次分布映射到正态分布曲线上，运用正态分布 3σ 原则或其它连续变量离散化方法进行分档。

9.1　算法步骤

设有 n 个评价对象，m 个评价指标，形成原始指标数据矩阵 X，其中 x_{ij} 表示第 i 个评价对象第 j 个指标的值。

9.1.1　编秩

整数秩

编出每个指标各评价对象的秩，其中正向指标从小到大编秩，负向指标从大到小编秩，同一指标数据相同者编平均秩。例如，某指标下第 3、4、5 名的值相同（原始应占秩次 3, 4, 5），则它们的秩次均为 $(3+4+5)/3=4$。下一个不同的值则编为秩次 6。

非整数秩

用类似于线性插值的方式对指标值进行编秩，以改进 RSR 法编秩方法的不足，所编秩次与原指标值之间存在定量的线性对应关系，从而克服了 RSR 法秩次化时易损失原指标值定量信息的缺点。

对于正向指标：

$$R_{ij}=1+(n-1)\frac{x_{ij}-\min\limits_i x_{ij}}{\max\limits_i x_{ij}-\min\limits_i x_{ij}}$$

对于负向指标：

$$R_{ij}=1+(n-1)\frac{\max\limits_i x_{ij}-x_{ij}}{\max\limits_i x_{ij}-\min\limits_i x_{ij}}$$

非整数秩是将传统整数排名（如 1, 2, 3, …）推广到连续区间的一种方法，其核心在于根据评价对象在数据中的相对位置，赋予其介于 1 与 n 之间的实数值，而非简单的整数秩次，从而更精细地刻画排名信息。该方法本质上是对原始数据进行线性归一化到 $[1,n]$（即线性插值处理），使排名由跳跃式的离散值转变为反映个体间细微差异的连续变量，提升综合评价的灵敏度与准确性。

该步得到秩矩阵，记为 $R=(R_{ij})_{m\times n}$。

9.1.2 计算秩和比并排序

秩和比 RSR_i 定义为第 i 个评价对象在所有 n 个指标下的秩和相对于最大可能秩次（为 n ）的比例：

$$RSR_i=\frac{1}{n}\sum_{j=1}^{m}w_jR_{ij}$$

其中，w_j 为第 j 个指标的权重，满足 $\sum_j=1^m w_j=1$。

当指标权重相同时，$w_j\equiv\frac{1}{m}$，此时秩和比可以表示为：

$$RSR_i=\frac{1}{mn}\sum_{j=1}^{m}R_{ij}$$

9.1.3 确定 RSR 的分布（转化为概率单位）

RSR 的分布是指用概率单位 Probit 表达的值特定的累计频率。Probit 模型是一种广义的线性模型，服从正态分布。其转换方法为：

(1) 编制 RSR 频数分布表，列出各组频数 f_i，计算各组累计频数 cf_i；
(2) 确定各组 RSR 的秩次范围及平均秩次；
(3) 计算累计频率 $p_i=cf_i/n\times100\%$，最后一项记为 $1-\frac{1}{4n}$ 进行修正。（使用离散分布作为正态分布的近似计算中，作些修正可以提高精度。这里若不做修正，得到的 $Probit\to\infty$，不能用于计算）

(4) 将累计频率换算为概率单位 Probit_i，为累计频率对应的标准正态分布的 p_i 分位数加 5。|

9.1.4 拟合线性回归模型，计算模型估计值

以累积频率所对应的概率单位 Probit 为自变量，以 RSR 值为因变量，计算直线回归方程，即：

$$RSR = a + b * Probit$$

回归方程需要做常规的回归诊断保证模型可用（略）。

计算线性回归模型的预测值：

$$RSRfit_i = a + b * \text{Probit}_i$$

9.1.5 进行分档排序

根据 $RSRfit_i$ 值对评价对象进行分档排序，分档数由研究者根据实际情况决定。

分档排序，实际上就是连续数值离散化，有很多种方法，只要合理就行。

9.2 案例：孕产妇保健评价

加载包：

```
library(tidyverse)
library(mathmodels)
```

9.2.1 准备数据

对某省 10 个地区孕产妇保健工作就 3 个指标（产前检查率 (%)、孕妇死亡率 (1/10 万)、围产儿死亡率 (%)），进行秩和比综合评价。

```
df = tibble(
  ID = LETTERS[1:10],
  x1 = c(99.54,96.52,99.36,92.83,91.71,95.35,96.09,99.27,94.76,84.80),
  x2 = c(60.27,59.67,43.91,58.99,35.40,44.71,49.81,31.69,22.91,81.49),
  x3 = c(16.15,20.10,15.60,17.04,15.01,13.93,17.43,13.89,19.87,23.63))
df
```

```
# A tibble: 10 x 4
   ID       x1    x2    x3
   <chr> <dbl> <dbl> <dbl>
 1 A      99.5  60.3  16.2
```

```
 2 B      96.5  59.7  20.1
 3 C      99.4  43.9  15.6
 4 D      92.8  59.0  17.0
 5 E      91.7  35.4  15.0
 6 F      95.4  44.7  13.9
 7 G      96.1  49.8  17.4
 8 H      99.3  31.7  13.9
 9 I      94.8  22.9  19.9
10 J      84.8  81.5  23.6
```

数据预处理，x2 和 x3 是负向指标，取倒数即可（不影响编秩）：

```
df1 = df |>
  mutate(across(x2:x3, \(x) 1 / x))
df1
```

```
# A tibble: 10 x 4
   ID       x1     x2     x3
   <chr> <dbl>  <dbl>  <dbl>
 1 A      99.5 0.0166 0.0619
 2 B      96.5 0.0168 0.0498
 3 C      99.4 0.0228 0.0641
 4 D      92.8 0.0170 0.0587
 5 E      91.7 0.0282 0.0666
 6 F      95.4 0.0224 0.0718
 7 G      96.1 0.0201 0.0574
 8 H      99.3 0.0316 0.0720
 9 I      94.8 0.0436 0.0503
10 J      84.8 0.0123 0.0423
```

9.2.2 秩和比法评价

mathmodels 包提供了 rank_sum_ratio() 函数实现用秩和比法综合评价，基本语法：

```
rank_sum_ratio(data, w = NULL, method = "int")
```

- data 为原始指标数据，首列为评价对象 ID；
- w 为指标权重，可以通过各种赋权法得到，默认是等权重；
- method 设置编整数秩（"int"）还是非整数秩（"non-int"）。

9.2.3 重现经典案例结果

采用等指标权重、编整数秩：

```
res = rank_sum_ratio(df1)
```

- 查看结果表（包含中间结果）：

```
res$resultTable
```

```
# A tibble: 10 x 8
   ID      RSR  barR     f  sumf barRn Probit RSRfit
   <chr> <dbl> <dbl> <int> <int> <dbl>  <dbl>  <dbl>
 1 J     0.1     1       1     1 0.1     3.72  0.216
 2 B     0.4     2.5     2     3 0.25    4.33  0.350
 3 D     0.4     2.5     2     3 0.25    4.33  0.350
 4 G     0.5     4       1     4 0.4     4.75  0.444
 5 I     0.567   5       1     5 0.5     5     0.500
 6 A     0.6     6.5     2     7 0.65    5.39  0.585
 7 E     0.6     6.5     2     7 0.65    5.39  0.585
 8 F     0.667   8       1     8 0.8     5.84  0.686
 9 C     0.767   9       1     9 0.9     6.28  0.784
10 H     0.9    10       1    10 0.975   6.96  0.934
```

- 查看编秩结果：

```
res$rankTable
```

```
# A tibble: 10 x 4
   ID       x1    x2    x3
   <chr> <dbl> <dbl> <dbl>
 1 A        10     2     6
 2 B         7     3     2
 3 C         9     7     7
 4 D         3     4     5
 5 E         2     8     8
 6 F         5     6     9
 7 G         6     5     4
 8 H         8     9    10
 9 I         4    10     3
10 J         1     1     1
```

- 查看线性回归模型结果：

```
res$reg |> summary()
```

```

Call:
lm(formula = RSR ~ Probit, data = rltTable)
```

```
Residuals:
      Min        1Q    Median        3Q       Max
-0.115792 -0.023458 -0.001329  0.051288  0.066768

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.60855    0.12516  -4.862  0.00282 **
Probit       0.22169    0.02329   9.520 7.66e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0653 on 6 degrees of freedom
Multiple R-squared:  0.9379,    Adjusted R-squared:  0.9276
F-statistic: 90.63 on 1 and 6 DF,  p-value: 7.662e-05
```

可以进一步提取回归模型的各种结果（略）。

9.2.4 更一般做法

- 指标权重采用熵权法，注意应该根据原始数据计算：

```
rlt = entropy_weight(df[-1], c("+","-","-"))
rlt$w
```

```
       x1        x2        x3
0.2945205 0.3631078 0.3423717
```

- 采用非整数秩，注意，此时负向指标正向化处理更适合采用”最大值减”而不是”取倒数”：

```
df2 = df |>
  mutate(across(x2:x3, \(x) max(x) - x))
res = rank_sum_ratio(df2, rlt$w, "non-int")
df2
```

```
# A tibble: 10 x 4
   ID       x1    x2    x3
   <chr> <dbl> <dbl> <dbl>
 1 A      99.5  21.2  7.48
 2 B      96.5  21.8  3.53
 3 C      99.4  37.6  8.03
 4 D      92.8  22.5  6.59
 5 E      91.7  46.1  8.62
```

```
 6 F      95.4  36.8  9.7
 7 G      96.1  31.7  6.2
 8 H      99.3  49.8  9.74
 9 I      94.8  58.6  3.76
10 J      84.8   0    0
```

- 查看结果表：

```
res$resultTable
```

```
# A tibble: 10 x 8
   ID      RSR  barR     f  sumf barRn Probit RSRfit
   <chr> <dbl> <dbl> <int> <int> <dbl>  <dbl>  <dbl>
 1 J     0.1       1     1     1 0.1     3.72  0.366
 2 B     0.544     2     1     2 0.2     4.16  0.456
 3 D     0.578     3     1     3 0.3     4.48  0.521
 4 G     0.676     4     1     4 0.4     4.75  0.576
 5 A     0.720     5     1     5 0.5     5     0.627
 6 I     0.725     6     1     6 0.6     5.25  0.679
 7 E     0.754     7     1     7 0.7     5.52  0.734
 8 F     0.802     8     1     8 0.8     5.84  0.798
 9 C     0.826     9     1     9 0.9     6.28  0.888
10 H     0.946    10     1    10 0.975   6.96  1.03
```

其它结果略。

第十章　模糊综合评价

模糊综合评价法（Fuzzy Comprehensive Evaluation, FCE）是一种基于模糊数学理论的综合评价方法，适用于处理具有模糊性和不确定性的多因素评价问题。该方法通过构建隶属函数，将定性指标转化为定量描述，有效刻画“亦此亦彼”的中间过渡状态，从而克服传统评价方法中“非黑即白”的二值判断局限，使评价结果更加贴近现实情况。

加载包：

```
library(tidyverse)
library(mathmodels)
```

10.1　模糊理论

10.1.1　模糊集与隶属度

用数学的眼光看世界，现象分为确定性现象、随机现象、模糊现象（如“今天天气有点冷”，“小伙子很高”等）。模糊理论的基本思想是，用属于程度代替属于或不属于，比如某人属于高个子的程度为 0.8）。

经典集合语言：

只有两种情况，要么 $x \in A$，要么 $x \notin A$，用特征函数 $\chi(\cdot) : A \to \{0, 1\}$ 表示：

$$\chi(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}$$

模糊集合语言：

用隶属度函数 $\mu_A(\cdot) : A \to [0, 1]$ 表示，它确定了 X 上的一个模糊集 A。$\mu_A(x)$ 越接近 1，表明 x 属于 A 的程度越大。

一般用 $A(x)$ 表示 x 对模糊集 A 的隶属度。

注：模糊集与隶属函数，是一一对应的，是同一个事物的两种表示。

例 1 考虑全集：周几 $= \{$周一, 周二, $\cdots$, 周日$\}$。

(1) 在经典集语义下说，子集：周末 $= \{$周六, 周日$\}$

- 周几是否属于周末是完全确定的：是或否
- 从非周末到周末，是突变过去的

(2) 从实际来说，周末并没有严格的界限，周五、周一甚至周四都有一部分也属于周末。这正好符合模糊集语义：

- 周几是否属于周末，可能是完全属于、部分属于、完全不属于，比如 周五属于周末的程度 $= 0.8$
- 从非周末到周末，是连续变化过去的

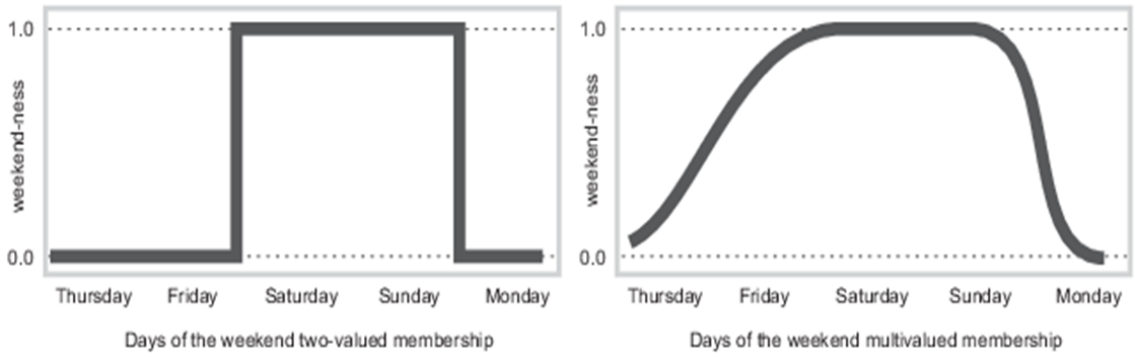


图 10.1: 严格周末与模糊周末

10.1.2 隶属函数

隶属函数 $\mu_A(x), x \in X$ 将 X 中每个元映射到 $[0,1]$ 上某个隶属度值。

例如，上面右图就是周几隶属于周末的程度的隶属函数曲线，根据该函数可以计算无论周几的任一时刻属于周末的程度是多少。

隶属函数是任意曲线，其形状可以人为定义，只要简单、方便、快速和高效。准确地确定隶属函数是定量刻化模糊概念的基础，也是利用模糊方法解决各种实际问题的关键。

常用隶属函数确定方法：模糊统计法，主观/专家经验法，二元对比排序法，拟合模糊分布法，最小模糊度法。

10.1.2.1 模糊统计法

模糊统计法，是基于评委评语/打分的数据统计，这是 tidyverse 最擅长的。

1. 从评委打分/评语数据到隶属矩阵 例如，模糊综合评价衣服的评语集为：“差”、“一般”、“好”。评委直接给出若干评价对象（衣服）各指标的评语，当然评委也可以是打分，根据分数阈值离散化到各评语。

这样的数据，不需要隶属函数，做分组统计就能得到隶属向量。

比如，某件衣服，对于色彩，经过统计发现：80% 的评委认为是 “好”，10% 的评委认为是” 一般”，10% 的评委认为是” 差”。

则该件衣服的色彩隶属于评语集每个评语：“好”、“一般”、“差” 的隶属度，即隶属向量为：$[0.8, 0.1, 0.1]$

同理，处理其他指标：做工、品牌、款式。

下面随机生成了一个数据作为演示，包含 10 个评委对 3 件衣服上述 4 个指标的打分：

```
set.seed(1)
df = matrix(sample(30:100, 120, replace = TRUE), ncol = 10) |>
  as_tibble(.name_repair = "universal") |>
  set_names(str_c(" 评委", 1:10)) |>
  mutate(ID = rep(1:3, each = 4),
         指标 = rep(c(" 颜色"," 做工"," 品牌"," 款式"), 3), .before = 1)
```

第一步，先宽变长变整洁，根据阈值，比如 $[60, 75]$ 离散化为评语：

```
df = df |>
  pivot_longer(-(1:2), names_to = " 评委", values_to = " 打分") |>
  mutate(评语 = case_when(打分 < 60 ~ " 差",
                          打分 < 75 ~ " 一般",
                          .default = " 好"),
         评语 = factor(评语, levels = c(" 差"," 一般"," 好")))
df
```

```
# A tibble: 120 x 5
      ID 指标  评委    打分 评语
   <int> <chr> <chr>  <int> <fct>
 1     1 颜色  评委1     97 好
 2     1 颜色  评委2     66 一般
 3     1 颜色  评委3     99 好
 4     1 颜色  评委4     31 差
 5     1 颜色  评委5     42 差
 6     1 颜色  评委6     80 好
 7     1 颜色  评委7     77 好
 8     1 颜色  评委8     66 一般
 9     1 颜色  评委9     36 差
10     1 颜色  评委10    46 差
# i 110 more rows
```

第二步，数据统计：分组汇总评语频数，分组计算评语百分比

```
df = df |>
  count(ID, 指标, 评语) |>
  mutate(p = n / sum(n), .by = c(ID, 指标))
df
```

```
# A tibble: 35 x 5
      ID 指标  评语      n     p
```

```
   <int> <chr> <fct> <int> <dbl>
 1     1 做工  差        2   0.2
 2     1 做工  一般      7   0.7
 3     1 做工  好        1   0.1
 4     1 品牌  差        6   0.6
 5     1 品牌  一般      3   0.3
 6     1 品牌  好        1   0.1
 7     1 款式  差        3   0.3
 8     1 款式  一般      3   0.3
 9     1 款式  好        4   0.4
10     1 颜色  差        4   0.4
# i 25 more rows
```

第三步，整理结果得到每件衣服的隶属矩阵

第二步已经计算出了所有隶属向量中的元素，只不过是堆在了一列 p 当中。每件衣服的取出来，每个评语的占一列，就是隶属矩阵。需要做的就是：按衣服切分 + 长变宽。

先把一件衣服做长变宽，写成函数：

```
f = function(data) {
  data |>
    pivot_wider(id_cols = 评语, names_from = 指标, values_from = p) |>
    select(-评语)
}
```

再按衣服切分数据，批量地做长变宽：

```
split(df, df$ID) |>
  map(f)
```

```
$`1`
# A tibble: 3 x 4
   做工  品牌  款式  颜色
  <dbl> <dbl> <dbl> <dbl>
1   0.2   0.6   0.3   0.4
2   0.7   0.3   0.3   0.2
3   0.1   0.1   0.4   0.4

$`2`
# A tibble: 3 x 4
   做工  品牌  款式  颜色
  <dbl> <dbl> <dbl> <dbl>
1   0.3   0.4   0.3  NA
```

```
2   0.3   0.3   0.4   0.5
3   0.4   0.3   0.3   0.5

$`3`
# A tibble: 3 x 4
   做工  品牌  款式  颜色
  <dbl> <dbl> <dbl> <dbl>
1   0.4   0.5   0.5   0.6
2   0.1   0.1   0.3   0.1
3   0.5   0.4   0.2   0.3
```

2. 从评委区间数据到隶属函数 这是模糊理论中理解隶属函数的经典示例，根据关于“青年人”年龄区间调查表，确定“青年人”的隶属函数。

该表记录了 129 位专家关于“青年人”年龄区间的调查结果。

```
df = readxl::read_excel("data/youth.xlsx",
                        col_names = FALSE) |>
  set_names(str_c("g", 1:13))
df
```

```
# A tibble: 10 x 13
   g1    g2    g3    g4    g5    g6    g7    g8    g9    g10   g11   g12   g13
   <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
 1 18-25 18-30 17-30 20-35 15-28 18-25 18-35 19-28 17-30 16-30 15-28 15-25 16-28
 2 18-30 18-25 18-28 17-30 15-30 18-30 18-35 15-25 17-25 17-30 18-35 18-25 18-30
 3 16-28 18-30 18-35 15-30 18-35 15-28 15-25 16-32 18-30 18-35 17-30 18-35 16-28
 4 20-30 16-30 18-35 18-35 18-29 17-28 18-35 18-35 18-25 18-30 16-28 17-27 15-26
 5 16-35 18-35 15-25 15-27 18-35 16-30 14-25 18-25 18-30 20-30 18-28 18-30 15-30
 6 18-28 18-25 16-25 20-30 18-35 18-30 18-30 16-28 17-25 16-30 18-30 15-25 18-35
 7 18-30 18-28 18-26 16-35 16-28 16-25 15-35 17-30 15-25 16-35 15-30 18-30 15-25
 8 16-30 16-30 15-28 15-36 15-25 17-28 18-30 16-25 18-30 17-25 18-29 17-29 15-30
 9 17-30 16-30 16-35 15-30 14-25 18-35 16-30 18-30 18-35 16-28 18-25 18-30 18-28
10 18-35 16-24 18-30 17-30 15-30 18-35 18-25 18-30 15-30 15-30 17-30 18-30 <NA>
```

第一步，数据处理：宽变长（忽略缺失）、分割列（转化数值）

```
df = df |>
  pivot_longer(everything(), names_to = "g", values_to = "x",
               values_drop_na = TRUE) |>
  separate(x, c("x1", "x2"), sep = "-", convert = TRUE) |>
  select(-g)
df
```

```
# A tibble: 129 x 2
      x1    x2
   <int> <int>
 1    18    25
 2    18    30
 3    17    30
 4    20    35
 5    15    28
 6    18    25
 7    18    35
 8    19    28
 9    17    30
10    16    30
# i 119 more rows
```

这样就得到了，每位专家认为的属于“青年人”的最小年龄和最大年龄。

第二步，构建隶属函数

“青年人”的隶属函数，就是包含年龄 x 的年龄区间（最小年龄和最大年龄之间）的占比：

```
mf = function(x) {
  summarise(df, p = mean(x1 <= x & x <= x2)) |>
    pull(p)
}
```

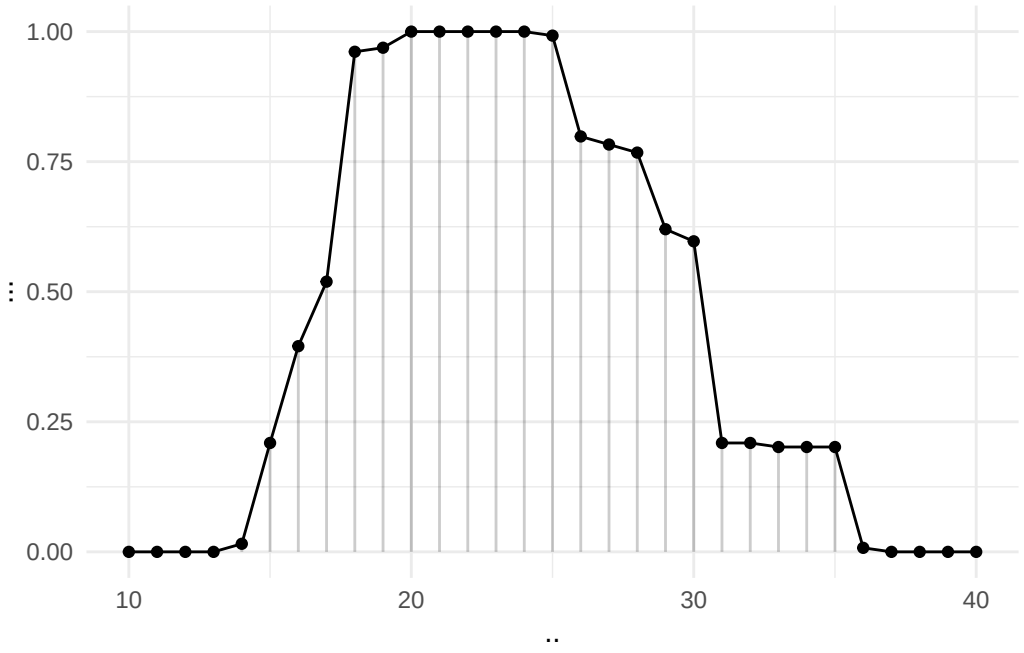
有一个或多个年龄，都能计算隶属度：

```
ages = c(25, 16.5, 20, 43)
map_dbl(ages, mf)
```

```
[1] 0.9922481 0.3953488 1.0000000 0.0000000
```

可视化该“青年人”隶属函数：

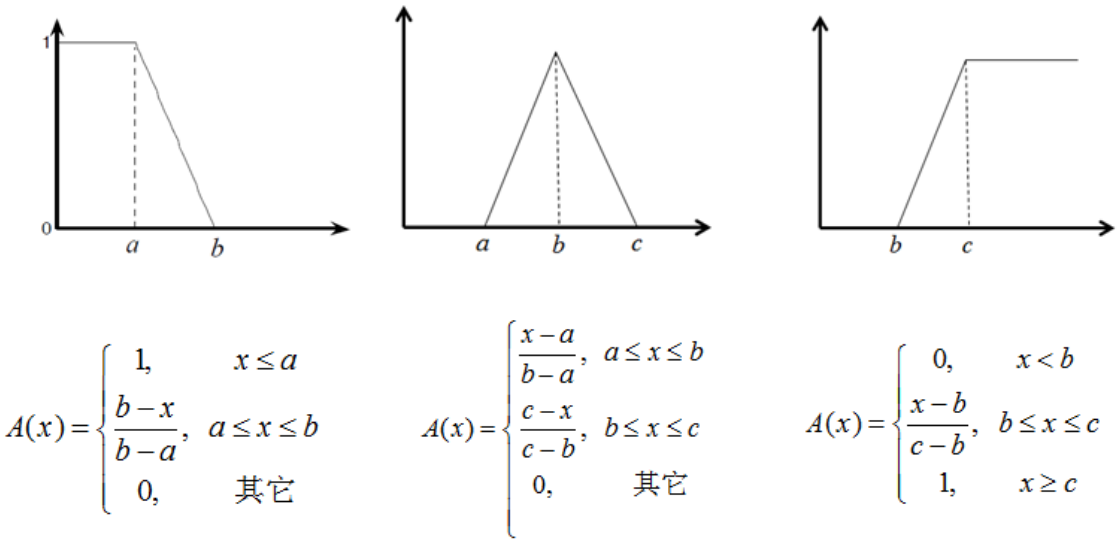
```
tibble(年龄 = 10:40, 隶属度 = map_dbl(年龄, mf)) |>
  ggplot(aes(年龄, 隶属度)) +
  geom_line() +
  geom_point() +
  geom_segment(aes(xend = 年龄, yend = 0), alpha = 0.2) +
  theme_minimal()
```



10.1.2.2 主观/专家经验法

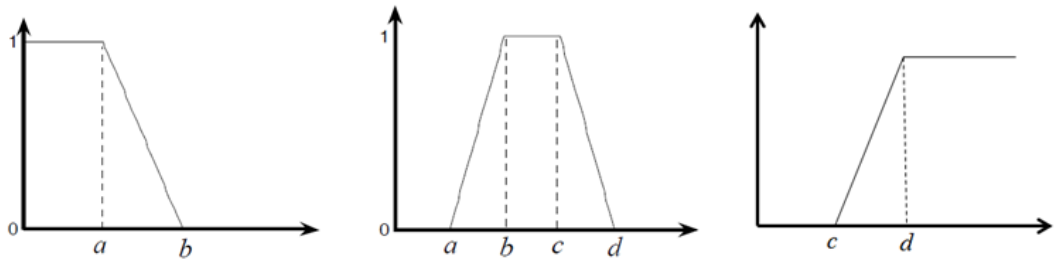
主观/专家经验法，是由领域专家根据经验或直观判断直接设定隶属函数的形式和参数，无需依赖数据统计或复杂计算。

比如，常用的三角隶属函数、梯形隶属函数、高斯隶属函数：



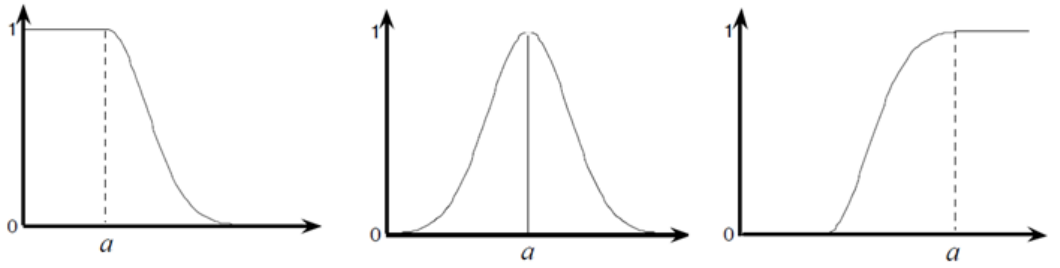
$$A(x)=\begin{cases}1, & x\le a\\ \dfrac{b-x}{b-a}, & a\le x\le b\\ 0, & \text{其它}\end{cases}\qquad A(x)=\begin{cases}\dfrac{x-a}{b-a}, & a\le x\le b\\ \dfrac{c-x}{c-b}, & b\le x\le c\\ 0, & \text{其它}\end{cases}\qquad A(x)=\begin{cases}0, & x<b\\ \dfrac{x-b}{c-b}, & b\le x\le c\\ 1, & x\ge c\end{cases}$$

图 10.2: 三角隶属函数



$$A(x)=\begin{cases}1, & x<a\\ \dfrac{b-x}{b-a}, & a\le x\le b\\ 0, & x>b\end{cases}\qquad A(x)=\begin{cases}\dfrac{x-a}{b-a}, & a\le x<b\\ 1, & b\le x<c\\ \dfrac{d-x}{d-c}, & c\le x\le d\\ 0, & x<a\text{或}x>d\end{cases}\qquad A(x)=\begin{cases}0, & x<c\\ \dfrac{x-c}{d-c}, & c\le x\le d\\ 1, & x>d\end{cases}$$

图 10.3: 梯形隶属函数



$$A(x)=\begin{cases}1, & x\le a\\ e^{-(\frac{x-a}{\sigma})^2}, & x>a\end{cases}\qquad A(x)=e^{-(\frac{x-a}{\sigma})^2},\quad -\infty<x<\infty\qquad A(x)=\begin{cases}e^{-(\frac{x-a}{\sigma})^2}, & x\le a\\ 1, & x>a\end{cases}$$

图 10.4: 高斯隶属函数

隶属度函数的理解：以梯形隶属度函数为例，比如某指标值最优区间是 $[5,10]$，则落在该区间内的该指标值，我们认为都是“最优”的，但 4.7 呢？我们不希望从 5 直接就到不是”最优”，而是让它们有一个自然的过度，比如取 $[4,5]$ 和 $[10,11]$ 线性过度到不是”最优”，那么关于 4.7 这个点，有 $\dfrac{4.7-4}{5-4}=0.7$，即 4.7 隶属于最优区间 $[5,10]$ 的程度是 0.7。

另外，还有双高斯隶属函数、广义贝尔隶属函数、Sigmoid 隶属函数族（单 Sigmoid、两 Sigmoid 之差、两 Sigmoid 之积）、多项式隶属函数族（分别形如 Z,π,S）。

`mathmodels` 包提供了上述所有隶属函数的实现，以及可视化隶属函数的函数：

```
tri_mf(x, params)
trap_mf(x, params)
```

```
gauss_mf(x, params)
gbell_mf(x, params)
gauss2mf(x, params)
sigmoid_mf(x, params)
dsigmoid_mf(x, params)
psigmoid_mf(x, params)
z_mf(x, params)
pi_mf(x, params)
s_mf(x, params)
plot_mf(mf, xlim = c(0, 10), main = NULL)
```

其中，x 为数值向量，params 为各隶属函数相应参数值向量，返回与 x 同样长度的隶属度值。

以梯形隶属函数为例，包含 4 个参数，比如取 1,5,7,8，计算 1:10 的隶属度值：

```
x = 1:10
trap_mf(x, params = c(1, 5, 7, 8))
```

```
 [1] 0.00 0.25 0.50 0.75 1.00 1.00 1.00 0.00 0.00 0.00
```

可视化该梯形隶属函数：

```
f = \(x) trap_mf(x, c(1, 5, 7, 8))
plot_mf(f, xlim = c(0,10), main = " 梯形隶属函数")
```

10.1.3 compute_mf() 函数

将单个指标值转化为属于各评语的隶属向量，是模糊综合评价计算隶属矩阵最核心的、大量重复使用的计算步。而非常常见且合理的做法，就是三角 + 梯形隶属函数法，或叫做分段线性隶属函数法。

mathmodels 包提供了 compute_mf() 函数，实现基于三角隶属（两端是半梯形）将单个指标值转化为属于各评语的隶属向量，基本语法为：

```
compute_mf(x, thresholds)
```

- x 为单个指标值；
- thresholds 为指标值划分到各评语的中间界限构成的向量（长度至少为 2）
- 返回 mv 为隶属向量，对应单个评价对象的单个指标值。

用法示例

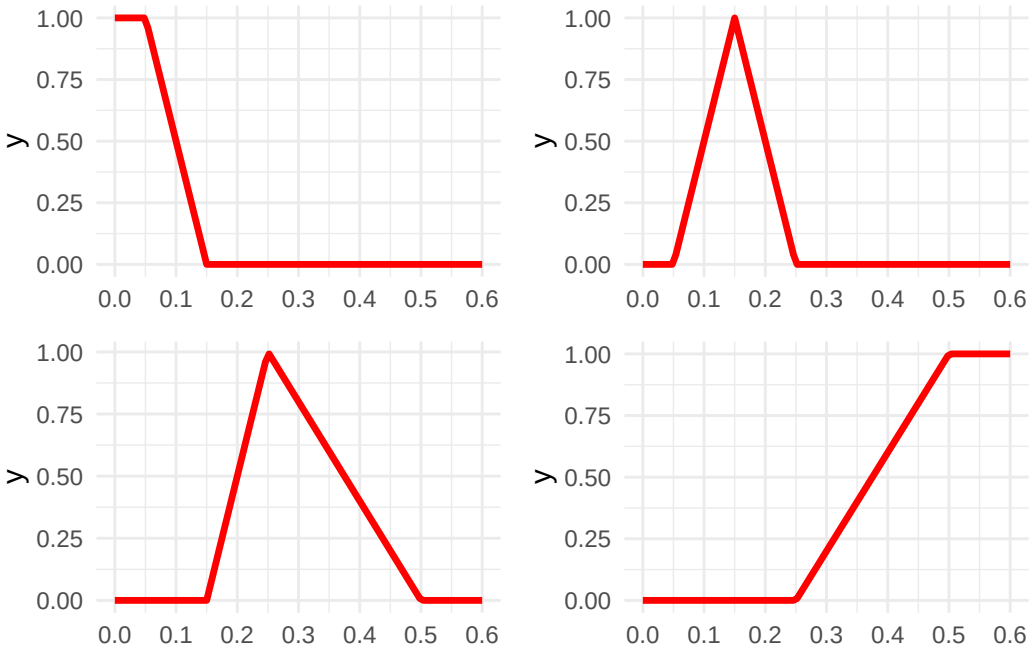
举个例子，大气污染物评价，其中 SO2 的四个等级阈值为：

污染物	一级	二级	三级	四级
SO2	0.05	0.15	0.25	0.50

这表示：0.05 是一级（100% 隶属），0.15 是二级，0.25 是三级，0.50 是四级；中间区间线性从 1 减到 0，比如 $[0.05, 0.15]$ 一级的隶属度是从 1 线性降到 0，二级隶属是从 0 线性升到 1，其他类似。

结合函数图形来理解更加直观。为了便于可视化演示，顺便提供了 compute_mf_funs 函数返回所有隶属函数，再用 plot_mf 函数做可视化：

```
th = c(0.05, 0.15, 0.25, 0.50)
mfs = compute_mf_funs(th)
plots = lapply(mfs, \(x) plot_mf(x, xlim = c(0, 0.6)))
gridExtra::grid.arrange(grobs = plots, nrow = 2)
```



分别是一级、二级、三级、四级的隶属函数，形状依次为右半梯形、三角形、三角形、左半梯形。以右上“二级”隶属函数为例，0.15 处为 1，区间 [0.05, 0.15] 上线性增长到 1。

比如，某天的 SO2 含量为 0.07，转化为隶属向量：

```
compute_mf(0.07, th)          # 计算隶属向量
```

```
[1] 0.8 0.2 0.0 0.0
```

这 4 个值，就是将 0.07 分别代入上述 4 个隶属函数得到的函数值构成。这表明，0.07 隶属于“一级”的程度是 0.8，隶属于“二级”的程度是 0.2，隶属于后两级的程度都是 0。

10.1.4 模糊运算

10.1.4.1 模糊逻辑运算

经典集特征函数值只取 0 和 1，标准布尔逻辑运算：与、或、非

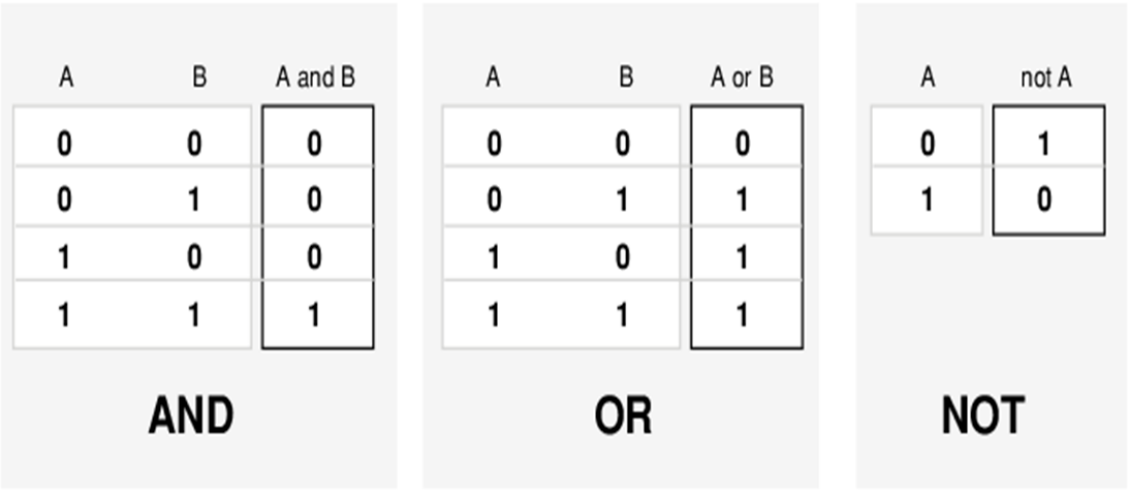


图 10.5: 布尔运算示意图

换个写法：与就是取 min，或就是取 max，非就是做 $1-x$

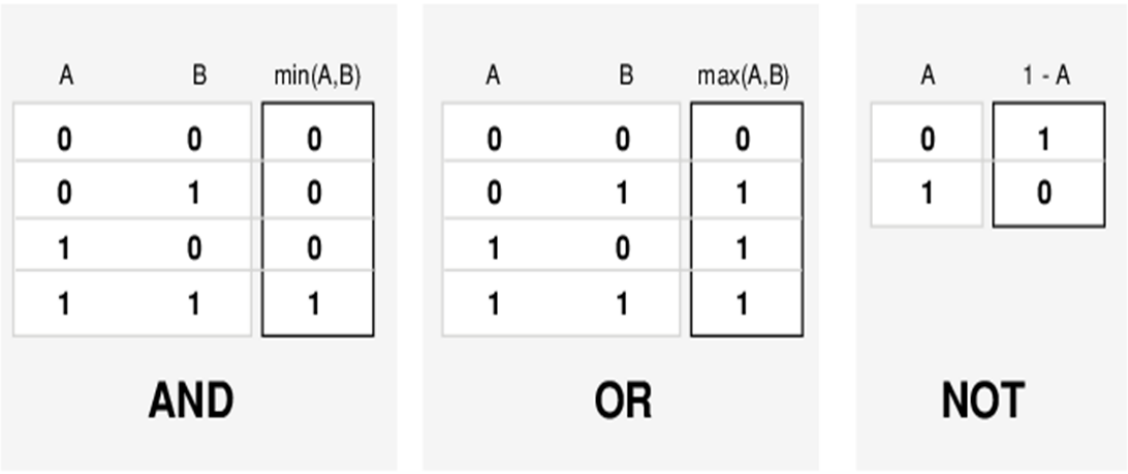


图 10.6: 布尔运算换个写法

从经典集的布尔运算到模糊集的模糊运算，模糊集的隶属函数取值是 $[0,1]$。

模糊集与隶属函数（曲线）一一对应，所以，很自然地，模糊集的逻辑运算，就等同于隶属函数（曲线）的逻辑运算（取 min、取 max、做 $1-x$）。

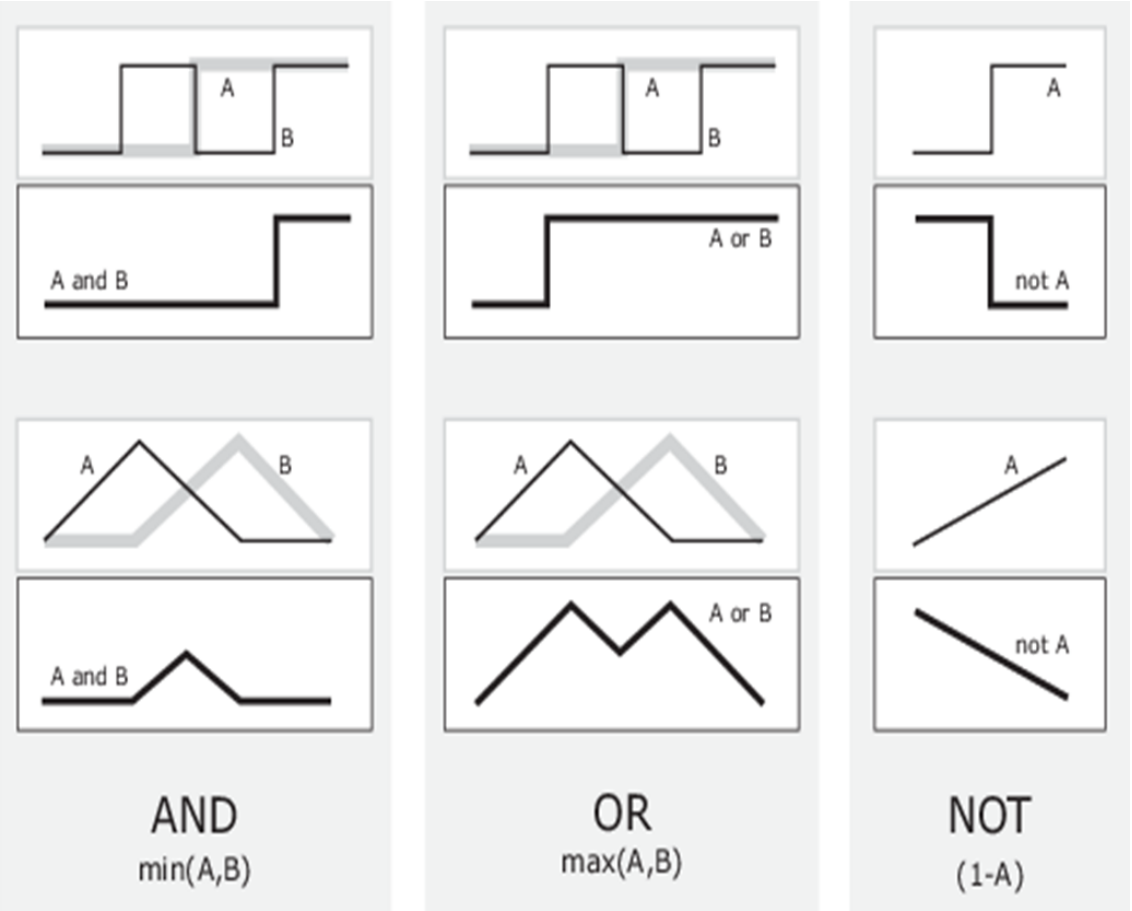


图 10.7: 从布尔运算到模糊运算

10.1.4.2 模糊合成

设 $R=(r_{ij})_{m\times n}$ 为矩阵，若满足 $0\le r_{ij}\le 1$，称为**模糊矩阵**，若 r_{ij} 只取 0 或 1 时，称为**布尔矩阵**。

模糊综合评价涉及模糊变换，即将模糊评价矩阵作用到向量，得到新向量：

$$B_{m\times 1}=R_{m\times n}\circ A_{n\times 1}$$

其中，$\circ$ 为模糊合成算子，记 $A=[a_1,\cdots,a_m]^T$, $B=[b_1,\cdots,b_m]^T$。

模糊变换根据目的不同，可以选择不同的模糊合成算子：

(1) 取小取大，主因素决定型

$$b_j=\max_i\big\{\min\{a_i,r_{ij}\}\big\},\quad j=1,\cdots,m$$

通常用的算子，其评判结果只取决于在总评价中起主要作用的那个因素，其余因素均不影响评判结果，比较适用于单项评判最优就能作为综合评判最优的情况。

(2) 乘积最大，主因素突出型

$$b_j = \max_i\{a_i \cdot r_{ij}\}, \quad j = 1, \cdots, m$$

与取小取大相近，但更精细些，不仅突出了主要因素，也兼顾了其他因素。此模型适用于模型失效（不可区别），需要 “加细” 的情况。

(3) 乘加，加权平均型

$$b_j = \sum_{i=1}^{n} a_i \cdot r_{ij}, \quad j = 1, \cdots, m$$

该算子依权重的大小对所有因素均衡兼顾，比较适用于求总和最大的情形。

(4) 取小上界和型

$$b_j = \min\left\{1, \sum_{i=1}^{n} \min\{a_i, r_{ij}\}\right\}, \quad j = 1, \cdots, m$$

在使用此算子时，需要注意：各个 a_i 不能取得偏大，否则可能出现 b_j 均等于 1 的情形；各个 a_i 也不能取得太小，否则可能出现 b_j 均等于各个 a_i 之和的情形，这将使单因素评判的有关信息丢失。

(5) 均衡平均型

$$b_j = \sum_{i=1}^{n} \left(a_i \wedge \frac{r_{ij}}{r_0}\right), \quad j = 1, \cdots, m$$

$r_0 = \sum_{k=1}^{m} r_{kj}$，该算子实际上先对模糊评价矩阵 R 中的列向量做了归一化处理, 适用于 R 中元素偏大或偏小的情形。

`mathmodels` 包提供了 `fuzzy_eval()` 函数实现模糊合成，基本语法为：

```
fuzzy_eval(w, R, type)
```

- `w` 为各因素的权重向量，`R` 为模糊评价矩阵，`type` 可选上述 5 种模糊合成算子：$1 \sim 5$。

10.2 模糊综合评价算法步骤

(1) 确定因素集及权重向量 设某评价对象的评价因素有 n 个，记作 $U = \{u_1, \cdots, u_n\}$，称为因素集。由于各种因素所处地位和作用的不同，考虑用权重向量来衡量，实际中该权重向量可以借助主客观赋权法来得到。

例如，某人要购买一件衣服，她要考虑 4 个因素：

$$u_1 = \text{色彩}, \quad u_2 = \text{做工}, \quad u_3 = \text{品牌}, \quad u_4 = \text{款式}$$

4 个因素在评价过程中的权重向量为 $w = [0.3, 0.3, 0.3, 0.1]^T$。

(2) 确定评语集 设所有可能的评语有 m 个，记为 $V=\{v_1,,v_m\}$，称为评语集。

例如，对衣服的评语集为

$$v_1=\text{好},\quad v_2=\text{一般},\quad v_3=\text{差}$$

(3) 建立模糊评价矩阵 先对该事物的每个因素隶属于各个评语的程度进行评价（评委打分或经验隶属函数）。

例如，某件衣服，对于" 色彩"，80% 的评委认为是 "好"，10% 的评委认为是" 一般"，10% 的评委认为是" 差"。

则该件衣服因素 1："色彩" 隶属于评语集每个评语："好"、"一般"、"差" 的隶属度为：$r_1=[0.8,0.1,0.1]^T$；

同样，因素 2："做工" 隶属于每个评语："好"、"一般"、"差" 的隶属度为：$r_2=[0.7,0.2,0.1]^T$；

因素 3："品牌" 隶属于每个评语："好"、"一般"、"差" 的隶属度为：$r_3=[0.6,0.2,0.2]^T$；

因素 4："款式" 隶属于每个评语："好"、"一般"、"差" 的隶属度为：$r_4=[0.7,0.1,0.2]^T$。

于是，得到模糊评价矩阵：

$$R=[r_1,r_2,r_3,r_4]=\begin{bmatrix}0.8 & 0.7 & 0.6 & 0.7\\ 0.1 & 0.2 & 0.2 & 0.1\\ 0.1 & 0.1 & 0.2 & 0.2\end{bmatrix}$$

(4) 做模糊综合合成 基于合适的模糊合成算子，比如选 3 加权平均型，做模糊合成得到模糊评价向量：$\mu=R\circ w$，一般对 μ 进行归一化处理。

这里的 R 是分别的、单独的评价（隶属程度），再综合各因素占的权重 w，所以叫做模糊综合评价。

```
w = c(0.3, 0.3, 0.3, 0.1)
R = matrix(c(0.8, 0.7, 0.6, 0.7,
             0.1, 0.2, 0.2, 0.1,
             0.1, 0.1, 0.2, 0.2), nrow = 3, byrow = TRUE)
mu = fuzzy_eval(w, R, 3)
mu
```

```
[1] 0.70 0.16 0.14
```

(5) 去模糊化得到最终评价 模糊评价向量 μ 是这件衣服对各评语："好"、"一般"、"差" 的隶属度，要得到最终评语或评分，还需要做一步去模糊化。

`mathmodels` 包，提供了 `defuzzify()` 函数，实现对模糊评价向量去模糊化，基本语法为：

```
defuzzify(mu, scores, method = "weighted_average")
```

- `mu` 为模糊评价向量；
- `scores` 为各评语的量化分数；

- `method` 选择去模糊化方法，可选 `"max_membership"`：最大隶属度法（隶属度最大值对应评语的量化分数）、`"weighted_average"`：加权平均法（默认，隶属度值与各评语量化分数做加权求和）、`"centroid"`：重心法。

注：对于归一化的模糊评价隶属向量，加权平均法与重心法结果是相同的。

```
scores = c(90, 75, 50)
defuzzify(mu, scores, method = "max_membership")        # 最大隶属度法
```

```
[1] 90
```

```
defuzzify(mu, scores)                                   # 加权平均法
```

```
[1] 82
```

10.3 案例：耕作方案的模糊综合评价

某平原产粮区进行耕作制度改革，制定了甲（三种三收）、乙（两茬平作）、丙（两年三熟）3 种方案。主要评价指标选取 5 项：粮食亩产量、农产品质量、每亩用工量、每亩纯收入、生态环境影响。根据当地实际情况，这 5 个因素的权重分别为 $0.2, 0.1, 0.15, 0.3, 0.25$，其评价等级如表 1 所示：

表 10.2: 耕作方案评价指标数据的等级划分

分数	亩产量/kg	产品质量/级	亩用工量/工日	亩纯收入/元	生态环境影响/级
优	550 以上	1	20 以下	130 以上	1
良	450-550	2	20-40	90-130	2
中	350-450	3	40-60	50-90	3
差	350 以下	4	60 以上	50 以下	4

经过调查并应用各种参数进行计算预测，发现 3 种方案的 5 项指标可达到表 2 中的数值：

表 10.3: 三种耕作方案的指标数据

方案	亩产量/kg	产品质量/级	亩用工量/工日	亩纯收入/元	生态环境影响/级
甲	592.5	3	55	72	4
乙	529	2	38	105	3
丙	412	1	32	85	2

问究竟应该选择哪种方案？

模糊综合评价，超级简洁、可以随便替换成自己数据的通用流程如下：

- `compute_mf` 函数是根据指标阈值和单个指标值，计算**隶属向量**；

- 批量调用 compute_mf 函数，对一个评价对象（耕作方案）的所有指标值和相应指标阈值，都算一遍再合并，就计算出**隶属矩阵**；
- 隶属矩阵和指标权重向量做模糊综合合成（fuzzy_eval 函数），就得到模糊综合评价结果向量，再去模糊化（defuzzify 函数）到最终评价分数；
- 再对每个评价对象重复上述过程。

下面具体求解。

10.3.1 准备评价对象数据（表 2）

```
df = tibble(product = c(592.5, 529, 412), quality = 3:1,
            labor = c(55, 38, 32),
            income = c(72, 105, 85), ecology = 4:2)
df
```

```
# A tibble: 3 x 5
  product quality labor income ecology
    <dbl>   <int> <dbl>  <dbl>   <int>
1    592.       3    55     72       4
2    529        2    38    105       3
3    412        1    32     85       2
```

数据预处理

注意，产品质量、亩用工量、生态环境影响是负向指标，需要转化为正向。应该从固定不变的阈值考虑（而不是可变的具体方案指标值），采用最大值 + 平移值做差方式。比如，1、2、3、4 要变成 4、3、2、1，需要的变换是 $5-x$；而 20、40、60 要变成 60、40、20，需要的变换是 $80-x$

```
df = df |>
  mutate(across(c(2,5), \(x) 5 - x), labor = 80 - labor)
df
```

```
# A tibble: 3 x 5
  product quality labor income ecology
    <dbl>   <dbl> <dbl>  <dbl>   <dbl>
1    592.       2    25     72       1
2    529        3    42    105       2
3    412        4    48     85       3
```

10.3.2 准备阈值数据（从表 1 提炼，用于计算隶属向量）

评语集是分数列，对应到优良中差：3（优）、2（良）、1（中）、0（差）。注意，共有 4 个评语，需要 4 个阈值（同示例）。产品质量和生态环境是 $1,2,3,4$（可以直接用），其他三个，比如亩产量只提供了 3 个阈值：

350, 450, 550，显然在前面补一个 0 是合理的。

```
th_mat = tibble(
  th1 = c(0, 350, 450, 550),            # 亩产量
  th2 = 1:4,                            # 产品质量
  th3 = seq(0,60,20),                   # 亩用工量
  th4 = c(0, 50, 90, 130),              # 亩收入
  th5 = 1:4)                            # 生态环境影响
```

10.3.3 批量模糊综合评价

本例只有 3 个评价对象，当然可以一个一个地分别计算。

为了通用于任意多个评价对象，把**对一个评价对象做模糊综合评价**的写成函数：

```
eval_one = function(x, type = 3, method = "max_membership") {
  # x 为一个评价对象 (提取 df 的某一行, 转化为向量)
  R = map2_dfc(x, th_mat, compute_mf)
  mu = fuzzy_eval(w, R, type)                # 默认对结果归一化
  s = defuzzify(mu, scores, method)
  list(s = s, mu = mu, R = R)
}
```

其中，type 选择模糊合成的方法，可选 1：取小取大（主因素决定）、2：乘积最大（主因素突出）、3：乘加（加权平均）、4：取小上界和、5：均衡平均。method 选择去模糊化方法，可选 "max_membership"：最大隶属度法、"weighted_average"：加权平均法、"centroid"：重心法。

注：对于归一化的模糊评价隶属向量，加权平均法与重心法结果是相同的。

准备共用数据：

```
w = c(0.2, 0.1, 0.15, 0.3, 0.25)  # 各指标权重, 可另由赋权法计算
scores = c(30, 60, 75, 90)        # 优良中差的量化分数, 用于去模糊化
```

方法 1：按默认的乘加法模糊合成，按最大隶属度法去模糊化。

在 df 上逐行迭代即可：

```
res_m = apply(df, 1, eval_one)
```

查看 3 种方案的模糊综合评价结果向量：

```
map(res_m, "mu")
```

```
[[1]]
[1] 0.2500 0.3475 0.2025 0.2000

[[2]]
```

```
[1] 0.0000 0.2500 0.4645 0.2855

[[3]]
[1] 0.0000 0.1135 0.7265 0.1600
```

查看 3 种方案的最终得分（上述模糊向量去模糊化），最大隶属度法即最大值所属评语的量化分数：

```
map_dbl(res_m, "s")
```

```
[1] 60 75 75
```

- 查看 3 种方案的隶属矩阵，由隶属向量构成：

```
map(res_m, "R")
```

```
[[1]]
# A tibble: 4 x 5
  product quality labor income ecology
    <dbl>   <dbl> <dbl>  <dbl>   <dbl>
1       0       0  0      0          1
2       0       1  0.75   0.45       0
3       0       0  0.25   0.55       0
4       1       0  0      0          0

[[2]]
# A tibble: 4 x 5
  product quality labor income ecology
    <dbl>   <dbl> <dbl>  <dbl>   <dbl>
1    0          0   0    0          0
2    0          0   0    0          1
3    0.21       1   0.9  0.625      0
4    0.79       0   0.1  0.375      0

[[3]]
# A tibble: 4 x 5
  product quality labor income ecology
    <dbl>   <dbl> <dbl>  <dbl>   <dbl>
1    0          0   0    0          0
2    0.38       0   0    0.125      0
3    0.62       0   0.6  0.875      1
4    0          1   0.4  0          0
```

方法 2：按默认的**乘加法**模糊合成，按加权平均法去模糊化。

对 `df` 逐行迭代，只查看 3 种方案的最终评分：

```
res_w = apply(df, 1, \(x) eval_one(x, method = "centroid"))
map_dbl(res_w, "s")
```

```
[1] 61.5375 75.5325 75.6975
```

第十一章　数据包络分析

数据包络分析（DEA），是一种基于线性规划的非参数评估方法。它通过构建相对效率评价模型，对多个具有多投入、多产出特征的决策单元（DMU）进行绩效评估和效率排序。DEA 的核心优势在于，它能够根据每个决策单元自身的投入和产出数据，自动优化其权重配置。这种方法无需事先设定固定的权重，从而能够更客观、真实地识别出对每个决策单元最有利的评价标准，准确反映其个体特征和实际运行效率。

DEA 已被广泛应用于生产管理、行政绩效评估等诸多领域。它不仅可以用于评估不同方案的相对有效性，例如投资项目的效益比较，还可以在决策前进行效果预测，或在政策实施后评估其成效。通过 DEA 的辅助，决策者能够更科学地制定策略，提升管理和决策的精准性。

DEA 优点：

(1) 多指标处理能力：DEA 擅长处理多输入和多输出指标的综合评价问题，适用于复杂系统的效率评估。
(2) 无需无量纲化：DEA 不对数据进行无量纲化处理，因为其效率指标的计算与输入输出数据的量纲无关。
(3) 客观性强：DEA 无需事先设定权重，而是根据实际数据计算最优权重，减少了主观因素的干扰，提高了评估的客观性。
(4) 灵活的关系建模：DEA 假定输入和输出之间存在某种联系，但不需要明确表达这种关系的具体形式，从而增加了模型的灵活性。

DEA 缺点：

(1) 数据依赖性：DEA 的效率评估结果依赖于所收集的数据，最优效率仅在当前样本范围内有效。
(2) 技术有效单元的局限性：DEA 无法对技术有效单元进行进一步比较，且未考虑系统中的随机因素，可能导致在存在异常点时评估结果失真。

11.1 DEA 相关概念

设有 n 个部门或决策单元（DMU），每个决策单元有 m 个输入变量和 q 个输出变量：

	m 个投入				q 个产出			
DMU/指标	x_1	x_2	$\cdots$	x_m	y_1	y_2	$\cdots$	y_q
1	x_{11}	x_{12}	$\cdots$	x_{1m}	y_{11}	y_{12}	$\cdots$	y_{1q}
2	x_{21}	x_{22}	$\cdots$	x_{2m}	y_{21}	y_{22}	$\cdots$	y_{2q}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
n	x_{n1}	x_{n2}	$\cdots$	x_{nm}	y_{n1}	y_{n2}	$\cdots$	y_{nq}
	$\uparrow$	$\uparrow$	$\cdots$	$\uparrow$	$\uparrow$	$\uparrow$	$\cdots$	$\uparrow$
权重	v_1	v_2	$\cdots$	v_m	u_1	u_2	$\cdots$	u_m

用 $i = 1, \cdots, n$ 表示决策单元的索引；$j = 1, \cdots, m$ 表示投入指标的索引；$r = 1, \cdots, q$ 表示产出指标的索引。

改用向量形式表示，记

$$X_i = [x_{i1}, x_{i2}, \cdots, x_{im}], \quad Y_i = [y_{i1}, y_{i2}, \cdots, y_{iq}], \quad i = 1, \cdots, n$$

$$v = [\nu_1, \cdots, \nu_m], \quad u = [u_1, \cdots, u_q]$$

则 X_i, Y_i 分别为第 i 个决策单元的输入向量、输出向量；v, u 分别为输入权重、输出权重。

DEA 评价的是技术效率，是指一个决策单元的生产过程达到本行业技术水平的程度。一般来说，技术效率可以使用产出和投入的比例衡量，但这种衡量方式一般仅适用于单投入单产出的情形。对于 m 个投入和 q 个产出，则第 k 个决策单元的技术效率，可以用加权方式确定其综合的投入产出来刻画：

$$h_k = \frac{\sum_{r=1}^{q} u_r y_{kr}}{\sum_{j=1}^{m} \nu_j x_{kj}}$$

关于投入与产出导向：

在径向 DEA 中，无效率往往是通过投入和产出的等比例变化定义的，因此既可以在给定投入的情况下最大化产出（产出导向），也可以在给定产出的情况下最小化投入（投入导向）。

对于不同的规模收益假设，不同导向的效率分析结果可能存在一定差异。对于规模收益不变的模型（CRS），两种导向的效率结果是一样的；而对于可变规模收益模型（VRS）中，二者是不同的。

在实践中，投入和产出导向的选择没有明确的要求，实际选择时最好是根据具体生产活动的实际，看是投入倾向于固定不变还是产出倾向于固定不变。

DEA 模型的编程实现：

R 语言有 deaR 包等可以实现 DEA 模型。

手动编程实现的话，因为本质上就是求解线性规划问题，在模型公式确定后，其编程过程可遵循如下步骤：

(1) 确定参数列向量；

(2) 将模型表示为线性规划标准形式；
(3) 改成用矩阵语言表示，梳理出各矩阵、向量；
(4) 调用线性规划求解器进行求解。

11.2 常用 DEA 模型

11.2.1 CCR 模型（规模收益不变假设下的径向 DEA 模型）

(1) 投入导向的 CCR 模型

在给定投入的条件下最大化产出。将前面加权方式的投入产出技术效率，再将其范围限制为 $[0,1]$，则得到投入导向的 CCR 模型：对每个决策单元 k，

$$\begin{aligned} & \max \frac{\sum_{r=1}^{q} u_r y_{kr}}{\sum_{j=1}^{m} v_j x_{kj}} \\ \mathbf{s.t.}\ & \frac{\sum_{r=1}^{q} u_r y_{kr}}{\sum_{j=1}^{m} v_j x_{kj}} \leq 1 \\ & v_j \geq 0, u_r \geq 0, \quad j=1,\cdots,m;\ r=1,\cdots q \end{aligned}$$

通过 Charnes-Cooper 变换线性化转化为线性规划：对每个决策单元 k，

$$\begin{aligned} & \max \sum_{r=1}^{q} \mu_r y_{kr} \\ \mathbf{s.t.}\ & \sum_{r=1}^{q} \mu_r y_{kr} - \sum_{j=1}^{m} \nu_j x_{kj} \leq 0 \\ & \sum_{j=1}^{m} \nu_j x_{kj} = 1 \\ & \mu_r \geq 0, \nu_j \geq 0, \quad j=1,\cdots,m;\ r=1,\cdots q \end{aligned}$$

上述问题的对偶形式为（对偶模型的决策变量中包含效率值）：对每个决策单元 k

$$\begin{aligned} & \min \theta \\ \text{s.t.}\ & \sum_{i=1}^{n} \lambda_i x_{ij} \leq \theta x_{kj} \\ & \sum_{i=1}^{n} \lambda_i y_{ir} \geq y_{kr} \\ & \lambda_i \geq 0, \quad j=1,\cdots,m;\ r=1,\cdots q \end{aligned}$$

该对偶模型中，$\lambda_i, i=1,\cdots,n$ 表示 DMU 的线性组合系数，参数 θ 的最优解 θ^* 即为效率值，其范围落在 $[0,1]$。

该模型的含义相当于用加权方法构造出一个不存在的 DMU，其投入不大于待评价的 DMU，产出不小于待评价的 DMU，即

$$x=\sum_{i=1}^{n}\lambda_i x_{ij},\quad y=\sum_{i=1}^{n}\lambda_i x_{ir}$$

为了便于求解，进一步改写为矩阵形式：

$$\begin{aligned}&\min\left[0,\cdots,0,1\right]\left[\lambda_1,\cdots,\lambda_n,\theta\right]^T\\ \mathbf{s.t.}\ &\begin{bmatrix}x_{11}&x_{21}&\cdots&x_{n1}&-x_{i1}\\x_{12}&x_{22}&\cdots&x_{n2}&-x_{i2}\\\vdots&\vdots&\ddots&\vdots&\vdots\\x_{1m}&x_{2m}&\cdots&x_{nm}&-x_{im}\end{bmatrix}\begin{bmatrix}\lambda_1\\\vdots\\\lambda_n\\\theta\end{bmatrix}\leqslant\begin{bmatrix}0\\0\\\vdots\\0\end{bmatrix}\\&\begin{bmatrix}-y_{11}&-y_{21}&\cdots&-y_{n1}&0\\-y_{12}&-y_{22}&\cdots&-y_{n2}&0\\\vdots&\vdots&\ddots&\vdots&\vdots\\-y_{1q}&-y_{2q}&\cdots&-y_{nq}&0\end{bmatrix}\begin{bmatrix}\lambda_1\\\vdots\\\lambda_n\\\theta\end{bmatrix}\leqslant\begin{bmatrix}-y_{i1}\\-y_{i2}\\\vdots\\-y_{iq}\end{bmatrix}\end{aligned}$$

注意，模型的决策变量向量为 $[\lambda_1,\cdots,\lambda_n,\theta]^T$；两个系数矩阵的主体部分都是原始指标数据的转置，约束条件可以对应地按分块矩阵合并来写：

$$\text{s.t.}\quad\begin{bmatrix}X\\-Y\end{bmatrix}_{2\times1}\lambda_{1\times1}\leqslant\begin{bmatrix}0\\-y\end{bmatrix}_{2\times1}$$

(2) 产出导向的 CCR 模型

在给定产出条件下最小化投入，具体推导过程略，只列出最终的对偶模型：对每个决策单元 k，

$$\begin{aligned}&\max\phi\\ \mathbf{s.t.}\ &\sum_{i=1}^{n}\lambda_i x_{ij}\leq x_{kj}\\&\sum_{i=1}^{n}\lambda_i y_{ir}\geq\phi y_{kr}\\&\lambda_i\geq0,\quad j=1,\cdots,m;\ r=1,\cdots q\end{aligned}$$

该模型的效率为 $1/\phi$。

11.2.2 BCC 模型（规模收益可变假设下的径向 DEA 模型）

在 DEA 模型中，对规模收益（RTS）的设定决定了前沿的形状，CCR 模型是假设规模收益不变（CRS），即模型中的 λ 满足 $\lambda>0$，此时生产可能集（DEA 技术集）为以 OB 射线为前沿面的集合：

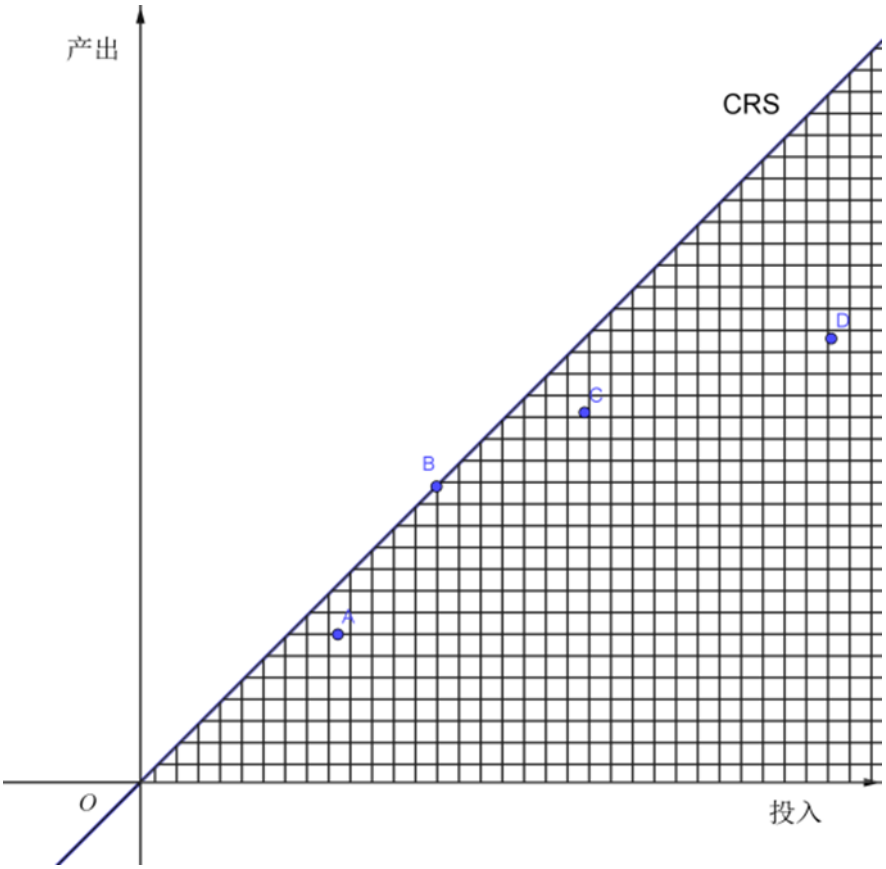


图 11.1: CRS 假设下的生产可能集

但在实际生产过程中，生产技术的规模收益并非 CRS，若采用 CRS 假设（CCR 模型），得出的技术效率并非完全是纯技术效率，而是包含了规模效率成分的综合效率。

一般来说，生产技术的规模收益要先后经历规模收益递增（IRS）、规模收益不变（CRS）、规模收益递减（DRS）三个阶段。

如果无法确定研究样本处于哪个阶段，则评价技术效率时应选择规模收益可变（VRS）模型，即模型中的 λ 满足 $\sum \lambda = 1$，此时生产可能集（DEA 技术集）为以线段 ABCD 以及 AD 往坐标轴的垂线为前沿（凸组合）：

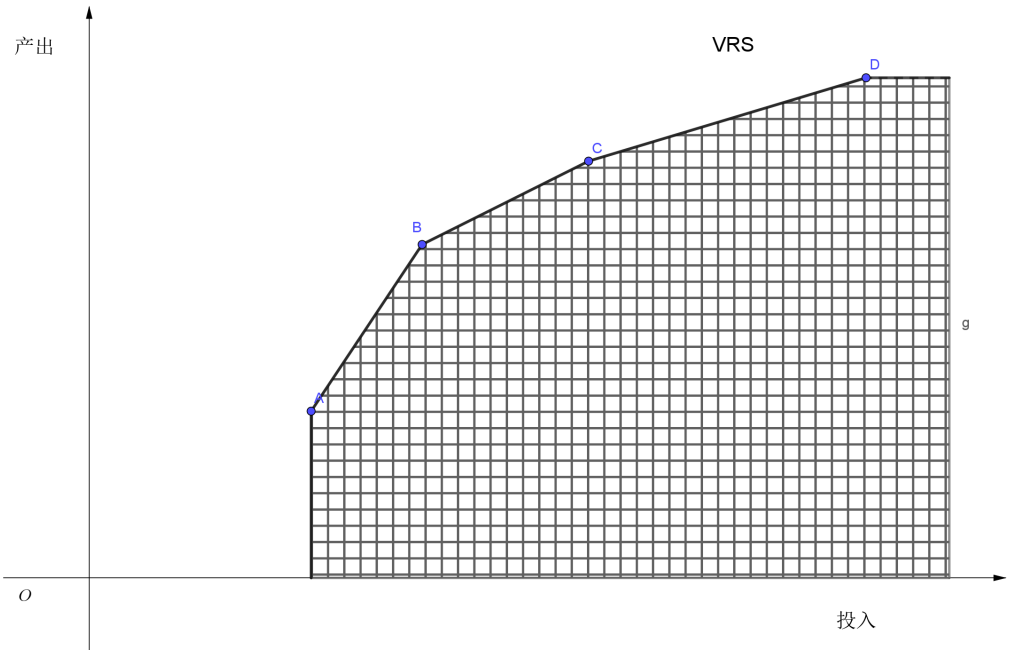


图 11.2: VRS 假设下生产可能集

VRS 模型得出的技术效率是纯技术效率。

BCC 模型即规模收益可变（VRS）假设下的径向 DEA 模型，与 CCR 模型的区别就是增加了等式约束 $\sum \lambda = 1$。

投入导向的 BCC（对偶）模型：对每个决策单元 k，

$$
\begin{aligned}
& \min \theta \\
\textbf{s.t.} \quad & \sum_{i=1}^{n} \lambda_i x_{ij} \leq \theta x_{kj} \\
& \sum_{i=1}^{n} \lambda_i y_{ir} \geq y_{kr} \\
& \sum_{i=1}^{n} \lambda_i = 1 \\
& \lambda_i \geq 0, \quad j = 1, \cdots, m;\ r = 1, \cdots q
\end{aligned}
$$

产出导向的 BCC（对偶）模型：对每个决策单元 k，

$$\begin{aligned}
&\max \phi \\
\mathbf{s.t.}\ &\sum_{i=1}^{n} \lambda_i x_{ij} \le x_{kj} \\
&\sum_{i=1}^{n} \lambda_i y_{ir} \ge \phi y_{kr} \\
&\sum_{i=1}^{n} \lambda_i = 1 \\
&\lambda_i \ge 0, \quad j = 1, \cdots, m;\ r = 1, \cdots q
\end{aligned}$$

11.2.3 带非期望产出的 SBM 模型

在径向模型中，效率改善主要指的是投入或产出的等比例线性放缩，同时忽略了平行于坐标轴的弱有效的情形，而 SBM 模型纳入无效率的松弛改进，保证最终的结果是强有效的。

标准 SBM 模型形式为：对每个决策单元 k，

$$\begin{aligned}
&\min \rho = \frac{1 - \frac{1}{m}\sum_{j=1}^{m} s_j^- / x_{kj}}{1 + \frac{1}{q}\sum_{r=1}^{q} s_r^- / y_{kj}} \\
\mathbf{s.t.}\ &X\lambda + s^- = x_k \\
&Y\lambda - s^+ = y_k \\
&\lambda, s^-, s^+ \ge 0, \quad j = 1, \cdots, m;\ r = 1, \cdots, q
\end{aligned}$$

该模型旨在通过最小化投入的相对减少量和最大化期望产出的相对增加量来衡量效率。目标函数最优解中 ρ^* 表示效率值，该模型同时从投入和产出两个方面考察无效率的表现，故称为非径向模型。

若同时加入非期望产出，则得到**带非期望产出的 SBM 模型**：对每个决策单元 k，

$$\begin{aligned}
&\min \rho = \frac{1 - \frac{1}{m}\sum_{i=1}^{m} \frac{s_i^-}{x_{ik}}}{1 + \frac{1}{q+b}\left(\sum_{r=1}^{q} \frac{s_r^g}{y_{rk}^g} + \sum_{t=1}^{b} \frac{s_t^b}{y_{tk}^b}\right)} \\
\mathbf{s.t.}\ &X\lambda + s^- = x_k \\
&Y^g\lambda - s^g = y_k^g \\
&Y^b\lambda + s^b = y_k^b \\
&\lambda, s^-, s^g, s^b \ge 0, \quad i = 1, \ldots, m; r = 1, \ldots, q; t = 1, \ldots, b
\end{aligned}$$

其中，m, q, b 分别为投入、期望产出、非期望产出的数量；

s^-, s^g, s^b 分别为投入、期望产出和非期望产出的松弛变量；

x_k, y_k^g, y_k^b 分别为第 k 个 DMU 的投入、期望产出和非期望产出；

X, Y^g, Y^b 分别为投入、期望产出和非期望产出的数据矩阵；

λ 为权重向量。

11.2.4 超效率模型

在传统的 DEA 模型中，效率前沿上的决策单元（DMU）都会被赋予效率值 1，这意味着这些单位都被视为“最优”或“有效”的代表。然而，这种设定也带来了一个问题：无法进一步区分这些高效单元之间的相对表现。

超效率模型正是为了解这一问题而提出的。其核心思想是：将原本处于效率前沿的某个 DMU 从参考集中移除，再评估它在其余 DMU 所构成的新前沿面上的表现。这样做的结果是，原本效率为 1 的 DMU 现在可能会得到一个大于 1 的效率值，从而反映出它在“超越现有最佳实践”时的表现。

换句话说，**超效率模型不是重新评估谁是有效的，而是衡量那些已经被认为是有效的 DMU 之间，谁更胜一筹**。这使得我们可以在不改变原有 DEA 基本框架的前提下，对高效单元进行排序和深入比较，尤其适用于需要甄选标杆或识别领先实践者的场景。

超效率模型在实际应用中非常有用，例如在绩效评估、资源分配、政策制定等方面，帮助决策者识别真正具有引领作用的个体，并据此推广先进经验。

以**投入导向的 BCC（对偶）模型**改写为超效率模型为例：对每个决策单元 k，

$$
\begin{aligned}
& \min \theta \\
\text{s.t.} \quad & \sum_{\substack{i=1 \\ i \neq k}}^{n} \lambda_i x_{ij} \leq \theta x_{kj}, \quad j = 1, \cdots, m \\
& \sum_{\substack{i=1 \\ i \neq k}}^{n} \lambda_i y_{ir} \geq y_{kr}, \quad r = 1, \cdots, q \\
& \sum_{\substack{i=1 \\ i \neq k}}^{n} \lambda_i = 1, \\
& \lambda_i \geq 0, \quad i \neq k
\end{aligned}
$$

`mathmodels` 包对 `deaR` 包中的相应函数做了封装，提供了四个函数：`basic_DEA`、`super_DEA`、`basic_SBM` 和 `super_SBM`，分别适用于不同类型的 DEA 或 SBM 模型变体：径向数据包络分析（DEA）模型（Charnes 等，1978；Banker 等，1984）和基于松弛变量测度的 SBM 模型（Tone，2001）计算效率得分提供了简化的操作接口，支持在规模报酬不变（CRS）或规模报酬可变（VRS）条件下，运行标准模型和超效率模型，并可选择性地处理非期望产出。

基本语法为（同其它函数）：

```
basic_DEA(data, inputs, outputs, ud_outputs = NULL,
          orientation = "io", rts = "vrs")
```

- `data`：数据框，第 1 列为 DMU 名字；
- `inputs/outputs`（包含 `ud_outputs`）：投入、产出的列名或列索引；
- `ud_outputs`：非期望产出位于产出中的位置索引，默认为 `NULL`；
- `orientation`：设置导向：`"io"`（默认，投入导向）、`"oo"`（产出导向）；
- `rts`：设置规模收益：`"crs"`（不变规模收益）、“vrs”‘（默认，可变规模收益）。

11.2.5 Malmquist 指数

Malmquist 指数是一种基于数据包络分析（DEA）框架、用于衡量决策单元（DMU）在不同时期之间**全要素生产率**变化的指标。它通过比较不同时间点的技术前沿，将生产率变化分解为 **技术效率变化（Efficiency Change, EC）**和 **技术进步（Technical Change, TC）**。

该指数建立在 **Shephard 距离函数**的基础上，适用于投入导向或产出导向模型。无论选择哪种导向，计算时均采用原始距离函数值，而非人为归一化的 “效率得分”。

1. Shephard 距离函数的定义

- **投入导向距离函数** $D_I^t(\mathbf{x}, \mathbf{y})$ 表示：在保持产出 $\mathbf{y}$ 不变的前提下，投入向量 $\mathbf{x}$ 可以按比例缩小的最大倍数 θ，使得 $(\theta\mathbf{x}, \mathbf{y})$ 仍属于时期 t 的生产可能集 P^t：

$$D_I^t(\mathbf{x}, \mathbf{y}) = \sup\left\{\theta \geq 0 \mid (\theta\mathbf{x}, \mathbf{y}) \in P^t\right\}$$

此时，$D_I^t \in (0, 1]$，且当 $D_I^t = 1$ 时表示技术有效；$D_I^t < 1$ 表示存在投入冗余。该值即为 DEA 模型求解出的效率值 θ^*。

- **产出导向距离函数** $D_O^t(\mathbf{x}, \mathbf{y})$ 表示：在保持投入 $\mathbf{x}$ 不变的前提下，产出向量 $\mathbf{y}$ 可以按比例放大的最大倍数 η，使得 $(\mathbf{x}, \eta\mathbf{y})$ 仍属于时期 t 的生产可能集 P^t：

$$D_O^t(\mathbf{x}, \mathbf{y}) = \sup\left\{\eta \geq 0 \mid (\mathbf{x}, \eta\mathbf{y}) \in P^t\right\}$$

此时，$D_O^t \in [1, \infty)$，且当 $D_O^t = 1$ 时表示技术有效；$D_O^t > 1$ 表示存在产出不足。

> **注意:** 尽管实践中常将产出导向的 “效率评分” 定义为 $1/D_O^t \in (0, 1]$ 以便解释，但在 Malmquist 指数的正式计算中，应直接使用原始的距离函数值 $D_O^t \geq 1$，以保证理论一致性。

2. 生产可能集说明

这里的 **生产可能集** $P^t = \{(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}$ 可生产$\mathbf{y}\}$ 是由 DEA 方法构建的技术集合，表示在时期 t 下所有可行的 “投入-产出” 组合。其边界构成生产前沿。

- 在 **规模报酬不变（CRS）**假设下，技术集允许自由缩放；
- 在 **规模报酬可变（VRS）**假设下，技术集限制了大规模扩张的有效性。

注：单投入单导出的生产可能集（DEA 技术集），就是如前文图示的前沿线（面）及其右下方的区域，可以是规模收益 CRS 或 VRS 假设下。

3. 跨期距离函数的构造

考虑两个相邻时期 t 和 $t+1$，我们需要计算以下四个距离函数（统一使用同一导向，例如产出导向）：

- $D^t(\mathbf{x}^t, \mathbf{y}^t)$：用 t 期技术评估 t 期自身表现；
- $D^{t+1}(\mathbf{x}^{t+1}, \mathbf{y}^{t+1})$：用 $t+1$ 期技术评估 $t+1$ 期自身表现；
- $D^t(\mathbf{x}^{t+1}, \mathbf{y}^{t+1})$：将 $t+1$ 期的数据投影到 t 期的技术前沿；
- $D^{t+1}(\mathbf{x}^t, \mathbf{y}^t)$：将 t 期的数据投影到 $t+1$ 期的技术前沿。

这些距离函数通过 DEA 模型分别求解，是构建 Malmquist 指数的基础。

4. 参照集的选择

Malmquist 指数需要比较不同时期的技术前沿（即生产可能集），因此必须明确：以哪些决策单元作为构建每个时期技术前沿的基础？这就是所谓的“参照集”选择问题。

主要有三种构建方式：

- **同期参照集**：每个时期的生产可能集仅基于该时期自身的观测数据构建，即各期技术前沿独立估计。距离函数的计算以当期样本为参考基准，不跨期混合数据。
- **序列参照集**：技术前沿随时间逐步累积扩展，后期的技术集包含当前及之前所有时期的数据。由此构建的生产可能集具有非递减性，反映技术演进过程中对历史最佳实践的持续吸收。
- **全局参照集**：将所有时间段的观测数据合并，共同构建一个跨期统一的“全局生产可能集”。该前沿代表整个样本期内的最优技术边界，不随时间变化，为所有时期提供一致的评估基准。

5. Malmquist 指数的定义

Malmquist 生产率指数（MI）定义为：

$$\mathrm{MI}^t = \sqrt{\frac{D^t(\mathbf{x}^{t+1}, \mathbf{y}^{t+1}) / D^t(\mathbf{x}^t, \mathbf{y}^t)}{D^{t+1}(\mathbf{x}^{t+1}, \mathbf{y}^{t+1}) / D^{t+1}(\mathbf{x}^t, \mathbf{y}^t)}}$$

其中：

- 若 $\mathrm{MI}^t > 1$：表示生产率提高；
- 若 $\mathrm{MI}^t = 1$：表示生产率不变；
- 若 $\mathrm{MI}^t < 1$：表示生产率下降。

6. CRS 与 VRS 假设下的分解差异

Malmquist 指数的分解结果会因是否假设 **规模报酬不变（CRS）**或 **规模报酬可变（VRS）**而有所不同。

(1) CRS 假设下的 Malmquist 分解 在 CRS 下，技术前沿满足线性齐次性，此时 Malmquist 指数仅反映：

- **技术效率变化（EC）**
- **技术进步（TC）**

没有进一步区分规模效应。

因此，CRS-Malmquist 指数分解为：

$$\mathrm{MI}^t_{\mathrm{CRS}} = \underbrace{\frac{D^{t+1}_{\mathrm{CRS}}(\mathbf{x}^{t+1},\mathbf{y}^{t+1})}{D^{t}_{\mathrm{CRS}}(\mathbf{x}^{t},\mathbf{y}^{t})}}_{\mathrm{EC}_{\mathrm{CRS}}} \times \underbrace{\sqrt{\frac{D^{t}_{\mathrm{CRS}}(\mathbf{x}^{t+1},\mathbf{y}^{t+1})/D^{t+1}_{\mathrm{CRS}}(\mathbf{x}^{t+1},\mathbf{y}^{t+1})}{D^{t}_{\mathrm{CRS}}(\mathbf{x}^{t},\mathbf{y}^{t})/D^{t+1}_{\mathrm{CRS}}(\mathbf{x}^{t},\mathbf{y}^{t})}}}_{\mathrm{TC}_{\mathrm{CRS}}}$$

此版本捕捉的是“纯技术”层面的变化，但包含了规模效率变动的影响。

(2) VRS 假设下的 Malmquist 分解 在 VRS 下，技术前沿不再允许任意缩放，因此可以剔除规模效率变化的影响，得到 **纯技术效率变化**和 **纯技术进步**。

VRS-Malmquist 指数为：

$$\mathrm{MI}^t_{\mathrm{VRS}} = \underbrace{\frac{D^{t+1}_{\mathrm{VRS}}(\mathbf{x}^{t+1},\mathbf{y}^{t+1})}{D^{t}_{\mathrm{VRS}}(\mathbf{x}^{t},\mathbf{y}^{t})}}_{\mathrm{PEC}} \times \underbrace{\sqrt{\frac{D^{t}_{\mathrm{VRS}}(\mathbf{x}^{t+1},\mathbf{y}^{t+1})/D^{t+1}_{\mathrm{VRS}}(\mathbf{x}^{t+1},\mathbf{y}^{t+1})}{D^{t}_{\mathrm{VRS}}(\mathbf{x}^{t},\mathbf{y}^{t})/D^{t+1}_{\mathrm{VRS}}(\mathbf{x}^{t},\mathbf{y}^{t})}}}_{\mathrm{PTC}}$$

其中：

- PEC：Pure Efficiency Change（纯技术效率变化）
- PTC：Pure Technical Change（纯技术进步）

> **关键优势**：VRS 分解排除了规模效率波动的干扰，更适合识别“管理改进”或“技术创新”的真实影响。

(3) 规模效率变化（SEC）的提取 通过 CRS 与 VRS 结果的对比，还可进一步计算 **规模效率变化（SEC）**：

$$\mathrm{SEC}^t = \frac{\mathrm{EC}_{\mathrm{CRS}}}{\mathrm{PEC}} = \frac{D^{t+1}_{\mathrm{CRS}}/D^{t}_{\mathrm{CRS}}}{D^{t+1}_{\mathrm{VRS}}/D^{t}_{\mathrm{VRS}}}$$

若 $\mathrm{SEC}^t > 1$：表示规模效率提升（更接近最优规模）；
若 $\mathrm{SEC}^t < 1$：表示规模效率退化。

最终完整的 TFP 分解为：

$$\mathrm{MI}^t = \mathrm{PEC} \times \mathrm{PTC} \times \mathrm{SEC}$$

这被称为 **三重分解法**，完整揭示了生产率增长的来源。

`mathmodels` 包提供了 `malmquist()` 计算跨时期的 Malmquist 指数及其分解，该函数是对 `deaR` 包中相关函数做了封装，更便于使用。基本语法：

```
malmquist(data, period, inputs, outputs,
          orientation = "oo", rts = "vrs",
          type1 = "glob", type2 = "rd")
```

- data: 长格式数据框，第 1 列为决策单元，包含时期列
- period, inputs, outputs: 时期、投入和产出对应的列索引或列名
- orientation: "io"（投入导向），"oo"（产出导向，默认）
- rts: "vrs"（可变规模报酬，默认），"crs"（不变规模报酬）
- type1: 参照集："cont"（当期），"seq"（序列）或 "glob"（全局，默认）
- type2: 分解方法："fgnz"（Färe 等，1994），"rd"（Ray 和 Desli，1997，默认），"gl"（广义 Malmquist）或 "bias"（偏向性技术变化）

返回包含 Period、DMU、Malmquist 指数、EC（效率变化）、TC（技术变化）、PECH（纯效率变化）和 SECH（规模效率变化）的数据框。

11.3 DEA 案例

加载包：

```
library(tidyverse)
library(mathmodels)
```

案例数据：2011 年各省（市、自治区）医院的部分投入和产出指标。以床位数和卫生技术人员数作为投入，以诊疗人次数和入院人数作为产出，（可选）医疗废弃物作为非期望产出。

```
df = read_csv("data/hospital.csv")
df
```

```
# A tibble: 31 x 6
   DMU   `床位数(万个)` `卫技人员数(万个)` `诊疗人次数(万人次)` `入院人数(万人)`
   <chr>          <dbl>              <dbl>                <dbl>            <dbl>
 1 安徽           14.1               13.3                 6334.            439.
 2 北京            8.76              12.9                10434.            188.
 3 福建            8.99               9.39                7206.            309.
 4 甘肃            6.67               5.31                2772.            171.
 5 广东           24.6               28.9                29378.            799.
 6 广西           95.8               10.6                 6445.            326.
 7 贵州            7.84               6.93                2911.            243.
 8 海南            2.14               2.63                1269.             61.2
 9 河北           18.8               18.4                 8249.            568.
10 河南           24.0               23.1                11408.            704.
# i 21 more rows
# i 1 more variable: `医疗废弃物(万套)` <dbl>
```

先以投入导向、VRS 假设下的 DEA 模型（BCC）为例，不考虑非期望产出：

```
bbc_in = basic_DEA(df, inputs = 2:3, outputs = 4:5,
```

```
                orientation = "io", rts = "vrs")
```

- 查看各 DMU 效率值

```
bbc_in$efficiencies
```

```
     安徽      北京      福建      甘肃      广东      广西      贵州      海南
0.9548803 0.9184863 1.0000000 0.8550507 1.0000000 0.7966680 0.9708290 0.9414552
     河北      河南    黑龙江      湖北      湖南      吉林      江苏      江西
0.9390585 0.9615116 0.6900295 0.9703535 1.0000000 0.7418425 0.9602820 1.0000000
     辽宁    内蒙古      宁夏      青海      山东      山西      陕西      上海
0.7314317 0.7214879 0.9100736 0.9074069 1.0000000 0.6190050 0.8281242 1.0000000
     四川      天津      西藏      新疆      云南      浙江      重庆
1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 0.9505551 0.8922566
```

- `bbc_in` 对象中还包含其它结果（略）：

```
bbc_in$slacks      # 松弛变量
bbc_in$lambdas     # 权重
bbc_in$targets     # 目标值
bbc_in$references  # 参照单元
bbc_in$returns     # 规模收益
```

- **松弛变量（s）**：衡量投入过度或产出不足的程度，反映决策单元在哪些方面可以进一步优化：
  - 投入松弛 = 实际投入 - 目标投入
  - 产出松弛 = 目标产出 - 实际产出
- **权重（λ）**：表示参考单元的权重，用于构建效率前沿；每个 DMU 的效率值由其参考单元的线性组合计算：
  - 对于非有效 DMU（效率值 < 1），λ 表示它应该向哪些有效 DMU（效率值 $= 1$）学习，$\lambda > 0$ 的 DMU 是其参考单元
- **目标值**： DMU 达到 DEA 有效时应有的投入和产出水平；对非有效 DMU，目标值 = 参考单元的加权组合：
  - 投入目标 = 实际投入 - 投入松弛
  - 产出目标 = 实际产出 + 产出松弛
- **参照单元**：效率前沿上的 DMU，非有效 DMU 通过模仿它们来提高效率；$\lambda > 0$ 的 DMU。
- **规模收益**：表示 DMU 的规模效率，基于 VRS 模型的 λ 值之和：
  - irs（$\sum\lambda < 1$）：规模报酬递增（扩大规模可提升效率）
  - drs（$\sum\lambda > 1$）：规模报酬递减（规模过大导致效率下降）
  - crs（$\sum\lambda = 1$）：规模报酬不变（最优规模）

若计算同样设置下的超效率：

```
res = super_DEA(df, inputs = 2:3, outputs = 4:5,
                orientation = "io", rts = "vrs")
res$efficiencies
```

```
     安徽      北京      福建      甘肃      广东      广西      贵州      海南
0.9548803 0.9184863 1.0518247 0.8550507        NA 0.7966680 0.9708290 0.9414552
     河北      河南     黑龙江      湖北      湖南      吉林      江苏      江西
0.9390585 0.9615116 0.6900295 0.9703535 1.0368439 0.7418425 0.9602820 1.0036738
     辽宁     内蒙古      宁夏      青海      山东      山西      陕西      上海
0.7314317 0.7214879 0.9100736 0.9074069        NA 0.6190050 0.8281242 1.1885921
     四川      天津      西藏      新疆      云南      浙江      重庆
1.0446519 1.0295240 2.9436174 1.0114148 1.1956914 0.9505551 0.8922566
```

注意，有些 DMU 的超效率为 NA，对应其线性规划模型无可行解，一种处理办法是将其替换为标准 DEA 效率值：

```
idx = is.na(res$efficiencies)
res$efficiencies[idx] = bbc_in$efficiencies[idx]
res$efficiencies
```

```
     安徽      北京      福建      甘肃      广东      广西      贵州      海南
0.9548803 0.9184863 1.0518247 0.8550507 1.0000000 0.7966680 0.9708290 0.9414552
     河北      河南     黑龙江      湖北      湖南      吉林      江苏      江西
0.9390585 0.9615116 0.6900295 0.9703535 1.0368439 0.7418425 0.9602820 1.0036738
     辽宁     内蒙古      宁夏      青海      山东      山西      陕西      上海
0.7314317 0.7214879 0.9100736 0.9074069 1.0000000 0.6190050 0.8281242 1.1885921
     四川      天津      西藏      新疆      云南      浙江      重庆
1.0446519 1.0295240 2.9436174 1.0114148 1.1956914 0.9505551 0.8922566
```

最后，再计算另外四种 DEA 效率：

```
ccr_in = basic_DEA(df, inputs = 2:3, outputs = 4:5,
                   orientation = "io", rts = "crs")
ccr_out = basic_DEA(df, inputs = 2:3, outputs = 4:5,
                   orientation = "oo", rts = "crs")
bbc_out = basic_DEA(df, inputs = 2:3, outputs = 4:5,
                   orientation = "oo", rts = "vrs")
sbm_ud_out = basic_SBM(df, inputs = 2:3, outputs = 4:6, ud_outputs = 3,
                       orientation = "oo", rts = "vrs")
```

合并结果，并计算规模效率（CRS 效率 = VRS 效率 * 规模效率）：

```
tibble(DMU = df$DMU, ccr_in = ccr_in$efficiencies,
       ccr_out = ccr_out$efficiencies,
       bbc_in = bbc_in$efficiencies,
```

```
       bbc_out = bbc_out$efficiencies,
       sbm_ud_out = sbm_ud_out$efficiencies,
       scale_eff = ccr_in / bbc_in)
```

```
# A tibble: 31 x 7
   DMU   ccr_in ccr_out bbc_in bbc_out sbm_ud_out scale_eff
   <chr>  <dbl>   <dbl>  <dbl>   <dbl>      <dbl>     <dbl>
 1 安徽   0.944   0.944  0.955   0.957      0.690     0.989
 2 北京   0.917   0.917  0.918   0.917      0.839     0.999
 3 福建   1.00    1      1.00    1          1.000     1.000
 4 甘肃   0.834   0.834  0.855   0.849      1         0.975
 5 广东   1.000   1      1.00    1          1.000     1.000
 6 广西   0.795   0.795  0.797   0.802      0.847     0.998
 7 贵州   0.963   0.963  0.971   0.970      0.728     0.992
 8 海南   0.833   0.833  0.941   0.932      0.771     0.885
 9 河北   0.904   0.904  0.939   0.945      0.619     0.963
10 河南   0.883   0.883  0.962   0.970      0.841     0.918
# i 21 more rows
```

11.4 DEA-malmquist 案例

来自 deaR 包的 2005-2009 年中国各省份工业经济数据（长格式），共 155 行、5 列，包含决策单元（各省份）、年份，以及

投入指标：

- Capital：总资产（亿元）；
- Labor：年平均从业人员（万人）。

产出指标：

- GIOV：工业总产值（亿元）。

该数据集可评估中国工业部门在“十一五”期间的生产效率变化、技术进步与规模效益。数据结构清晰，适合进行跨期、跨区域的效率比较分析。

```
dat = read_csv("data/economy.csv")
dat
```

```
# A tibble: 155 x 5
   DMUs         Period Capital Labor   GIOV
   <chr>         <dbl>   <dbl> <dbl>  <dbl>
 1 Beijing        2005  12830.  117.   6946.
 2 Tianjin        2005   6348.  122.   6774.
```

```
 3 Hebei          2005   9474. 292.  11008.
 4 Shanxi_1       2005   7045. 213.   4851.
 5 Neimenggu      2005   4596.  83.7  2996.
 6 Liaoning       2005  11902. 277.  10815.
 7 Jilin          2005   4507. 102.   3792.
 8 Heilongjiang   2005   5174. 137.   4715.
 9 Shangai        2005  15906. 260.  15768.
10 Jiangsu        2005  25489. 704.  32707.
# i 145 more rows
```

调用 `malmquist()` 函数计算 Malmquist 指数及其分解，需要提供长格式数据框，设置日期、投入、产出列，可选提供导向、规模收益假设、参照集、计算方法：

```
malmquist(dat, period = 2, inputs = 3:4, outputs = 5,
          orientation = "oo", rts = "vrs",
          type1 = "glob", type2 = "rd")
```

```
# A tibble: 124 x 7
   Period    DMU             mi    ec    tc  pech  sech
   <chr>     <chr>        <dbl> <dbl> <dbl> <dbl> <dbl>
 1 2005~2006 Beijing      1.18  0.970  1.22 0.970 1.00
 2 2005~2006 Tianjin      1.22  0.993  1.23 1.00  0.993
 3 2005~2006 Hebei        1.03  0.979  1.05 0.977 1.00
 4 2005~2006 Shanxi_1     0.969 0.939  1.03 0.935 1.00
 5 2005~2006 Neimenggu    1.20  0.987  1.22 0.990 0.997
 6 2005~2006 Liaoning     1.13  1.01   1.11 1.01  1.000
 7 2005~2006 Jilin        1.10  0.956  1.15 0.950 1.01
 8 2005~2006 Heilongjiang 1.05  0.966  1.09 0.963 1.00
 9 2005~2006 Shangai      1.12  0.990  1.13 1.00  0.990
10 2005~2006 Jiangsu      0.942 0.891  1.06 1     0.891
# i 114 more rows
```

`mi` 为 Malmquist 指数，`ec` 为技术效率变化，`tc` 为技术进步，`pech` 为纯技术效率变化，`sech` 为规模效率变化。

更多的 DEA 模型还有：考虑环境因素和随机噪声对决策单元效率影响的三阶段 DEA 模型；固定前沿生产函数的参数法随机前沿模型；以及以效率为因变量研究效率的影响因素的 Tobit 回归模型。

第十二章　不平等度量

在经济学和社会科学领域，衡量收入或财富分配的不平等程度是一个重要的研究课题。为了量化这种不平等，研究人员开发了多种统计指标，其中最常用的两个是**基尼系数**和**泰尔指数**。

加载包：

```
library(tidyverse)
library(mathmodels)
```

12.1　个体数据与平均数据

个体数据，就是每个人占一行，记录其收入；（分组）**平均数据**，是将人按某分组变量（如部门）分组，每组一行，记录该组的平均收入和人数。

比如，这是 5 名员工的个体数据：

```
df = tibble(dep = c(rep(" 技术部", 3), rep(" 销售部", 2)),
            emp = LETTERS[1:5],
            income = c(10, 12, 20, 5, 8))
df
```

```
# A tibble: 5 x 3
  dep    emp   income
  <chr>  <chr>  <dbl>
1 技术部 A         10
2 技术部 B         12
3 技术部 C         20
4 销售部 D          5
5 销售部 E          8
```

按部门分组汇总，计算每个部门的人均收入和人数，就得到平均数据：

```
df |>
  summarise(income = mean(income), pop = n(), .by = dep)
```

```
# A tibble: 2 x 3
  dep    income   pop
  <chr>   <dbl> <int>
1 技术部    14       3
2 销售部     6.5     2
```

最常见的平均数据，就是人均指标和人口数的数据，也称为人口加权数据。

计算基尼系数或泰尔指数时，需要区分个体数据和平均数据（人口加权数据），二者的计算公式是不同的。

12.2 基尼系数

基尼系数，通过洛伦茨曲线计算，反映实际收入分布与完全平等线之间的面积比例。该指标直观且易于理解，但对中间群体的变化较为敏感，而对极端值的敏感性相对不足。

12.2.1 洛伦兹曲线与基尼系数

洛伦兹曲线，是一种用来展示社会中收入或财富的分配情况的曲线图。具体来说，它展示的是：

- x 轴：表示人口的累计百分比，从最贫穷的人到最富有的人排列。
- y 轴：表示收入或财富的累计百分比。

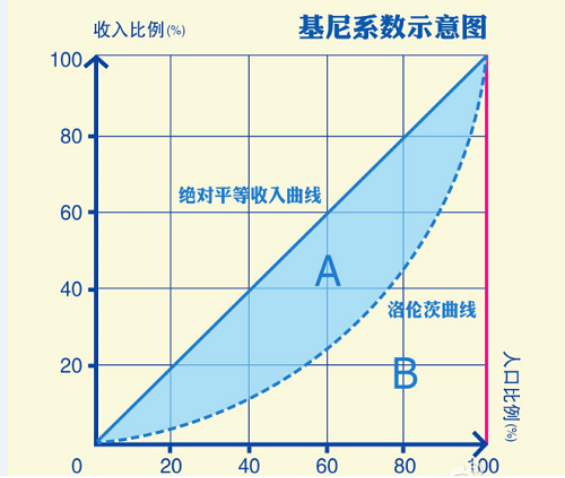


图 12.1: 洛伦兹曲线与基尼系数

基尼系数，通过洛伦兹曲线与绝对平等线之间的面积来衡量不平等性。具体来说：

- 如果洛伦兹曲线与绝对平等线之间的面积越大，表示不平等性越高。
- 基尼系数实际上就是该面积的量化，是介于 0 到 1 之间的数值。

具体解释：

- 基尼系数 $= 0$：意味着绝对平等，所有人的收入或财富是完全一样的。洛伦兹曲线和绝对平等线重合，没有差距。
- 基尼系数 $= 1$：意味着绝对不平等，所有收入或财富都集中在一个人的手里。洛伦兹曲线向下弯到几乎贴近 x 轴和 y 轴，除了一个点之外，其他人都几乎没有收入或财富。

12.2.2 基尼系数计算方法

(1) 针对个体数据（比如个体收入向量）：

$$Gini = \frac{\sum_{i=1}^{n}\sum_{i=1}^{n}|x_i - x_j|}{2n^2\bar{x}}$$

mathmodels 包提供了 gini0() 函数按上述公式计算个体数据的基尼系数，基本语法：

```
gini0(y)
```

- y 为个体收入向量。

(2) 针对平均数据：即面积法，利用数值积分，近似计算绝对平等线与洛伦兹曲线围成的面积。

记平均收入向量 $\mathbf{I} = (I_1, I_2, \ldots, I_n)$ 和人口向量 $\mathbf{P} = (P_1, P_2, \ldots, P_n)$。

- 对 $\mathbf{I}/\mathbf{P}$ 从小到大排序，按排序索引重排 $\mathbf{I}$ 和 $\mathbf{P}$，为了简便仍记为 $\mathbf{I}$ 和 $\mathbf{P}$。
- 计算累积人口比例和累积收入比例：

$$x_k = \frac{\sum_{i=1}^{k} P_i}{\sum_{i=1}^{n} P_i}, \quad y_k = \frac{\sum_{i=1}^{k} I_i}{\sum_{i=1}^{n} I_i}$$

  其中，$k = 0, 1, \cdots, n$ 且 $x_0 = y_0 = 0$。

- 用梯形法数值积分计算洛伦兹曲线下面积：

$$A = \sum_{i=1}^{n} \frac{(x_i - x_{i-1}) \cdot (y_i + y_{i-1})}{2}$$

- 绝对平等线下面积为 0.5，基尼系数为两面积之差与绝对平等线面积的比值：

$$G = \frac{0.5 - A}{0.5} = 1 - 2A$$

mathmodels 包提供了 gini() 函数按上述步骤计算平均数据的基尼系数，基本语法：

```
gini(y, pop)
```

- y 和 pop 分别为各分组的人均收入和人口数构成的向量。

12.2.3 案例：批量计算基尼系数

数据整理自国家统计局网站，包含 2014-2023 年全国 31 个省份八个主要行业：“信息业”、“制造业”、“医疗”、“商贸”、“建筑业”、“教育”、“科研”、“金融业” 的平均工资和就业人数。

```
dat = readxl::read_excel("data/income.xlsx")
dat
```

```
# A tibble: 2,480 x 5
   行业   地区   年份 平均工资 就业人数
   <chr>  <chr> <dbl>    <dbl>    <dbl>
 1 信息业 上海   2014   170174   248000
 2 信息业 上海   2015   183365   254000
 3 信息业 上海   2016   200657   268000
 4 信息业 上海   2017   212063   307000
 5 信息业 上海   2018   232522   356000
 6 信息业 上海   2019   237405   418000
 7 信息业 上海   2020   270619   448000
 8 信息业 上海   2021   303573   507000
 9 信息业 上海   2022   330126   544000
10 信息业 上海   2023   363745   528000
# i 2,470 more rows
```

1. 每个行业每年份内省份间工资的不平等程度（基尼系数）

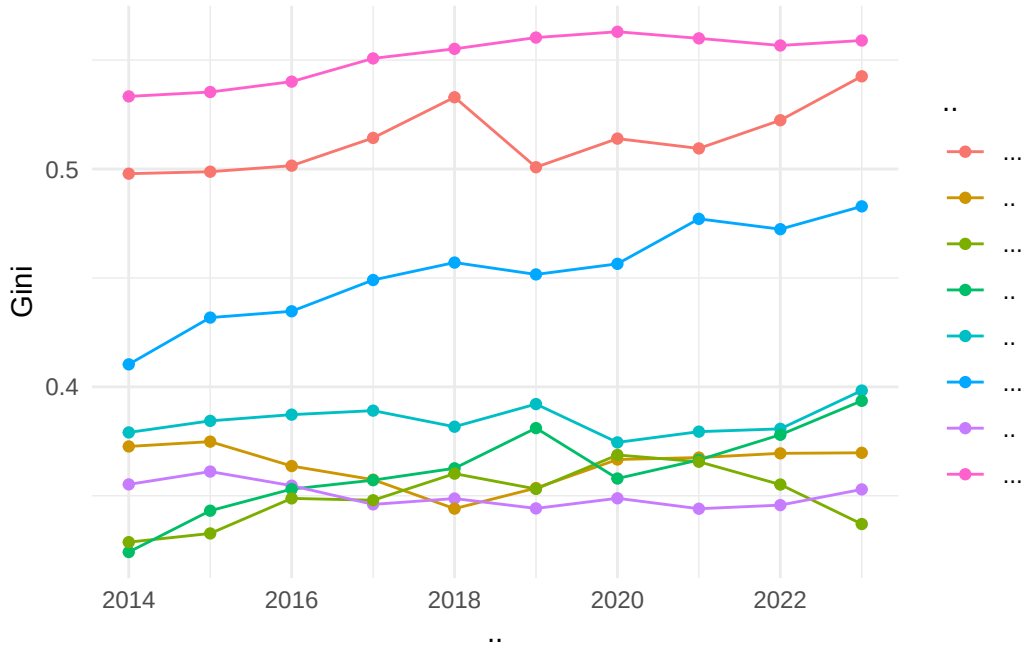
- 就是简单的分组汇总：按 年份和 行业分组，每组计算出一个基尼系数值

```
res = dat |>
  summarise(Gini = gini(平均工资, 就业人数), .by = c(年份, 行业))
res
```

```
# A tibble: 80 x 3
    年份 行业     Gini
   <dbl> <chr>  <dbl>
 1  2014 信息业 0.410
 2  2015 信息业 0.432
 3  2016 信息业 0.435
 4  2017 信息业 0.449
 5  2018 信息业 0.457
 6  2019 信息业 0.452
 7  2020 信息业 0.456
 8  2021 信息业 0.477
 9  2022 信息业 0.472
10  2023 信息业 0.483
# i 70 more rows
```

- 结果是整洁长表，接着做可视化非常容易：

```
res |>
  ggplot(aes(年份, Gini, color = 行业)) +
  geom_line() +
  geom_point() +
  theme_minimal()
```



- 若想结果更适合人类阅读，再来个长变宽：

```
res |>
  pivot_wider(names_from = 行业, values_from = Gini)
```

```
# A tibble: 10 x 9
    年份 信息业 制造业  医疗  商贸 建筑业  教育  科研 金融业
   <dbl>  <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>
 1  2014  0.410  0.533 0.355 0.379  0.498 0.373 0.324  0.329
 2  2015  0.432  0.535 0.361 0.384  0.499 0.375 0.343  0.333
 3  2016  0.435  0.540 0.355 0.387  0.502 0.364 0.353  0.349
 4  2017  0.449  0.551 0.346 0.389  0.514 0.357 0.357  0.348
 5  2018  0.457  0.555 0.349 0.382  0.533 0.344 0.363  0.360
 6  2019  0.452  0.560 0.344 0.392  0.501 0.354 0.381  0.353
 7  2020  0.456  0.563 0.349 0.374  0.514 0.367 0.358  0.369
 8  2021  0.477  0.560 0.344 0.379  0.509 0.368 0.366  0.366
 9  2022  0.472  0.557 0.346 0.381  0.522 0.369 0.378  0.355
10  2023  0.483  0.559 0.353 0.398  0.543 0.370 0.394  0.337
```

2. 每个省份每年份内行业间工资的不平等程度（基尼系数）

```
dat |>
  summarise(Gini = gini(平均工资, 就业人数), .by = c(年份, 地区)) |>
  pivot_wider(names_from = 地区, values_from = Gini)
```

```
# A tibble: 10 x 32
    年份  上海  云南 内蒙古  北京  吉林  四川  天津  宁夏  安徽  山东  山西
   <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  2014 0.542 0.541  0.423 0.325 0.484 0.546 0.637 0.471 0.522 0.613 0.475
 2  2015 0.525 0.546  0.423 0.305 0.484 0.525 0.601 0.461 0.521 0.603 0.455
 3  2016 0.501 0.538  0.411 0.276 0.471 0.508 0.566 0.446 0.510 0.589 0.450
 4  2017 0.478 0.536  0.378 0.261 0.441 0.500 0.517 0.448 0.508 0.575 0.455
 5  2018 0.457 0.527  0.356 0.226 0.407 0.515 0.479 0.427 0.537 0.559 0.438
 6  2019 0.393 0.433  0.334 0.151 0.357 0.517 0.460 0.399 0.502 0.504 0.410
 7  2020 0.403 0.436  0.348 0.130 0.347 0.481 0.438 0.397 0.491 0.477 0.399
 8  2021 0.397 0.426  0.377 0.138 0.365 0.465 0.417 0.407 0.506 0.475 0.402
 9  2022 0.377 0.424  0.385 0.136 0.373 0.462 0.429 0.439 0.527 0.486 0.410
10  2023 0.360 0.434  0.402 0.137 0.390 0.425 0.441 0.484 0.533 0.474 0.405
# i 20 more variables: 广东 <dbl>, 广西 <dbl>, 新疆 <dbl>, 江苏 <dbl>,
#   江西 <dbl>, 河北 <dbl>, 河南 <dbl>, 浙江 <dbl>, 海南 <dbl>, 湖北 <dbl>,
#   湖南 <dbl>, 甘肃 <dbl>, 福建 <dbl>, 西藏 <dbl>, 贵州 <dbl>, 辽宁 <dbl>,
#   重庆 <dbl>, 陕西 <dbl>, 青海 <dbl>, 黑龙江 <dbl>
```

12.2.4 绘制洛伦兹曲线

以“教育”行业 2023 年的数据为例，

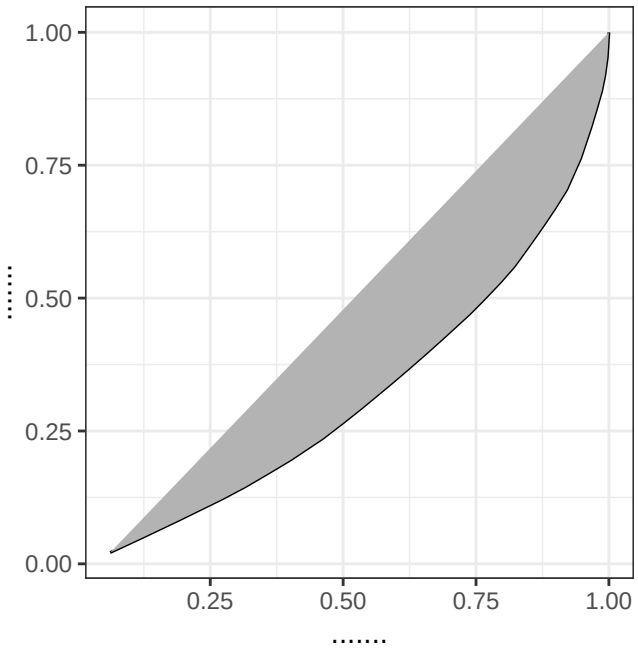
```
edu23 = dat |>
  filter(行业 == " 教育", 年份 == "2023")
edu23
```

```
# A tibble: 31 x 5
   行业  地区    年份 平均工资 就业人数
   <chr> <chr>  <dbl>    <dbl>    <dbl>
 1 教育  上海    2023   235503   388000
 2 教育  云南    2023   116750   644000
 3 教育  内蒙古  2023   110236   366000
 4 教育  北京    2023   230965   508000
 5 教育  吉林    2023   100625   346000
 6 教育  四川    2023   116527  1238000
 7 教育  天津    2023   149140   212000
```

```
 8 教育   宁夏     2023   111737   112000
 9 教育   安徽     2023   119779   678000
10 教育   山东     2023   124519  1291000
# i 21 more rows
```

洛伦兹曲线（也是计算基尼系数），关键的一步是按 x / pop 排序，注意不是按平均收入 x 排序，然后计算人口数和平均收入的累计百分比，分别作为 x 轴和 y 轴绘图即可。

```
edu23 |>
  arrange(平均工资 / 就业人数) |>
  mutate(frac_pop = cumsum(就业人数) / sum(就业人数),
         frac_inc = cumsum(平均工资) / sum(平均工资)) |>
  ggplot(aes(frac_pop, frac_inc)) +
  geom_line() +
  geom_polygon(fill = "grey70") +
  coord_equal() +
  theme_bw() +
  labs(x = " 累计人数百分比", y = " 累计收入百分比")
```



- 再看一下，该洛伦兹曲线对应的基尼系数值：

```
gini(edu23$平均工资,edu23$就业人数)
```

```
[1] 0.3697145
```

12.3 泰尔指数

泰尔指数，是一种基于信息理论的不平等度量方法，其核心思想是衡量收入分布偏离 “完全平等” 状态的信息熵。泰尔指数的一个显著优点是它可以自然地分解为组间和组内不平等，从而便于分析不同层次（如省、市）对总体不平等的贡献。

12.3.1 总体泰尔指数

(1) 针对个体数据

$$T = \frac{1}{n}\sum_{i=1}^{n}\frac{Y_i}{\overline{Y}}\ln(\frac{Y_i}{\overline{Y}}) = \sum_{i=1}^{n}\frac{Y_i}{Y}\ln(\frac{Y_i}{\overline{Y}}) = \sum_{i=1}^{n}\frac{Y_i}{Y}\ln(\frac{Y_i/Y}{1/n})$$

其中，T 为总体泰尔指数，Y_i 表示第 i 个体的收入，$\overline{Y}$ 表示所有个体的平均收入，Y 表示所有个体的总收入。

三个表达式，按哪个计算都可以。

`mathmodels` 包提供了 `theil0()` 函数计算个体数据的总体泰尔指数，基本语法：

```
theil0(y)
```

- `y` 为个体收入向量。

(2) 针对平均数据（人口加权数据）

$$T = \sum_{i}\frac{Y_i}{Y}\ln\frac{Y_i/Y}{N_i/N} = \sum_{i}\frac{N_i}{N}\frac{\overline{Y}_i}{\overline{Y}}\ln\frac{\overline{Y}_i}{\overline{Y}}$$

其中，Y_i, N_i 分别表示第 i 组的总收入和总人口，Y, N 分别表示所有组的总收入和总人口，故 $\frac{Y_i}{Y}$ 表示第 i 组收入占总收入的比重，$\frac{N_i}{N}$ 表示第 i 组人口数占总人口数的比重。

总收入当然等于人口数乘以平均收入，所以，引入 $\overline{Y_i}$, $\overline{Y}$ 改写为第 2 式，以与平均数据相对应。

`mathmodels` 包提供了 `theil()` 函数计算个体数据的总体泰尔指数，基本语法：

```
theil(y, pop)
```

- `y` 和 `pop` 分别为各分组的人均收入和人口数构成的向量。

12.3.2 单分组变量的泰尔指数及其分解

泰尔指数分解，就是总体泰尔指数关于单个分组变量 `g` 分解为**组内**和**组间**泰尔指数，同样也需要区分：个体数据和平均数据。

总体泰尔指数，当然可以忽略分组，直接按个体数据或平均数据的计算公式计算。因为要做分解，故可采用分解后的组内与组间泰尔指数加和得到总体泰尔指数：

$$T = T_w + T_b$$

从逻辑上叙述清楚比摆一堆公式更好理解，泰尔指数分解计算步骤如下：

(1) 数据关于分组变量 g 分为多组，分别对第 i 组的数据，计算每组的总体泰尔指数即 T_{wi} （根据个体数据/平均数据选用相应的前文公式即可）；

(2) 组内泰尔指数 T_w 就是每组的总体泰尔指数关于各组收入对总收入占比的加权和：

$$T_w = \sum_i \frac{Y_i}{Y} T_{wi}$$

(3) 组间泰尔指数 T_b，是对原始数据按分组变量 g 分组汇总到各组的平均收入和人口数，再计算总体泰尔指数（此时一定是平均数据）；

(4) 进一步，可以计算第 i 组组内差距的贡献率以及组内、组间差距的贡献率：

$$R_{wi} = \frac{Y_i}{Y} * \frac{T_{wi}}{T} = \frac{N_i}{N}\frac{\overline{Y}_i}{\overline{Y}} * \frac{T_{wi}}{T}$$

$$R_w = \frac{T_w}{T}, \quad R_b = \frac{T_b}{T}$$

这些贡献率满足：$R_w = \sum_i R_{wi}$, $R_w + R_b = 1$。

`mathmodels` 包提供了 `theil0_g()` 和 `theil_g()` 函数，分别针对个体数据和平均数据，关于一个分组变量计算泰尔指数及其分解，基本语法：

```
theil0_g(data, group, y)
theil_g(data, group, y, pop)
```

- `data` 为包含所需变量的数据框；
- `group` 为单个分组变量的名字；
- `y` 和 `pop` 分别为各分组的人均收入和人口数构成的向量。

返回包含两个成分的列表：

- `theil`: 总体、组间、组内、每个分组的泰尔指数；
- `ratio`: 相应的贡献率。

12.3.3 两分组变量的泰尔指数及其分解

1. 两交叉分组变量的泰尔指数及其分解

对于两个交叉分组变量，例如 `group1`（如行业）和 `group2`（如区域：东部、中部、西部），是对单分组变量泰尔指数分解的扩展。总体泰尔指数仍分解为基于 `group1` 的组间和组内部分，但组内部分进一步分解为基于 `group2` 的组间和组内贡献，从而提供交叉分组下更细致的不平等分析。

组内泰尔指数（Tw）及其分解，目的是衡量 group1 内部的不平等，并按 group2 进一步分解。对每个 group1 分组，

- **group2 组间泰尔指数：**

在该 group1 内按 group2（如区域）汇总数据。计算每 group2 的平均收入和人口。使用泰尔公式计算：

$$T_{g2b} = \sum_j \left(\frac{Y_{ij}}{Y_i}\right) \ln\left(\frac{Y_{ij}/N_{ij}}{Y_i/N_i}\right)$$

其中，Y_{ij} 和 N_{ij} 是 group1 内 group2 第 j 组的收入和人口。

- **group2 组内泰尔指数**

若 group2 内有更细的数据（如区域内的省份），计算每 group2 单元的泰尔指数，并按人口占比加权：

$$T_{g2w} = \sum_j \left(\frac{N_{ij}}{N_i}\right) T_{wij}$$

其中，T_{wij} 是 group2 第 j 组内的泰尔指数（若仅为平均数据，T_{g2w} 为 0）。

mathmodels 包提供了 theil_g2_cross() 函数实现两交叉分组变量的泰尔指数及其分解，基本语法：

```
theil_g2_cross(data, group1, group2, y, pop)
```

- data 为包含所需变量的数据框；
- group1 和 group2 分别为交叉的第一和第二分组变量的名字；
- y 和 pop 分别为各分组的人均收入和人口数构成的向量。

返回包含两个成分的列表：

- theil: 总体、group1 组间、group1 组内、group2 组间、group2 组内、每个 group1 分组的组内泰尔指数；
- ratio: 相应的贡献率。

2. 两嵌套分组变量的泰尔指数及其分解

对于两嵌套分组变量 group1 和 group2，其中 group2 嵌套在 group1 内（比如省嵌套市），总体泰尔指数分解为基于 group1 的组间不平等、基于 group2 的组间不平等以及 group2 内的组内不平等。与两交叉分组变量不同，嵌套结构的泰尔指数分解更侧重于层次结构。

(1) group1 组间泰尔指数（衡量 group1 各分组之间的不平等）

按 group1 汇总数据，计算每组的总收入和总人口，计算组级平均收入，使用平均数据的泰尔公式计算。

(2) group2 组间泰尔指数（衡量每个 group1 内 group2 各分组之间的不平等）

在每个 group1 分组下，计算每个 group2 分组的平均收入和总人口，计算分泰尔指数：

$$T_b^{g2i} = \sum_j \left(\frac{Y_{ij}}{Y_i}\right) \ln\left(\frac{Y_{ij}/N_{ij}}{Y_i/N_i}\right)$$

其中，Y_{ij} 和 N_{ij} 是 group1 内 group2 第 j 分组的收入和人口，Y_i 和 N_i 是该 group1 的总收入和人口。

若某 group1 内仅有一个 group2，则 $T_b^{g2i}=0$。

总体 T_b^{g2} 为各 group1 的 T_b^{g2i} 加权和：

$$T_b^{g2}=\sum_i\left(\frac{Y_i}{Y}\right)T_b^{g2i}$$

(3) group2 组内泰尔指数（衡量每个 group2 分组内部的不平等）

按 group1 和 group2 分组数据，计算每个分组内个体数据的泰尔指数 T_w^{ij}（若为平均数据，$T_w^{ij}=0$）。

总体 T_w 为各 group2 T_w^{ij} 的加权和：

$$T_w=\sum_i\sum_j\left(\frac{Y_{ij}}{Y}\right)T_w^{ij}$$

mathmodels 包提供了 theil_g2_nest() 函数实现两交叉分组变量的泰尔指数及其分解，基本语法：

```
theil_g2_nest(data, group1, group2, y, pop)
```

- data 为包含所需变量的数据框；
- group1 和 group2 分别为嵌套的第一和第二分组变量的名字；
- y 和 pop 分别为各分组的人均收入和人口数构成的向量。

返回包含两个成分的列表：

- theil: 总体、group1 组间、group2 组间、group2 组内、每个 group1/group2 分组的组内泰尔指数；
- ratio: 相应的贡献率。

总结区分：

- 单分组变量仅将总体泰尔指数分解为组间部分 T_b 和组内部分 T_w，无进一步细分。
- 两交叉分组变量将组内部分 T_w 分解为 T_{wb}（group2 组间）和 T_{ww}（group2 组内），强调 group2 在 group1 内的交叉作用。
- 两嵌套分组将组间部分 T_b 分为 T_{b1}（group1 间）和 T_{b2}（group2 间），T_w 仅为 group2 内的不平等，反映严格的层次结构。

12.3.4 案例：批量计算泰尔指数及其分解

仍使用前文的 2014-2023 年全国 31 个省份八个主要行业的平均工资和就业人数数据，增加一列 区域，将省份划分至” 东部”、“中部”、“西部”、“东部”。

```
df = dat |>
  mutate(区域 = case_match(地区,
    c(" 北京"," 天津"," 河北"," 上海"," 江苏"," 浙江"," 福建"," 山东"," 广东"," 海南") ~ " 东部",
    c(" 山西"," 安徽"," 江西"," 河南"," 湖北"," 湖南") ~ " 中部",
    c(" 辽宁"," 吉林"," 黑龙江") ~ " 东北",
```

```
    .default = " 西部"), .before = 地区)
df
```

```
# A tibble: 2,480 x 6
   行业   区域  地区   年份 平均工资 就业人数
   <chr>  <chr> <chr> <dbl>    <dbl>    <dbl>
 1 信息业 东部  上海   2014   170174   248000
 2 信息业 东部  上海   2015   183365   254000
 3 信息业 东部  上海   2016   200657   268000
 4 信息业 东部  上海   2017   212063   307000
 5 信息业 东部  上海   2018   232522   356000
 6 信息业 东部  上海   2019   237405   418000
 7 信息业 东部  上海   2020   270619   448000
 8 信息业 东部  上海   2021   303573   507000
 9 信息业 东部  上海   2022   330126   544000
10 信息业 东部  上海   2023   363745   528000
# i 2,470 more rows
```

1. 每年关于行业的收入不平等性（泰尔指数及其分解）

`theil_*()` 系列函数的返回值，特意设计为两个成分的列表，且每个成分是长度相同、元素命名相同的向量，非常方便分组汇总并展开批量结果。

每年就是按年份分组汇总，关于行业就是以 行业为分组变量，分别对每组数据计算关于单个分组变量的泰尔指数及其分解。因为每组返回长度为 2 的对象（列表），所以用 `reframe()` 代替 `summarise()`，结果是行数翻倍。

```
res = dat |>
  reframe(Theil = theil_g(pick(everything()), group = " 行业",
                          y = " 平均工资", pop = " 就业人数"),
          .by = 年份)
res
```

```
# A tibble: 20 x 2
    年份 Theil
   <dbl> <named list>
 1  2014 <dbl [11]>
 2  2014 <dbl [11]>
 3  2015 <dbl [11]>
 4  2015 <dbl [11]>
 5  2016 <dbl [11]>
 6  2016 <dbl [11]>
```

```
 7  2017 <dbl [11]>
 8  2017 <dbl [11]>
 9  2018 <dbl [11]>
10  2018 <dbl [11]>
11  2019 <dbl [11]>
12  2019 <dbl [11]>
13  2020 <dbl [11]>
14  2020 <dbl [11]>
15  2021 <dbl [11]>
16  2021 <dbl [11]>
17  2022 <dbl [11]>
18  2022 <dbl [11]>
19  2023 <dbl [11]>
20  2023 <dbl [11]>
```

Theil 列是嵌套的向量，再做横向（往宽）展开即可：

```
res |>
  unnest_wider(Theil)
```

```
# A tibble: 20 x 12
    年份      T     Tb     Tw 信息业 制造业   医疗   商贸 建筑业   教育   科研
   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
 1  2014 0.0609 0.0304 0.0305 0.0641 0.0148 0.0400 0.0660 0.0111 0.0243 0.0591
 2  2014 1      0.499  0.501  0.0490 0.0897 0.0461 0.0735 0.0334 0.0535 0.0447
 3  2015 0.0628 0.0326 0.0302 0.0603 0.0157 0.0339 0.0654 0.0113 0.0193 0.0511
 4  2015 1      0.519  0.481  0.0476 0.0885 0.0411 0.0701 0.0312 0.0449 0.0377
 5  2016 0.0633 0.0331 0.0302 0.0600 0.0170 0.0314 0.0669 0.0112 0.0187 0.0483
 6  2016 1      0.522  0.478  0.0497 0.0919 0.0404 0.0706 0.0295 0.0449 0.0364
 7  2017 0.0663 0.0348 0.0315 0.0598 0.0169 0.0322 0.0681 0.0122 0.0206 0.0445
 8  2017 1      0.525  0.475  0.0520 0.0831 0.0428 0.0674 0.0297 0.0492 0.0334
 9  2018 0.0670 0.0350 0.0319 0.0599 0.0156 0.0338 0.0681 0.0122 0.0221 0.0416
10  2018 1      0.523  0.477  0.0570 0.0710 0.0459 0.0684 0.0304 0.0538 0.0320
11  2019 0.0684 0.0333 0.0352 0.0666 0.0162 0.0412 0.0661 0.0147 0.0260 0.0439
12  2019 1      0.486  0.514  0.0675 0.0668 0.0624 0.0673 0.0302 0.0670 0.0351
13  2020 0.0718 0.0333 0.0386 0.0711 0.0163 0.0399 0.0774 0.0157 0.0308 0.0431
14  2020 1      0.463  0.537  0.0758 0.0633 0.0596 0.0720 0.0290 0.0790 0.0320
15  2021 0.0759 0.0352 0.0407 0.0683 0.0169 0.0387 0.0762 0.0171 0.0323 0.0443
16  2021 1      0.463  0.537  0.0758 0.0635 0.0568 0.0693 0.0270 0.0751 0.0321
17  2022 0.0835 0.0411 0.0425 0.0697 0.0182 0.0396 0.0827 0.0200 0.0304 0.0452
18  2022 1      0.492  0.508  0.0744 0.0608 0.0548 0.0686 0.0264 0.0655 0.0309
```

```
19  2023 0.0810 0.0406 0.0403 0.0693 0.0162 0.0375 0.0768 0.0173 0.0326 0.0452
20  2023 1      0.502  0.498  0.0777 0.0550 0.0557 0.0683 0.0222 0.0718 0.0320
# i 1 more variable: 金融业 <dbl>
```

每个年份，第一行是总体、行业间、行业内泰尔指数，各行业的（总体）泰尔指数；第二行是相应的贡献率，各行业贡献率之和等于行业内贡献率，行业间贡献率 + 行业内贡献率 = 1。

2. 每年关于区域的收入不平等性（批量泰尔指数及其分解）

同样做法，只需将分组变量换成 区域：

```
df |>
  reframe(Theil = theil_g(pick(everything()), group = " 区域",
                          y = " 平均工资", pop = " 就业人数"),
          .by = 年份) |>
  unnest_wider(Theil)
```

```
# A tibble: 20 x 8
    年份      T      Tb      Tw   东部   西部    东北   中部
   <dbl>  <dbl>   <dbl>  <dbl>  <dbl>  <dbl>   <dbl>  <dbl>
 1  2014 0.0609 0.00915 0.0517 0.0732 0.0263 0.0156  0.0150
 2  2014 1      0.150   0.850  0.718  0.0751 0.0150  0.0417
 3  2015 0.0628 0.00900 0.0538 0.0754 0.0283 0.0180  0.0175
 4  2015 1      0.143   0.857  0.713  0.0800 0.0161  0.0477
 5  2016 0.0633 0.00908 0.0542 0.0748 0.0294 0.0166  0.0209
 6  2016 1      0.143   0.857  0.702  0.0831 0.0138  0.0575
 7  2017 0.0663 0.00939 0.0570 0.0779 0.0312 0.0159  0.0239
 8  2017 1      0.142   0.858  0.699  0.0850 0.0118  0.0629
 9  2018 0.0670 0.00859 0.0584 0.0804 0.0298 0.0128  0.0232
10  2018 1      0.128   0.872  0.724  0.0798 0.00885 0.0596
11  2019 0.0684 0.0119  0.0565 0.0791 0.0251 0.0111  0.0217
12  2019 1      0.174   0.826  0.698  0.0659 0.00738 0.0541
13  2020 0.0718 0.0130  0.0588 0.0836 0.0234 0.0105  0.0210
14  2020 1      0.181   0.819  0.703  0.0604 0.00630 0.0489
15  2021 0.0759 0.0144  0.0615 0.0861 0.0245 0.00839 0.0236
16  2021 1      0.190   0.810  0.696  0.0587 0.00454 0.0511
17  2022 0.0835 0.0159  0.0676 0.0942 0.0264 0.00920 0.0272
18  2022 1      0.191   0.809  0.694  0.0575 0.00441 0.0528
19  2023 0.0810 0.0140  0.0670 0.0953 0.0223 0.00920 0.0243
20  2023 1      0.172   0.828  0.724  0.0506 0.00453 0.0482
```

每个年份，第一行是总体、区域间、区域内泰尔指数，各区域的（总体）泰尔指数；第二行是相应的贡献率，

各区域贡献率之和等于区域内贡献率，区域间贡献率 + 区域内贡献率 = 1。

3. 两交叉分组泰尔指数及其分解

批量计算每年 行业与 区域两交叉分组泰尔指数及其分解：

```
df |>
  reframe(Theil = theil_g2_cross(pick(everything()),
                                 group1 = " 行业", group2 = " 区域",
                                 y = " 平均工资", pop = " 就业人数"),
          .by = 年份) |>
  unnest_wider(Theil)
```

```
# A tibble: 20 x 14
    年份      T     Tb     Tw   Tw_b   Tw_w 信息业 制造业   医疗   商贸 建筑业
   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
 1  2014 0.0590 0.0304 0.0286 0.0129 0.0157 0.0612 0.0144 0.0369 0.0599 0.0108
 2  2014 1      0.515  0.485  0.219  0.266  0.0483 0.0902 0.0439 0.0689 0.0335
 3  2015 0.0609 0.0326 0.0283 0.0128 0.0155 0.0578 0.0152 0.0316 0.0595 0.0110
 4  2015 1      0.535  0.465  0.210  0.255  0.0471 0.0884 0.0395 0.0657 0.0312
 5  2016 0.0615 0.0331 0.0285 0.0129 0.0156 0.0584 0.0165 0.0294 0.0609 0.0109
 6  2016 1      0.537  0.463  0.209  0.253  0.0498 0.0916 0.0390 0.0661 0.0294
 7  2017 0.0644 0.0348 0.0295 0.0133 0.0163 0.0576 0.0163 0.0299 0.0618 0.0118
 8  2017 1      0.541  0.459  0.206  0.253  0.0517 0.0830 0.0410 0.0630 0.0294
 9  2018 0.0650 0.0350 0.0300 0.0127 0.0173 0.0577 0.0151 0.0315 0.0620 0.0118
10  2018 1      0.539  0.461  0.196  0.265  0.0565 0.0711 0.0440 0.0641 0.0302
11  2019 0.0663 0.0333 0.0330 0.0153 0.0178 0.0639 0.0157 0.0383 0.0604 0.0140
12  2019 1      0.502  0.498  0.231  0.268  0.0669 0.0668 0.0599 0.0635 0.0297
13  2020 0.0693 0.0333 0.0361 0.0169 0.0192 0.0680 0.0159 0.0376 0.0700 0.0150
14  2020 1      0.480  0.520  0.243  0.277  0.0751 0.0636 0.0581 0.0675 0.0287
15  2021 0.0731 0.0352 0.0380 0.0171 0.0209 0.0659 0.0164 0.0365 0.0692 0.0163
16  2021 1      0.481  0.519  0.233  0.286  0.0758 0.0639 0.0557 0.0652 0.0267
17  2022 0.0807 0.0411 0.0397 0.0187 0.0210 0.0670 0.0176 0.0375 0.0747 0.0190
18  2022 1      0.509  0.491  0.231  0.260  0.0741 0.0610 0.0536 0.0642 0.0259
19  2023 0.0783 0.0406 0.0377 0.0171 0.0206 0.0663 0.0157 0.0353 0.0694 0.0166
20  2023 1      0.519  0.481  0.218  0.263  0.0768 0.0552 0.0542 0.0639 0.0221
# i 3 more variables: 教育 <dbl>, 科研 <dbl>, 金融业 <dbl>
```

注意分解关系：

- 泰尔指数：T = Tb + Tw，Tw = Tw_b + Tw_w
- 贡献率：R_Tw = R_Tw_b + R_Tw_w = R_T_ 信息业 + ... + R_T_ 金融业

4. 两嵌套分组泰尔指数及其分解

仅用数据中有嵌套关系的 区域和 地区测试：

```
df |>
  reframe(Theil = theil_g2_nest(pick(everything()),
                                group1 = " 区域", group2 = " 地区",
                                y = " 平均工资", pop = " 就业人数"),
          .by = 年份) |>
  unnest_wider(Theil)
```

```
# A tibble: 20 x 40
    年份      T   Tb_g1  Tb_g2     Tw   东部     西部      东北     中部 东部_上海
   <dbl>  <dbl>   <dbl>  <dbl>  <dbl>  <dbl>   <dbl>     <dbl>   <dbl>     <dbl>
 1  2014 0.0609 0.00915 0.0234 0.0283 0.0374 0.00286 0.000441  0.00308    0.0547
 2  2014 1      0.150   0.384  0.465  0.367  0.00818 0.000425  0.00858    0.0573
 3  2015 0.0628 0.00900 0.0233 0.0305 0.0373 0.00321 0.000148  0.00331    0.0529
 4  2015 1      0.143   0.371  0.486  0.353  0.00909 0.000132  0.00904    0.0522
 5  2016 0.0633 0.00908 0.0230 0.0312 0.0369 0.00271 0.000187  0.00334    0.0505
 6  2016 1      0.143   0.364  0.493  0.347  0.00765 0.000155  0.00921    0.0499
 7  2017 0.0663 0.00939 0.0236 0.0334 0.0382 0.00203 0.000352  0.00267    0.0494
 8  2017 1      0.142   0.355  0.503  0.342  0.00553 0.000262  0.00702    0.0472
 9  2018 0.0670 0.00859 0.0253 0.0331 0.0407 0.00197 0.000638  0.00230    0.0455
10  2018 1      0.128   0.378  0.494  0.366  0.00528 0.000442  0.00589    0.0439
11  2019 0.0684 0.0119  0.0260 0.0305 0.0417 0.00194 0.000469  0.00237    0.0351
12  2019 1      0.174   0.380  0.446  0.369  0.00509 0.000313  0.00592    0.0361
13  2020 0.0718 0.0130  0.0272 0.0316 0.0436 0.00199 0.000969  0.00310    0.0430
14  2020 1      0.181   0.379  0.439  0.366  0.00514 0.000584  0.00723    0.0418
15  2021 0.0759 0.0144  0.0294 0.0321 0.0459 0.00234 0.000409  0.00473    0.0502
16  2021 1      0.190   0.387  0.423  0.371  0.00560 0.000221  0.0103     0.0501
17  2022 0.0835 0.0159  0.0314 0.0362 0.0489 0.00231 0.000311  0.00489    0.0522
18  2022 1      0.191   0.375  0.434  0.361  0.00503 0.000149  0.00948    0.0499
19  2023 0.0810 0.0140  0.0319 0.0351 0.0499 0.00254 0.0000470 0.00450    0.0490
20  2023 1      0.172   0.394  0.433  0.380  0.00578 0.0000231 0.00894    0.0486
# i 30 more variables: 西部_云南 <dbl>, 西部_内蒙古 <dbl>, 东部_北京 <dbl>,
#   东北_吉林 <dbl>, 西部_四川 <dbl>, 东部_天津 <dbl>, 西部_宁夏 <dbl>,
#   中部_安徽 <dbl>, 东部_山东 <dbl>, 中部_山西 <dbl>, 东部_广东 <dbl>,
#   西部_广西 <dbl>, 西部_新疆 <dbl>, 东部_江苏 <dbl>, 中部_江西 <dbl>,
#   东部_河北 <dbl>, 中部_河南 <dbl>, 东部_浙江 <dbl>, 东部_海南 <dbl>,
#   中部_湖北 <dbl>, 中部_湖南 <dbl>, 西部_甘肃 <dbl>, 东部_福建 <dbl>,
#   西部_西藏 <dbl>, 西部_贵州 <dbl>, 东北_辽宁 <dbl>, 西部_重庆 <dbl>, ...
```

注意分解关系：

- 泰尔指数：T = Tb_g1 + Tb_g2 + Tw
- 贡献率:R_Tb_g2 = R_ 东部 + ... + R_ 中部,R_Tw = R_ 东部 _ 上海 + ... + R_ 东北 _ 黑龙江

第十三章　系统评价

13.1　耦合协调度

耦合协调度，是分析事物协调发展水平的重要工具。其中，耦合度反映两个或多个系统间的相互作用与影响，体现系统间协调发展的动态关联关系，能够量化系统间的相互依赖与制约程度；协调度则衡量耦合关系中良性耦合的程度，反映协调状况的优劣。

在经济地理学研究中，耦合协调度模型应用广泛。该学科主要关注由多个子系统（如产业系统、技术创新系统、社会文化系统、生态环境系统等）构成的区域经济复杂系统。因此，该模型成为分析区域经济系统中不同子系统间关联性、协同性、协调性及其对区域经济发展影响的常用方法。

然而，区域经济中多系统间的耦合协调关系往往难以通过耦合协调度模型简单揭示。该模型仅能从宏观层面反映区域经济系统中不同子系统间的耦合协调程度，存在明显局限性：既无法深入分析各子系统内部的结构、功能和动力机制，也难以充分考虑多系统间的非线性关系、反馈机制和时滞效应等复杂因素。特别是在缺乏对两系统间关系明确认知的情况下，仅通过简单分析其共同发展态势难以获得深刻洞察，需要结合实际情况辩证运用这一方法。建议借助理论分析，深化系统间关系的机理阐释与理论模型构建。

耦合协调度共涉及 3 个指标值的计算，分别是耦合度 C 值，协调指数 T 值，耦合协调度 D 值。并且最终结合耦合协调度 D 值和协调等级划分标准，最终得出各项的耦合协调程度。

13.1.1　耦合度

耦合度的概念源自物理学，用来描述两个或多个系统之间的相互作用与关联性。在社会经济领域，耦合度被用来评估诸如基础设施与公共服务、经济发展与环境保护等不同系统之间的相互影响。

简单来说，耦合度越高，意味着两个系统之间的互动越强，彼此的促进作用越明显。例如，当农村基础设施的改善带动了公共服务水平的提升，反过来，公共服务的提高又促进了基础设施的进一步完善，这样的互动关系就会表现为一个较高的耦合度。

耦合度指两个或两个以上系统之间的相互作用影响，实现协调发展的动态关联关系，可以反映系统之间的相互依赖相互制约程度。n 个系统相互作用耦合度可以定义为：

$$C_n = \frac{(u_1 u_2 \cdots u_n)^{1/n}}{(u_1 + u_2 + \cdots + u_n)/n}$$

其中，u_i 为第 i 个子系统的标准化值，其分布区间为 $[0,1]$。C 值越大，子系统间离散程度越小，耦合度越高；反之，子系统间耦合度越低。

耦合度区间	耦合程度
[0, 0.3)	低度耦合
[0.3, 0.6)	中低度耦合
[0.6, 0.8)	中高度耦合
[0.8, 1.0]	高度耦合

耦合度取决于多个系统的发展程度及其相互关系。比如，当两个系统都发展得很好且相互促进时，耦合度接近于 1；反之，当两个系统中有一个发展滞后或彼此关联性弱时，耦合度会很低。

13.1.2 协调指数

协调指数，是各子系统标准化值的加权综合得分，反映系统整体的协调发展程度。其计算公式为：

$$T = \sum_{i=1}^{n} w_i u_i$$

其中，w_i 为各子系统的权重，满足 $\sum_{i=1}^{n} w_i = 1$。

13.1.3 耦合协调度

耦合度虽然能够反映系统间的互动强度，但无法全面描述系统的整体发展状况。因此，协调度模型应运而生。该模型不仅考量系统间的相互作用，更关注其是否实现和谐发展。

耦合协调度，是综合耦合度与协调指数的复合指标，定义为二者的几何平均：

$$D = \sqrt{CT}$$

耦合协调度的取值范围为 $[0,1]$，D 值越高，耦合协调程度越高，两系统发展越协调，反之亦然。

协调度区间	表示的意义
[0, 0.1)	极度失调衰退类
[0.1, 0.2)	严重失调衰退类
[0.2, 0.3)	中度失调衰退类
[0.3, 0.4)	轻度失调衰退类
[0.4, 0.5)	濒临失调衰退类
[0.5, 0.6)	勉强协调发展类
[0.6, 0.7)	初级协调发展类
[0.7, 0.8)	中级协调发展类

协调度区间	表示的意义
[0.8, 0.9)	良好协调发展类
[0.9, 1]	优质协调发展类

mathmodels 包提供了 coupling_degree() 函数计算耦合协调度，基本语法：

```
coupling_degree(data, w = NULL, id_cols = NULL)
```

- data 为已归一化的各子系统得分的数据框，可以包含不参与计算的 id_cols；
- w 为各系统的权重，默认为 NULL 表示等权重；
- id_cols 不参与计算的列名向量，默认为 NULL。

返回 tibble 包含 ID （若提供 id_cols），coupling（耦合度）、coord（协调指数）、coupling_coord（耦合协调度）。

13.2 障碍度

障碍度模型，旨在评估各指标或因素对系统协调发展的制约程度，识别影响协调水平提升的关键瓶颈。该模型常与耦合协调度模型结合使用，用于诊断协调水平偏低的内在原因，具有较强的解释力和实践指导价值。例如，在区域经济发展研究中，可通过障碍度分析判断经济活力不足、协调机制缺失，还是绿色发展滞后等因素构成主要制约。

作为耦合协调度模型的有益补充，障碍度模型有助于深入揭示系统内部的非均衡问题，为制定精准化、针对性的优化策略提供科学依据。

障碍度模型包含三个关键概念，用于识别制约系统协调发展的主要因素：

- **指标贡献度**：反映各指标对系统整体发展目标（如协调发展）的影响程度。通常由其权重体现，权重越高，表明该指标在系统中的作用越重要。例如，GDP 增长可能在经济子系统中具有较高的贡献度。
- **指标偏差度**：衡量指标实际表现与其理想状态之间的差距。标准化值越低（接近 0），表示实际水平与目标差距越大，偏差度越高。例如，若环境保护水平显著低于预期目标，则其偏差度较大。
- **障碍度**：综合考虑指标的**权重（贡献度）**和**与理想值的偏离程度（偏差度）**，量化其对系统协调发展的阻碍作用。障碍度越高，说明该指标是制约系统协调的关键瓶颈。

这些概念有助于研究者识别优先改进的领域，广泛应用于政策评估、区域规划与可持续发展分析。

障碍度模型通常基于多层级指标体系。设有 m 个一级指标，第 j 个一级指标下包含 m_j 个二级指标（注意：各一级指标下的二级指标数量可不同）。障碍度计算步骤如下：

- **二级指标 j 在一级指标 i 下的障碍度：**

$$O_{ij} = (1 - X_{ij}) \cdot w_{ij}$$

其中：

- $X_{ij} \in [0,1]$ 是第 ij 个指标的标准化值（且已正向化处理）；
- w_{ij} 是该指标的权重，通常通过熵值法、AHP 或主成分分析等方法确定；
- $(1 - X_{ij})$ 表示该指标与理想状态的偏离程度。

说明：障碍度本身是相对值，用于比较各指标的阻碍作用大小，一般不进行全局归一化，通过比较其数值大小判断影响程度。

- **一级指标 i 的总体障碍度：**

$$U_i = \sum_{j=1}^{m_i} O_{ij} = \sum_{j=1}^{m_i} (1 - X_{ij}) \cdot w_{ij}$$

即该一级指标下所有二级指标障碍度的加总，用于评估该子系统对整体协调发展的阻碍程度。

最终，可根据 O_{ij} 和 U_i 的大小排序，识别出最主要的障碍因素。

`mathmodels` 包提供了 `obstacle_degree()` 函数计算障碍度，基本语法：

```
obstacle_degree(data, w = NULL, id_cols = NULL)
```

参数同 `coupling_degree()` 函数。为了便于批量计算，该函数设计为只计算二级指标障碍度 O_{ij}，想进一步计算一级指标障碍度 U_i，在外面接着使用 `summarise(..., .by)` 即可。

13.3 案例：耦合协调度、障碍度

加载包：

```
library(tidyverse)
library(mathmodels)
```

使用模拟数据：中国东部沿海地区 8 个省市“科技-经济-生态”系统协调发展研究（2015–2024 年）

- 科技创新子系统：R_D 投入强度, 高新技术企业数, 专利授权量
- 经济发展子系统：GDP 总量, 人均可支配收入, 产业结构高级化
- 生态环境子系统：空气质量优良率, 单位 GDP 能耗, 绿地覆盖率

```
cdev = read_csv("data/cdev-sim.csv")
cdev
```

```
# A tibble: 80 x 11
   地区   年份 R_D投入强度 高新技术企业数 专利授权量 GDP总量 人均可支配收入
   <chr> <dbl>       <dbl>          <dbl>      <dbl>   <dbl>          <dbl>
 1 上海   2015         3.9           8820       17.7   26881          42655
 2 上海   2016        4.44           9551         18   29300          44731
 3 上海   2017        4.53          10000       20.9   31255          47753
 4 上海   2018        4.44          10406       23.3   33185          50151
 5 上海   2019        4.66          11044       24.3   34222          52690
 6 上海   2020        5.07          11561       26.8   36722          54781
```

```
 7 上海    2021        4.78          12146       29.8    38388          57669
 8 上海    2022        5.67          12769       30.9    40910          59499
 9 上海    2023        5.53          13561       33.8    42373          62583
10 上海    2024        5.15          14016       35.7    44758          65044
# i 70 more rows
# i 4 more variables: 产业结构高级化 <dbl>, 空气质量优良率 <dbl>,
#   单位GDP能耗 <dbl>, 绿地覆盖率 <dbl>
```

13.3.1 分别综合评价每个子系统

- 用熵权法分别综合评价每个子系统

```
tech = entropy_weight(cdev[3:5], rep("+",3))
eco = entropy_weight(cdev[6:8], rep("+",3))
env = entropy_weight(cdev[9:11], c("+","-","+"))
```

- 查看指标权重

```
rlt = list(tech, eco, env)
map(rlt, "w") |> unlist()
```

```
   R_D投入强度 高新技术企业数      专利授权量        GDP总量 人均可支配收入
     0.3215465      0.3414780      0.3369755      0.3098166      0.2481369
产业结构高级化 空气质量优良率    单位GDP能耗     绿地覆盖率
     0.4420465      0.3934835      0.3515999      0.2549167
```

- 计算三个子系统的综合得分

```
S = map_dfc(rlt, "s") |>
  set_names(" 科技", " 经济", " 生态")
S = bind_cols(cdev[1:2], S)
S
```

```
# A tibble: 80 x 5
   地区    年份   科技   经济   生态
   <chr> <dbl> <dbl> <dbl> <dbl>
 1 上海    2015  36.2  50.2  40.6
 2 上海    2016  41.9  52.7  42.2
 3 上海    2017  45.1  57.9  46.4
 4 上海    2018  46.4  60.7  55.4
 5 上海    2019  49.8  62.9  58.7
 6 上海    2020  55.4  65.8  61.3
 7 上海    2021  55.9  71.1  68.0
 8 上海    2022  64.5  72.8  76.1
```

```
 9 上海   2023  66.5  75.2  80.1
10 上海   2024  65.4  77.3  79.8
# i 70 more rows
```

13.3.2 批量计算耦合协调度

- 归一化子系统得分

```
S = S |>
  mutate(across(3:5, rescale))
```

- 按年份分组，分别计算耦合协调度

```
S |>
  nest(.by = 年份) |>
  mutate(model = map(data, \(d) coupling_degree(d, id_cols = " 地区"))) |>
  select(-data) |>
  unnest(model)
```

```
# A tibble: 80 x 5
    年份 地区  coupling  coord coupling_coord
   <dbl> <chr>    <dbl>  <dbl>          <dbl>
 1  2015 上海  0.965    0.466         0.671
 2  2015 北京  0.953    0.449         0.655
 3  2015 天津  0.00201  0.0245        0.00701
 4  2015 山东  0.00106  0.0940        0.00998
 5  2015 广东  0.987    0.349         0.587
 6  2015 江苏  0.786    0.193         0.389
 7  2015 浙江  0.951    0.302         0.536
 8  2015 福建  0.000820 0.260         0.0146
 9  2016 上海  0.970    0.507         0.701
10  2016 北京  0.971    0.510         0.703
# i 70 more rows
```

13.3.3 批量计算障碍度

- 按年份分组，分别计算二级指标障碍度

```
res = S |>
  nest(.by = 年份) |>
  mutate(model = map(data, \(d) obstacle_degree(d, id_cols = " 地区"))) |>
  select(-data) |>
```

```
  unnest(model)
res
```

```
# A tibble: 80 x 5
    年份 地区    科技   经济   生态
   <dbl> <chr> <dbl> <dbl> <dbl>
 1  2015 上海  0.204 0.117 0.213
 2  2015 北京  0.204 0.118 0.229
 3  2015 天津  0.325 0.333 0.317
 4  2015 山东  0.320 0.253 0.333
 5  2015 广东  0.194 0.217 0.240
 6  2015 江苏  0.265 0.227 0.316
 7  2015 浙江  0.265 0.245 0.188
 8  2015 福建  0.333 0.281 0.125
 9  2016 上海  0.180 0.107 0.207
10  2016 北京  0.174 0.108 0.208
# i 70 more rows
```

- 若要查看每年 科技前三大的障碍度

```
res |>
  slice_max(科技, n = 3, by = 年份)
```

```
# A tibble: 30 x 5
    年份 地区    科技   经济    生态
   <dbl> <chr> <dbl> <dbl>  <dbl>
 1  2015 福建  0.333 0.281 0.125
 2  2015 天津  0.325 0.333 0.317
 3  2015 山东  0.320 0.253 0.333
 4  2016 天津  0.324 0.327 0.313
 5  2016 福建  0.320 0.280 0.114
 6  2016 山东  0.304 0.238 0.317
 7  2017 天津  0.325 0.318 0.294
 8  2017 山东  0.310 0.223 0.306
 9  2017 福建  0.289 0.261 0.0979
10  2018 天津  0.304 0.313 0.272
# i 20 more rows
```

- 若要查看每个子系统前三大的障碍度，先宽变长

```
res |>
  pivot_longer(科技:生态, names_to = " 子系统", values_to = " 障碍度") |>
  slice_max(障碍度, n = 3, by = c(年份, 子系统))
```

```
# A tibble: 90 x 4
    年份 地区  子系统 障碍度
   <dbl> <chr> <chr>   <dbl>
 1  2015 福建  科技    0.333
 2  2015 天津  科技    0.325
 3  2015 山东  科技    0.320
 4  2015 天津  经济    0.333
 5  2015 福建  经济    0.281
 6  2015 山东  经济    0.253
 7  2015 山东  生态    0.333
 8  2015 天津  生态    0.317
 9  2015 江苏  生态    0.316
10  2016 天津  科技    0.324
# i 80 more rows
```

- 批量计算一级指标障碍度 U_i，即按年份分组汇总列和

```
res |>
  summarise(across(科技:生态, sum), .by = 年份)
```

```
# A tibble: 10 x 4
    年份  科技  经济  生态
   <dbl> <dbl> <dbl> <dbl>
 1  2015  2.11 1.79  1.96
 2  2016  1.91 1.70  1.86
 3  2017  1.86 1.58  1.70
 4  2018  1.71 1.47  1.56
 5  2019  1.60 1.37  1.46
 6  2020  1.50 1.28  1.34
 7  2021  1.38 1.17  1.20
 8  2022  1.26 1.07  1.00
 9  2023  1.18 0.968 0.954
10  2024  1.12 0.865 0.821
```

参考文献

[1] 张敬信等. 数学建模：算法与编程实现 [M]. 北京: 机械工业出版社, 2022.