基于 R 语言的 CATA 与 TCATA 分析

李嘉伟

目录

第一部分 Check-All-That-Apply(CATA) 4

1 数据准备 4
1.1 原始数据格式说明 . . . 4
1.2 数据导入 . . . 4
1.2.1 导入原始数据 . . . 4
1.2.2 指定分析数据 . . . 4
1.2.3 有无理想点 . . . 5

2 Cochran's Q test for each attribute 5
2.1 R 包的准备 . . . 5
2.2 基于每个属性的全体产品显著性检验 . . . 6
2.2.1 定义全局 Q 检验函数 . . . 6
2.2.2 计算各属性全局检验 P 值 . . . 7
2.3 多重成对比较 (Multiple pairwise comparisons) . . . 7
2.3.1 基于 Critical difference (Sheskin) 多重比较 . . . 7
2.3.2 基于 McNemar(Bonferroni) 多重成对比较 . . . 10

3 列联表 (Contingency table) 的构建与检验 12
3.1 构建列联表 . . . 12
3.2 列联表的独立性检验 . . . 12

4 对应分析 (Correspondence analysis) 13
4.1 R 包的准备 . . . 13
4.2 特征值和惯性百分比 (Eigenvalues and percentages of inertia) . . . 14
4.3 碎石图 (Scree plot) . . . 15
4.3.1 方法一 . . . 15
4.3.2 方法二 . . . 16
4.4 对称图 (Symmetric plot) . . . 17
4.4.1 方法一 . . . 17
4.4.2 方法二 . . . 19
4.5 R 包 factoextra 简单介绍 . . . 21
4.5.1 fviz_ca_biplot() . . . 21
4.5.2 fviz_ca_col() . . . 21
4.5.3 fviz_ca_row() . . . 23

5 主坐标分析 (Principal Coordinate Analysis) 24
5.1 使用数据说明 . . . 25
5.2 R 包的准备 . . . 25
5.3 相关系数表 (Correlations) . . . 25
5.3.1 四分相关系数 (tetrachoric correlation) . . . 25
5.3.2 点二列相关系数 (tetrachoric correlation) . . . 26
5.3.3 相关系数全表 . . . 26
5.4 主坐标分析可视化 . . . 28
5.4.1 碎石图 (Scree plot) . . . 28
5.4.2 对称图 (Symmetric plot) . . . 29

6 惩戒分析 (Penalty analysis) 31
6.1 R 包准备 . . . 31
6.2 有理想点数据 . . . 31
6.2.1 所有属性分析 (Analysis of attributes) . . . 32
6.2.1.1 必读说明 . . . 32
6.2.1.2 汇总表 (Summary table) . . . 32
6.2.1.3 百分比柱形图 . . . 33
6.2.1.4 对比表 (Comparison table) . . . 35
6.2.1.5 平均影响显示 (Mean impact display) . . . 37
6.2.1.6 均值与百分比散点图 (Mean drops vs %) . . . 39
6.2.2 汇总分析 (Penalty analysis summary) . . . 41
6.2.3 属性分析 (Attributes analysis:) . . . 42
6.2.3.1 属性四格表 . . . 42
6.2.3.2 属性分类汇总结果 . . . 43
6.2.3.3 诱发率比较 (Comparison of elicitation rates) . . . 44
6.3 无理想点数据 . . . 45
6.3.1 数据导入 . . . 45
6.3.2 汇总表 (Summary table) . . . 45
6.3.3 百分比柱形图 . . . 45
6.3.4 对比表 (Comparison table) . . . 46
6.3.5 平均影响显示 (Mean impact display) . . . 48
6.3.6 均值与百分比散点图 (Mean drops vs %) . . . 49

7 文献分析方法补充 50
7.1 凝聚层次聚类 (Agglomerative Hierarchical Clustering) . . . 50
7.1.1 R 包的准备 . . . 50
7.1.2 聚类可视化 . . . 51
7.2 组合条形图 . . . 53
7.2.1 数据整理 . . . 53
7.2.2 绘图 . . . 53
7.3 惩罚提升分析 (Penalty-Lift Analysis) . . . 55
7.3.1 数据说明 . . . 55
7.3.2 数据整理 . . . 55
7.3.3 绘图 . . . 55
7.4 相关矩阵图 . . . 56
7.4.1 数据说明 . . . 56
7.4.2 数据处理于绘图 . . . 56

7.5 马赛克图 58

第二部分 Temporal Check-All-That-Apply(TCATA) 60

8 数据准备 60
8.1 原始数据格式说明 60
8.2 数据导入 60

9 汇总表 (Summary table) 60

10 引用比例条形图 (Bar chart of citation proportions) 61
10.1 单图绘制 61
10.2 多图汇总 62

11 产品曲线 (Curves by product) 64
11.1 必读说明 64
11.2 曲线绘图 64
11.3 显著性差异 (Significative differences of citation proportions) 65

12 对应分析 (Correspondence Analysis) 66
12.1 必读说明 66
12.2 R 包的准备 66
12.3 数据整理 66
12.4 特征值 (Eigenvalues) 67
12.5 碎石图 (Scree plot) 67
12.5.1 方法一 67
12.5.2 方法二 69
12.6 产品轨迹图 (Product trajectories) 70
12.7 R 包 factoextra 可视化简单示例 71

13 文献分析方法补充 73
13.1 主成分分析 (PCA) 73
13.2 多重因子分析 (MFA) 76

第一部分　Check-All-That-Apply(CATA)

1　数据准备

1.1　原始数据格式说明

本次 R 语言对 CATA 的分析对 CATA 原始数据格式有一定的要求：

- 原始数据整体与老师所发的 **CATA 实例**文件相同
- 需要注意的是：
  - 评估者或消费者所在列的变量命名必须为 **Consumer**
  - 产品所在列的变量命名必须为 **Products**
  - 喜好得分所在列变量命名必须为 **Liking**
  - 所有属性列一定要都在前面所述三列的右边，对所有属性列之间的顺序并无要求
  - 除此之外的其他列，如标签列等则与前面三列放在一起，位于所有属性列的左边
  - **Liking** 列必须在从左到右第一个属性列的前面一列
  - 若原始数据含有理想点数据，则名称必须为 **Ideal**
  - 总体上，最好与老师所发数据一致
- 由于 R 语言导入 CSV 数据格式最为方便，所以如果原始数据是 xls 格式，则需导出为 csv 格式，从而将 csv 格式文件导入到 R 中成为数据框格式，之后对其做进一步处理

1.2　数据导入

1.2.1　导入原始数据

导入数据只需运行以下代码。需要说明的是：

- 双引号内部是 csv 文件的**绝对路径**, 在哪一台电脑上导入文件，则就需要用该电脑上该文件的绝对路径，也就是说以下代码只能在本人电脑上实现，不同电脑只需修改文件的绝对路径即可；
- windows 中文件的绝对路径一般是以**反斜杠**连接，在 R 中则需要用**两个反斜杠**连接，除此之外也可以用**一个斜杠**，如：“C:/Users/里高房价/Desktop/CATA&TCATA/CATA/CATA 实例.csv”, 两种写法都可以。
- 如果该数据文件与您运行以下 R 代码的 R 文件在同级目录，则双引号内部可以只需要写文件名 (包括后缀名)。

```
# 导入数据
data <- read.csv("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\CATA 实例.csv")
```

1.2.2　指定分析数据

有些时候因为获取的数据质量问题，只对某些属性数据进行分析，有两种方法可以实现：

- 对于需要分析的数据**列数不多**的情况，我们只需要去数用于分析的数据的每一列是第几列即可。如下代码的含义是：选举 *data* 数据里的第 *1*，*2*，*3*，*4*，*5*，*6*，*7*，*9*，*10* 列，所有行用于分析，并替换原来的 *data* 数据；

```
# 指定需要分析的数据
#data 为一个数据框，也就是类似于原始 xls 数据里面的格式
# 中括号里面，逗号前面是行索引，逗号后面是列索引
# 逗号前面为空表示默认选取所有行
# 符号"<-" 是 R 中特有的赋值符："a<-1" 的意思是将 1 赋给变量 a
# 符号"=" 与符号"<-" 等价，即也可以使用"a=1" 进行赋值
```

```

data<-data[,c(1,2,3,4,5,6,7,9,10)]
```

- 对于只有**很少的列不需要分析**的情况，我们也只需要数不用于分析的数据的每一列是第几列即可。如下代码的含义是：除了第 *8* 列和第 *14* 列，其他所有的列以及所有行我都要分析，并替换为新的 *data*。

```
# 指定需要分析的数据
#c() 的含义是将目标转化为一个向量，如：c(1,2) 就是 1 和 2 构成的一个一维向量
# 前面加一个负号表示去除列，即除了这些列之外，其他的列我都要拿去分析

data<-data[,-c(8,14)]
```

1.2.3 有无理想点

一般原始数据有两种：有理想点数据和无理想点数据。如果原始数据不含理想点数据，则只需按上述两种方法操作即可；若原始数据含有理想点数据，但是有些分析方法不需要理想点数据，则可以运行以下代码，意味着：只选取 Product 列中不为 “Ideal” 的所有行以及所有列。

```
#"$" 美元符号在 R 中代表取某列
# 如 data$Product 代表选取 data 数据框里的名字为 Product 的列
# 返回的是该列所有的值，长度为该列的行数
#!= 在 R 中指的是表示 " 不等于 " 的逻辑判断符号

data<-data[data$Product!="Ideal",]
```

由于 CATA 方法中，有些方法的实现的前提是必须含有理想点数据，所以在提到这些方法时会着重说明仅适用于含有理想点数据；对于除此之外的方法，一般不需要理想点数据，提到时会加以说明。

2 Cochran’s Q test for each attribute

注意：此方法所需数据对理想点数据无要求，下面以前面已经导入的无理想点数据为例。

2.1 R 包的准备

在实现最终汇总结果前，需要使用两个 R 包：

- **DescTools**：使用该包的 `CochranQTest` 函数进行 Q 检验
- **reshape2**：使用该包的 `melt` 与 `dcast` 函数，对数据进行整合与重构 (此包后面我们会经常用到，是 R 中一个非常强大的 R 包)。

如果您的 R 中未安装这两个包，运行以下代码即可安装。

```
install.packages("DescTools")
install.packages("reshape2")
```

2.2 基于每个属性的全体产品显著性检验

Cochran 的 Q 检验是用来检验每个属性的所有产品是否有显著性差异。为了能够得到我们想要的如下汇总格式，进行全局检验之前，我们需要做一些准备工作。

Attributes	p-values	C18	C19	C3	C4	C7	N1	N4
Sticky (黏	0.0000	0.313 (bc)	0.209 (ab)	0.582 (de)	0.478 (cd)	0.746 (e)	0.806 (e)	0.030 (a)
Caramel fla	0.0045	0.388 (a)	0.239 (a)	0.254 (a)	0.254 (a)	0.328 (a)	0.463 (a)	0.463 (a)
Brown (褐	0.0178	0.493 (ab)	0.537 (b)	0.328 (a)	0.493 (ab)	0.463 (ab)	0.358 (ab)	0.448 (ab)
Bubble (气	0.0000	0.104 (a)	0.194 (ab)	0.612 (d)	.224 (abc)	0.433 (cd)	0.343 (bc)	0.015 (a)
Sweet sme	0.2430	0.299 (a)	0.328 (a)	0.358 (a)	0.299 (a)	0.448 (a)	0.328 (a)	0.418 (a)
Jujube flavc	0.0000	0.179 (a)	0.134 (a)	0.045 (a)	0.075 (a)	0.119 (a)	0.119 (a)	0.716 (b)
Milk flavor	0.0000	0.537 (b)	0.478 (b)	0.642 (b)	0.507 (b)	0.567 (b)	0.552 (b)	0.179 (a)
Grain flavo	0.0000	0.149 (a)	0.075 (a)	0.060 (a)	0.045 (a)	0.045 (a)	0.149 (a)	0.448 (b)
Acid fragra	0.0000	0.731 (bc)	.642 (abc)	0.791 (c)	.672 (abc)	0.821 (c)	0.507 (ab)	0.448 (a)
Fruity (果	0.0000	0 (a)	0.104 (a)	0.045 (a)	0.045 (a)	0.030 (a)	0.060 (a)	0.239 (b)
Coffee flav	0.0000	0.209 (a)	0.104 (a)	0.045 (a)	0.045 (a)	0.030 (a)	0.209 (a)	0.403 (b)
Caramel cc	0.0000	0.687 (c)	0.687 (c)	0.403 (ab)	0.582 (bc)	0.597 (bc)	0.358 (a)	.507 (abc)
Sour (酸味	0.0005	0.731 (ab)	0.716 (ab)	0.746 (ab)	0.851 (b)	0.761 (b)	0.552 (a)	0.657 (ab)
Sweet (甜	0.1416	0.567 (a)	0.672 (a)	0.612 (a)	0.507 (a)	0.657 (a)	0.627 (a)	0.522 (a)
Fermented	0.0058	0.403 (ab)	0.507 (b)	0.343 (ab)	0.403 (ab)	0.358 (ab)	0.343 (ab)	0.224 (a)
Smoothnes	0.0033	0.567 (ab)	0.687 (b)	0.537 (ab)	0.537 (ab)	0.552 (ab)	0.358 (a)	0.537 (ab)
Astringent	0.7648	0.149 (a)	0.164 (a)	0.119 (a)	0.149 (a)	0.179 (a)	0.134 (a)	0.090 (a)
Bitter (苦	0.0012	0.149 (ab)	0.030 (a)	0.060 (ab)	0.045 (ab)	0.030 (a)	0.179 (b)	0.149 (ab)
Chinese he	0.0000	0.134 (ab)	0.045 (a)	0 (a)	0.075 (a)	0.030 (a)	0.254 (b)	0.119 (ab)

最终汇总部分结果

2.2.1 定义全局 Q 检验函数

```
#R 中，调用 R 包，一般使用 library 函数，参数为包的名字
library(DescTools)# 调用此 R 包，使用该包的 CochranQTest 函数
library(reshape2)

## 定义 Cochran Test 检验函数
#attr 参数是需要检验的属性名
#data 参数是需要分析的数据
#digit 参数为结果保留小数位数
fun_QTest<-function(attr,data,digit){
  data_attr = melt(data,id=c("Consumer","Product"),measure=attr)
  test <- CochranQTest(value~Product | Consumer,data=data_attr)
  if(is.nan(test$statistic)){
    return(list(p.value=1.0,chisq.value=10^10))
  }
  else{
    return(list(p.value=round(test$p.value,digit),
                chisq.value=round(test$statistic,digit)))
  }
}
```

在实际使用中，您可以不必每次复制这个函数定义代码再重新运行一遍，您只需运行以下一行代码即可。运行之后，您就可以调用这个函数。

```
# 注意:
# 双引号内部的绝对路径为当前电脑的该 R 文件路径，您需要修改
# 该 R 文件即为上述 fun_QTest 的函数源代码
# 一定要提前找到该 R 文件的位置
#source 函数的意义是根据该 R 文件的位置找到该文件，再从上到下运行一遍

source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\fun_QTest.R")
```

2.2.2 计算各属性全局检验 P 值

运行以下代码即可计算出各属性全局检验 P 值，并保存在变量 p_vec 中。

需要修改参数如下:

- **index**: 这个参数为 data 数据中从左到右第一个属性所在列的列索引

```
## 进行 CochranQTest 检验，并计算 P 值
index=5                                          # 对应不同数据，index 会有所不同
attr_all<- colnames(data)[index:ncol(data)]      # 提取所有属性名
p_vec <- vector()
for(attr in attr_all){
  #p_vec 为各属性 CochranQTest 检验的 P 值
  p_vec <- c(p_vec,fun_QTest(attr,data,digit=4)$p.value)# 此处调用了自定义函数
}
```

2.3 多重成对比较 (Multiple pairwise comparisons)

在进行全局检验之后，如果全局检验显著只能说明该属性下，所有产品并不是都显著无差异，于是我们需要知道哪些有差异，哪些无差异。在 XLSTAT 中有两种成对比较方法，接下来我们一一实现。

2.3.1 基于 Critical difference (Sheskin) 多重比较

自定义 Marascuilo_MPC 函数, 该函数用于构建基于 Critical difference (Sheskin) 多重比较的汇总表，函数源代码如下。

```
## 多重比较函数
#data: 原始数据
#digit: 保留小数位数
#alpha: 犯第一类错误概率即显著性水平
#index: 第一个属性列所在的列索引

Marascuilo_MPC<-function(data,digit,alpha,index){
  #data: 原始数据，digit: 保留位数,alpha: 犯第一类错误概率即显著性水平
  #index: 第一个属性列所在的列索引
  attr_all<- colnames(data)[index:ncol(data)]# 保存所有属性名
  # 计算样本数
  n_sample<-aggregate(data[index:ncol(data)],by=list(Product=data$Product),sum)
```

```r
# 计算样本比例
p_sample<-round(n_sample[,-1]/length(unique(data$Consumer)),digit)
# 计算各属性下产品比率
p_table<-t(p_sample)
product <- unique(data$Product)# 将产品名称单独拿出
colnames(p_table)<-product

# 进行多重成对比较并进行字母显著性标记
sig_mark<-matrix(nrow=nrow(p_table),ncol=ncol(p_table))
for(k in 1:length(attr_all)){
  i=1
  n=1
  u=2
  n_product<-length(unique(data$Product))
  # 计算 CD(Critical Difference)
  z_adj<-qnorm(alpha/(n_product*(n_product-1)),lower.tail=FALSE)
  k_<-n_product
  n_all<-length(unique(data$Consumer))
  table<-aggregate(data[attr_all[k]],by=list(data$Consumer),sum)
  T_<-sum(table[attr_all[k]])
  R<-apply(table[attr_all[k]],2,function(x){sum(x^2)})
  CD<-z_adj*sqrt(2*(k_*T_-R)/(n_all^2*k_*(k_-1)))
  # 将各产品比率由大到小排序
  p_order<-order(p_sample[attr_all[k]])
  # 定义存储字母标记矩阵
  sig_mark[k,p_order[i]]<-letters[n]
  while(u<=n_product){
    keep<-vector()
    p_i<-p_sample[p_order[i],attr_all[k]]
    flag=FALSE
    for(j in u:n_product){
      p_j<-p_sample[p_order[j],attr_all[k]]# 样本率
      if(abs(p_i-p_j)<=CD){sig_mark[k,p_order[j]]<-letters[n]}
      else{
        keep<-c(keep,j)
        sig_mark[k,p_order[j]]<-letters[n+1]
        flag=TRUE
        break
      }
    }
    if(flag==TRUE){
      for(w in 1:(j-1)){
        p_w<-p_sample[p_order[w],attr_all[k]]
        if(abs(p_w-p_j)<=CD){
          keep<-c(keep,w)
          sig_mark[k,p_order[w]]<-paste(sig_mark[k,p_order[w]],letters[n+1],sep="")
        }
```

```
        }
      }
      if(length(keep)==1){i<-keep[1]}
      else if(length(keep)>1){i<-keep[2]}
      else{break}
      n<-n+1
      u<-keep[1]+1
    }
  }
  # 将字母标记表与产品比率表整合
  for(i in 1:nrow(p_table)){
    for(j in 1:ncol(p_table)){
      p_table[i,j]<-paste(p_table[i,j],"(",sig_mark[i,j],")",sep="")
    }
  }
  return(MPC_table=p_table)
}
```

同理，我将这个函数源代码打包成一个 R 文件，便于一行代码即可调用该函数。最后，只需运行下面的代码，即可实现最终检验汇总结果。

可修改参数有:

- **data**: 需要分析的数据
- **digit**: 结果保留小数位数
- **alpha**: 犯第一类错误概率即显著性水平
- **index**: 所分析数据从左到右第一个属性列所在的列索引

```
# 运行 Marascuilo_MPC 函数 R 文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Marascuilo_MPC.R")

## 合并 P 值与多重比较表格
MPC_table<-Marascuilo_MPC(data=data,digit=3,alpha=0.05,index=5)
rownames(MPC_table)<-1:nrow(MPC_table)
Cochran_table<-cbind(data.frame(Attributes=attr_all,p_values=p_vec),MPC_table)

# 由于 Cochran_table 汇总表行数太多，我们只取前 10 行查看汇总效果
Cochran_table[1:10,]   #1:10 代表 1,2,3,...,10 组成的向量
```

Attributes	p_values	C18	C19	C3	C4	C7	N1	N4
Sticky	0.0000	0.313(bc)	0.209(ab)	0.582(de)	0.478(cd)	0.746(e)	0.806(e)	0.03(a)
Caramel	0.0045	0.388(a)	0.239(a)	0.254(a)	0.254(a)	0.328(a)	0.463(a)	0.463(a)
Brown	0.0178	0.493(ab)	0.537(b)	0.328(a)	0.493(ab)	0.463(ab)	0.358(ab)	0.448(ab)
Bubble	0.0000	0.104(a)	0.194(ab)	0.612(d)	0.224(abc)	0.433(cd)	0.343(bc)	0.015(a)
Sweet	0.2430	0.299(a)	0.328(a)	0.358(a)	0.299(a)	0.448(a)	0.328(a)	0.418(a)
Jujube	0.0000	0.179(a)	0.134(a)	0.045(a)	0.075(a)	0.119(a)	0.119(a)	0.716(b)
Milk	0.0000	0.537(b)	0.478(b)	0.642(b)	0.507(b)	0.567(b)	0.552(b)	0.179(a)
Grain	0.0000	0.149(a)	0.075(a)	0.06(a)	0.045(a)	0.045(a)	0.149(a)	0.448(b)

Attributes	p_values	C18	C19	C3	C4	C7	N1	N4
Acid	0.0000	0.731(bc)	0.642(abc)	0.791(c)	0.672(abc)	0.821(c)	0.507(ab)	0.448(a)
Fruity	0.0000	0(a)	0.104(a)	0.045(a)	0.045(a)	0.03(a)	0.06(a)	0.239(b)

2.3.2 基于 McNemar(Bonferroni) 多重成对比较

该方法是 XLSTAT 的成对比较方法中的第二种方法。由于没有该方法的结果示例，所以最后呈现的汇总结果是以两两比较的形式，而没有使用第一种方法所使用的字母标记法。首先我们自定义汇总检验函数 `McNemar_MPC` 如下。

```
#data: 原始数据
#digit: 保留小数位数
#alpha: 犯第一类错误概率即显著性水平
#index: 第一个属性列所在的列索引

McNemar_MPC<-function(data,digit=3,alpha,index){
  attr_all<- colnames(data)[index:ncol(data)]# 保存所有属性名
  n_product<-length(unique(data$Product))
  # 定义矩阵存储 p 值
  pvalue_matrix<-matrix(nrow=length(attr_all),ncol=n_product*(n_product-1)/2)
  colname<-vector()
  for(k in 1:length(attr_all)){
    n_product<-length(unique(data$Product))
    n=1
    for(i in 1:(n_product-1)){
      for(j in (i+1):n_product){
        data_attr = melt(data,id=c("Consumer","Product"),measure=attr_all[k])
        n11<-sum(data_attr[data_attr$Product==data$Product[i],]$value==1
                 & data_attr[data_attr$Product==data$Product[j],]$value==1)
        n12<-sum(data_attr[data_attr$Product==data$Product[i],]$value==1
                 & data_attr[data_attr$Product==data$Product[j],]$value==0)
        n21<-sum(data_attr[data_attr$Product==data$Product[i],]$value==0
                 & data_attr[data_attr$Product==data$Product[j],]$value==1)
        n22<-sum(data_attr[data_attr$Product==data$Product[i],]$value==0
                 & data_attr[data_attr$Product==data$Product[j],]$value==0)
        nij_matrix<-matrix(c(n11,n12,n21,n22),byrow=TRUE,nrow=2)
        #Bonferroni 校正
        alpha_adj<-alpha/(n_product*(n_product-1))
        pvalue<-mcnemar.test(nij_matrix)$p.value
        if(is.nan(pvalue)){pvalue<-1.0}
        if(alpha_adj>pvalue){mask<-"(S)"}
        else{mask<-"(N)"}
        pvalue_matrix[k,n]<-paste(round(mcnemar.test(nij_matrix)$p.value,
                                        digit),mask,sep="")
        n<-n+1
        if(k==1){
          colname<-c(colname,paste(data$Product[i],"vs",data$Product[j]))
```

```
        }
      }
    }
  }
  p_value<-data.frame(pvalue_matrix,row.names=attr_all)
  colnames(p_value)<-colname
  return(p_value)
}
```

同理，我将这个函数源代码打包成一个 R 文件，便于一行代码即可调用该函数。最后，只需运行下面的代码，即可实现最终检验汇总结果。

可修改参数有:

- **data**: 需要分析的数据
- **digit**: 结果保留小数位数
- **alpha**: 犯第一类错误概率即显著性水平
- **index**: 所分析数据从左到右第一个属性列所在的列索引

```
# 运行 McNemar_MPC 函数 R 文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\McNemar_MPC.R")

## 合并 P 值与多重比较表格
MPC_table<-McNemar_MPC(data=data,digit=3,alpha=0.05,index=5)
rownames(MPC_table)<-1:nrow(MPC_table)
Cochran_table<-cbind(data.frame(Attributes=attr_all,p_values=p_vec),MPC_table)

# 由于 Cochran_table 汇总表行数与列数太多，我们只取前 10 行与前 9 列查看汇总效果
#1:10 前 10 行
#1:9 代表前 9 列
Cochran_table[1:10,1:9]
```

Attributes	p_values	C3 vs C4	C3 vs C7	C3 vs C18	C3 vs C19	C3 vs N1	C3 vs N4	C4 vs C7
Sticky	0.0000	0.248(N)	0.072(N)	0.003(N)	0(S)	0.009(N)	0(S)	0.002(N)
Caramel	0.0045	1(N)	0.302(N)	0.095(N)	1(N)	0.03(N)	0.026(N)	0.332(N)
Brown	0.0178	0.01(N)	0.081(N)	0.022(N)	0.002(N)	0.789(N)	0.043(N)	0.838(N)
Bubble	0.0000	0(S)	0.038(N)	0(S)	0(S)	0.001(S)	0(S)	0.014(N)
Sweet	0.2430	0.502(N)	0.307(N)	0.453(N)	0.85(N)	0.814(N)	0.54(N)	0.044(N)
Jujube	0.0000	0.617(N)	0.182(N)	0.008(N)	0.149(N)	0.131(N)	0(S)	0.546(N)
Milk	0.0000	0.095(N)	0.404(N)	0.19(N)	0.063(N)	0.307(N)	0(S)	0.522(N)
Grain	0.0000	1(N)	1(N)	0.149(N)	1(N)	0.149(N)	0(S)	1(N)
Acid	0.0000	0.17(N)	0.803(N)	0.522(N)	0.078(N)	0.001(S)	0(S)	0.055(N)
Fruity	0.0000	1(N)	1(N)	0.248(N)	0.221(N)	1(N)	0.002(N)	1(N)

上述汇总表中，从第三列开始，每列都是两两产品进行显著性比较。N 代表差异不显著，S 代表差异显著。标记前面的数值代表调整前的 McNemar 检验 P 值。

3 列联表 (Contingency table) 的构建与检验

注意：此方法所需数据对是否有理想点数据无要求。我们以有理想点数据为例，由于在进行 Cochran 检验时使用的是无理想点的数据，所以我们需要重新导入原始数据，以便使理想点数据被包含在我们分析数据之中：

```
# 导入数据
data <- read.csv("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\CATA 实例.csv")
data<-data[,1:72]
```

3.1 构建列联表

列联表只需一行代码即可构建，代码如下。可修改的参数只有：

- **index**: 所分析数据从左到右第一个属性列所在的列索引

```
## 构建汇总列联表
index=5
Contingency.table <-aggregate(data[,-c(1:(index-1))],
                by=list(Product=data[,"Product"]),sum,na.rm=TRUE)

# 由于 Contingency.table 列联表列数太多，我们只取前 9 列查看汇总效果
Contingency.table[,1:9]
```

Product	Sticky	Caramel	Brown	Bubble	Sweet	Jujube	Milk	Grain
C18	21	26	33	7	20	12	36	10
C19	14	16	36	13	22	9	32	5
C3	39	17	22	41	24	3	43	4
C4	32	17	33	15	20	5	34	3
C7	50	22	31	29	30	8	38	3
Ideal	50	31	14	1	47	7	60	10
N1	54	31	24	23	22	8	37	10
N4	2	31	30	1	28	48	12	30

3.2 列联表的独立性检验

二维列联表独立性检验一般使用卡方检验或者 Fisher 精确检验，但是卡方检验具有局限性：

- 进行 RxC 列联表卡方检验的要求是：
  1. R×C 表中理论数小于 5 的格子不能超过 1/5；
  2. 不能有小于 1 的理论数; 如果不满足上述两个条件，则不能用卡方检验，可以用 Fisher 精确检验。

两种检验的实现代码如下, 只需要 Contingency.table 作为参数即可。

```
#Contingency.table[,-1] 表示去掉第一列，因为第一列是产品名称
# 卡方检验
#correct=TRUE 表示做连续性修正
chisq.test(Contingency.table[,-1],correct=TRUE)
```

```
##
```

```
##  Pearson's Chi-squared test
##
## data:  Contingency.table[, -1]
## X-squared = 1116, df = 469, p-value < 2.2e-16
```

```r
#Fisher 精确检验
#simulate.p.value=TRUE 表示估计 p 值
fisher.test(Contingency.table[,-1],simulate.p.value=TRUE)
```

```
##
##  Fisher's Exact Test for Count Data with simulated p-value (based on
##  2000 replicates)
##
## data:  Contingency.table[, -1]
## p-value = 0.0004998
## alternative hypothesis: two.sided
```

结果说明:

- 可以明显看到卡方检验的 P 值为 NA，这是因为由于 Contingency.table 中含有大量的 0, 所以并不适合用卡方检验;
- 关于卡方检验自由度之所以和老师给的示例自由度不同，这是因为老师所给的数据是取了 68 个属性做卡方独立性检验，而本人使用了全部的 115 个属性，所以会有不同。

如果想对特定的一些属性做独立性检验，则只需选取在列联表中这些属性的所有列索引即可，示例如下:

```r
# 假设对第 1, 2, 3, 4, 5, 7, 9 列属性进行独立性检验
table<-Contingency.table[,c(1,2,3,4,5,7,9)]
chisq.test(table,correct=TRUE)
```

4 对应分析 (Correspondence analysis)

注意：此方法分析数据中理想点数据可有可无，且所使用代码完全一致。为了与模板数据一致，我使用带有理想点数据的分析数据进行分析，并且此方法需要用到前面构建的列联表数据。

4.1 R 包的准备

在正式开始对应分析之前，我们需要使用两个 R 包 FactoMineR 与 factoextra:

- **FactoMineR**：该 R 包提供了很多因子分析的方法，比如 CA,PCA 等等;
- **factoextra**：该 R 包是 FactoMineR 的可视化扩展包，用于可视化分析结果。

如果您还未安装此包，请运行以下代码:

```
install.packages("FactoMineR")
install.packages("factoextra")
```

4.2 特征值和惯性百分比 (Eigenvalues and percentages of inertia)

由于老师所给的示例中,只利用了前 68 个属性构建列联表,所以为了保持一致性,选取 `Contingency.tabl` 的前 69 列（包括第一列产品名称）用来做对应分析。

```
Contingency.table<-Contingency.table[,1:69]
```

运行以下代码，得到特征值和惯性百分比列联表。

可修改参数为:

- **digit**: 结果保留小数位数

```
# 加载 R 包
library(FactoMineR)  # 主要使用该包的 CA 函数
library(factoextra)  # 使用该包的可视化函数

#ca_table 为用于做对应分析的数据
ca_table<-Contingency.table[,-1]
rownames(ca_table)<-Contingency.table$Product

# 使用 CA 函数执行 CA 分析
#graph=FALSE 表示不返回图形
ca<-CA(ca_table,graph=FALSE)

# 构建特征值和惯性百分比列联表
#digit 参数表示结果保留小数点位数
digit=4
epi.table<-data.frame(round(t(ca$eig),digit))
# 输出结果
epi.table
```

	dim.1	dim.2	dim.3	dim.4	dim.5	dim.6	dim.7
eigenvalue	0.1105	0.0418	0.0343	0.0222	0.0166	0.0087	0.0069
percentage of variance	45.8479	17.3352	14.2483	9.2007	6.8986	3.6225	2.8468
cumulative percentage of variance	45.8479	63.1830	77.4313	86.6320	93.5307	97.1532	100.0000

关于一个 CA 对象变量 ca 主要有以下几个属性:

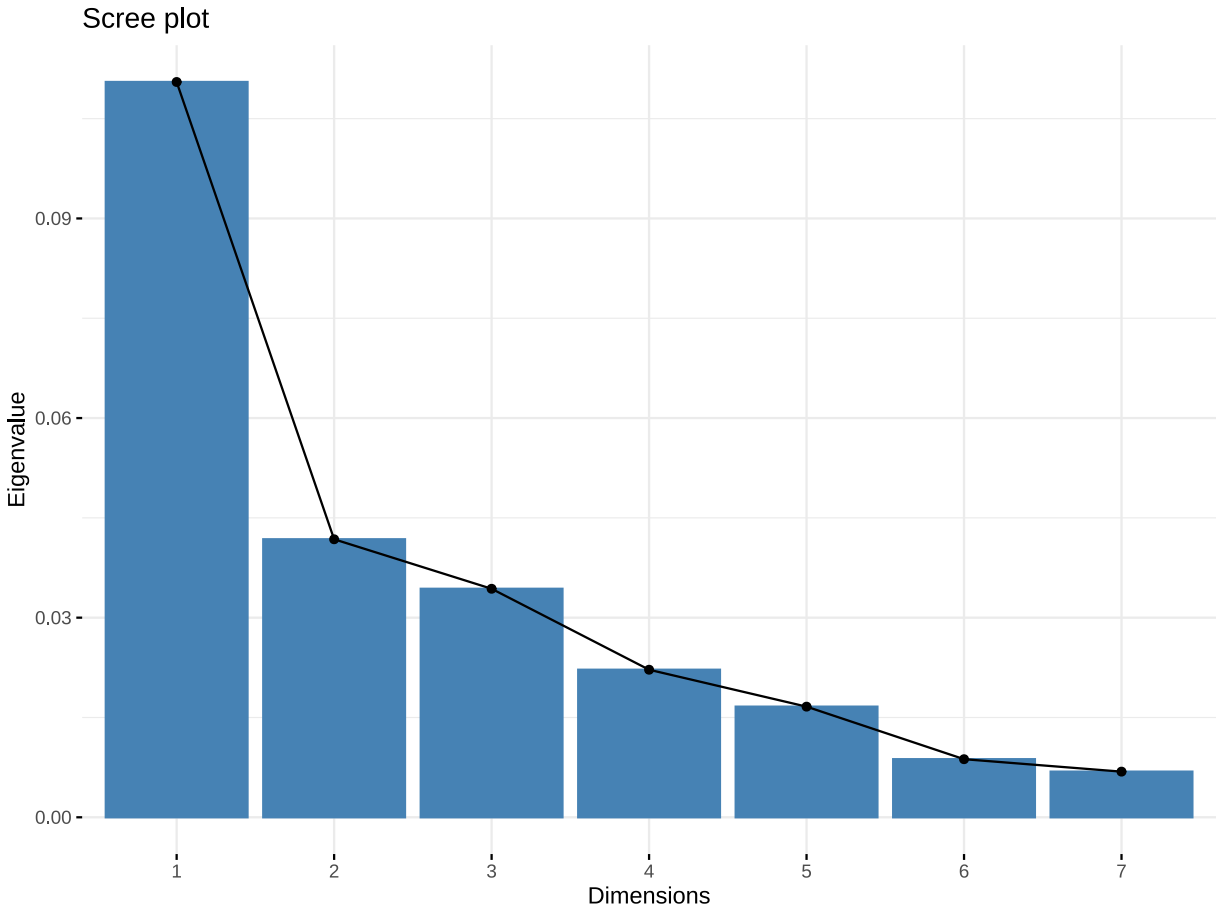
- `ca$eig`: 执行该代码，会获得对应分析的所有特征值数值以及惯性百分比组成的数据框
- `ca$row`: 执行该代码，会获得所有行变量 (产品) 的维度坐标，相关性等一些数据
- `ca$col`: 执行该代码，会获得所有列变量 (产品属性) 的维度坐标，相关性等一些数据

4.3 碎石图 (Scree plot)

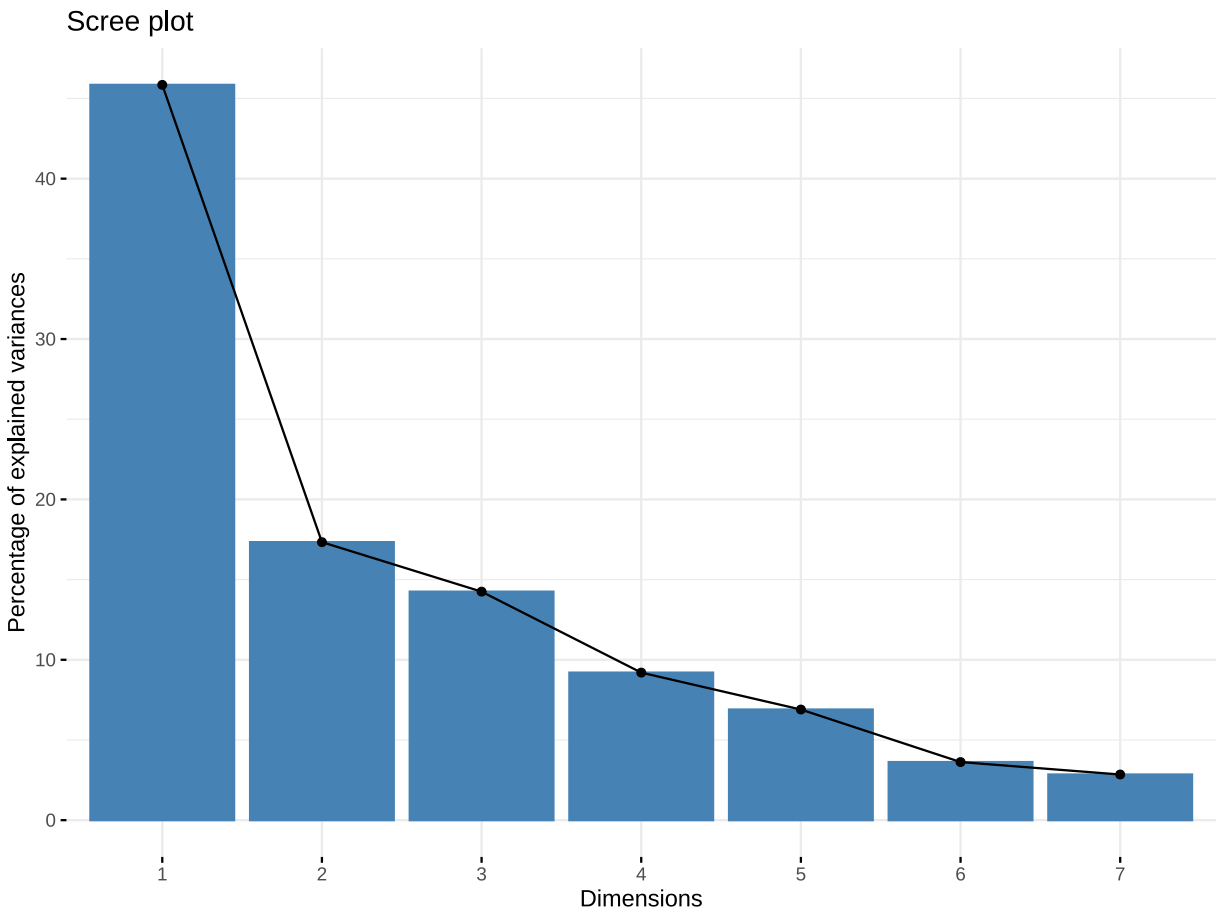
4.3.1 方法一

第一种方法只需调用 R 包 `factoextra` 包中的 `fviz_eig` 函数:

```
# 基于特征值的碎石图
fviz_eig(ca,choice="eigenvalue")
```




```
# 基于惯性百分比的碎石图
fviz_eig(ca,choice="variance")
```



函数说明： `fviz_eig` 函数主要有以下参数：

- **X**: 该参数为一个 CA、PCA 或 MCA 等对象，在上述对应分析中即 CA 函数的返回结果
- **choice**: 该参数有两个值可供选择：
  - `eigenvalue`: 表示画关于各维度特征值的碎石图，如上图所示；
  - `variance`: 表示画关于各维度方差贡献率的碎石图，即惯性百分比
- **geom**: 该参数有三种可选值：
  - `bar`: 表示只画条形图；
  - `line`: 表示只画折线图；
  - `c("bar","line")`: 表示两者都画 (默认两者都画)

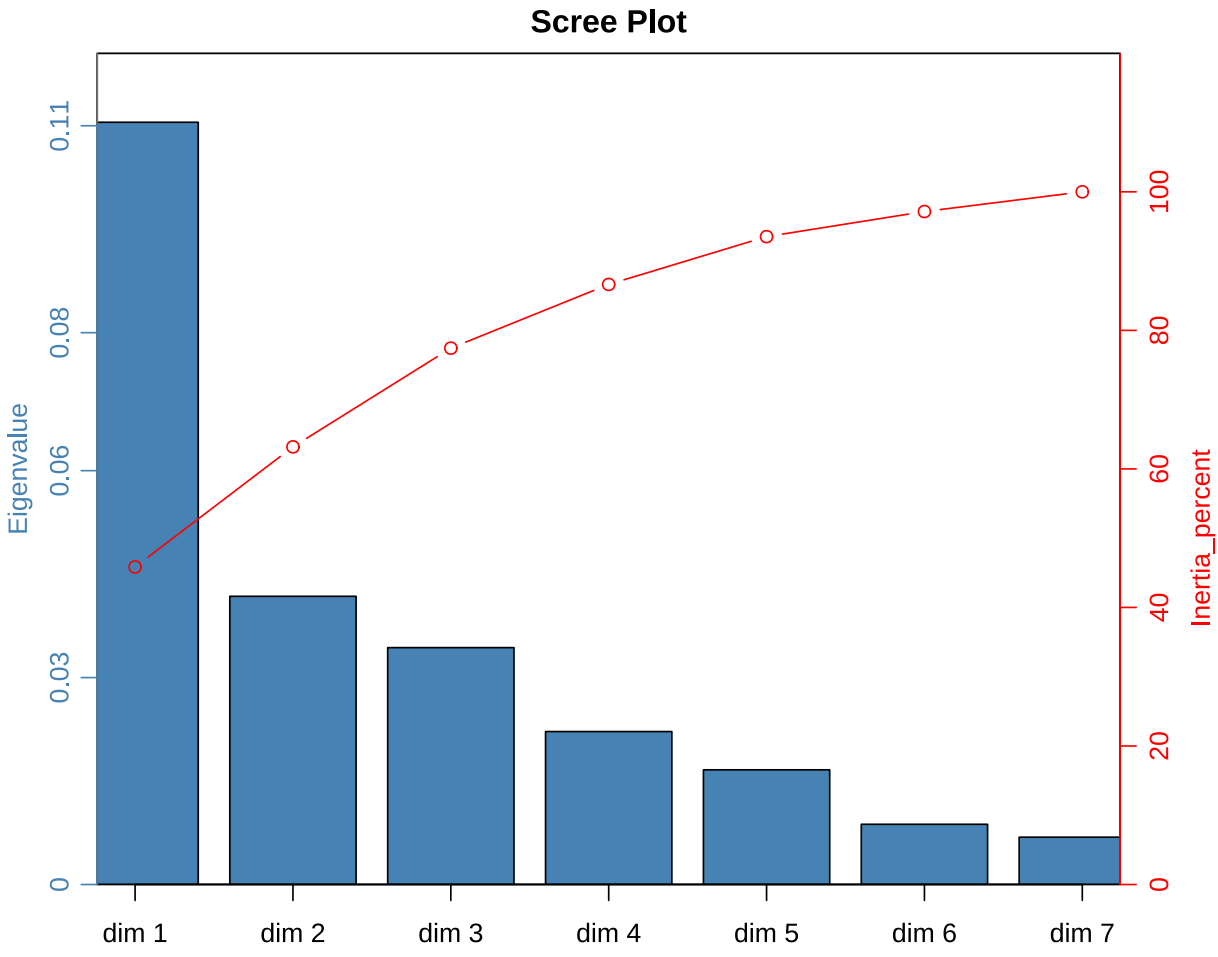
关于其他更多参数可以通过运行 `?fviz_eig`(在函数名面前加一个问号)，查看函数帮助。

4.3.2 方法二

可以发现，上述函数所绘制的碎石图只含有一个维度的信息：特征值或者惯性百分比，而无法同时包含两者信息，与 XLSATA 插件绘图有区别，所以我们利用 R 包 `plotrix` 里的 `twoord.plot` 函数自定义绘制，如果未安装该 R 包，按前面的代码自行安装。代码如下：

```
# 加载 R 包
library(plotrix)
# 绘图
# 该图可根据 CA 分析的 ca 变量直接运行得出，不需要修改其他参数
```

```r
# 当然前提是 ca 变量存在
twoord.plot(lx=1:nrow(ca$eig),ly=ca$eig[,1],
            rx=1:nrow(ca$eig),ry=ca$eig[,3],
            type=c("bar","b"),xlab="axis",ylab="Eigenvalue",
            rylab="Inertia_percent",rylim=c(0,120),
            lylim=c(0,max(Eigenvalue=ca$eig[,1])+0.01),
            ylab.at=(max(Eigenvalue=ca$eig[,1])+0.01)/2,
            rylab.at=50,lcol="steelblue",rcol="red",main="Scree Plot",
            lytickpos=round(seq(0,max(Eigenvalue=ca$eig[,1]),length.out=5),2),
            rytickpos=seq(0,100,by=20),rpch=1,
            xticklab=rownames(ca$eig),
            mar=c(2,4,2,4),    # 调整边距
            do.first="plot_bg(col='white')")
```

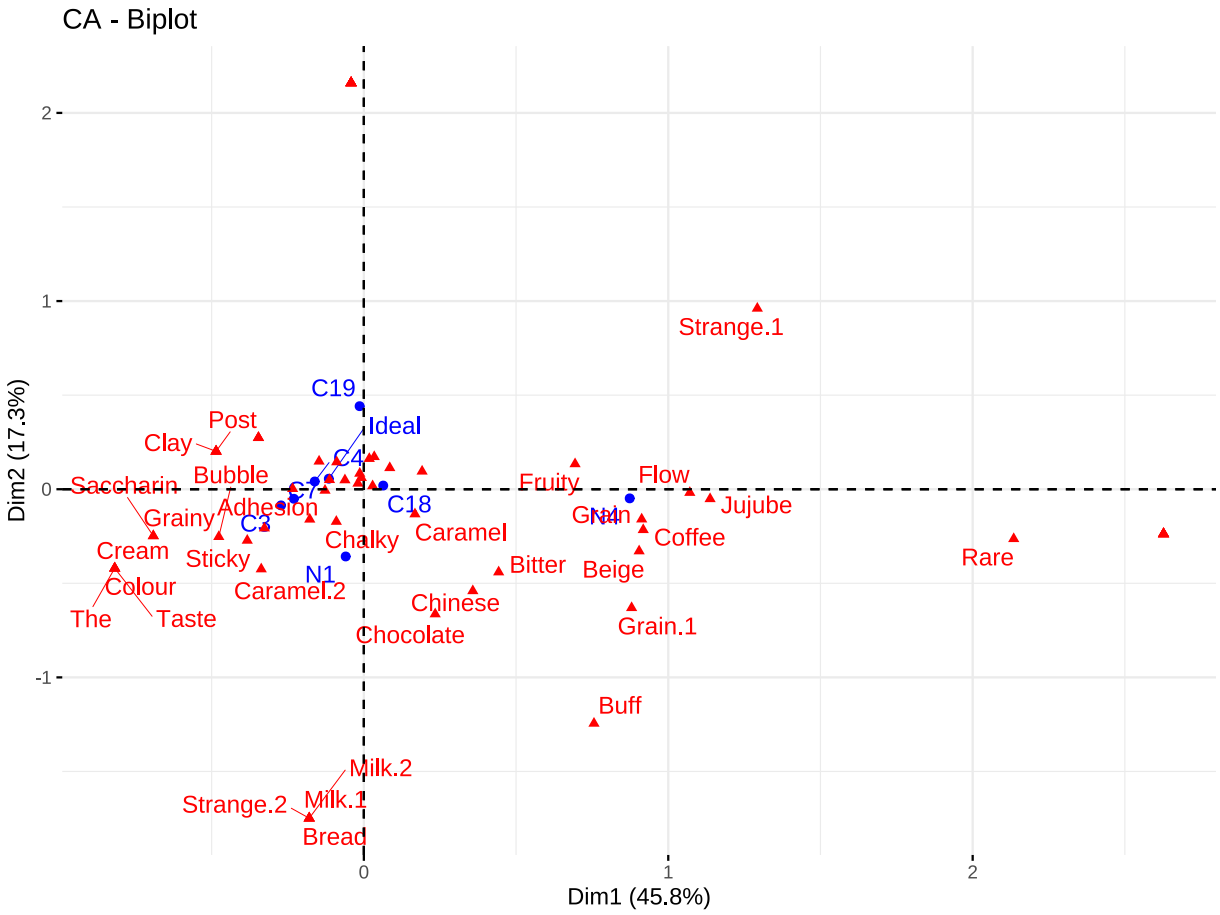


4.4 对称图 (Symmetric plot)

4.4.1 方法一

第一种方法只需一行代码，我们使用的是 `fviz_ca_biplot` 函数。

```r
# 加载需要的 R 包
library(FactoMineR)
library(factoextra)

#repel=TRUE 表示避免文本将散点覆盖
fviz_ca_biplot(ca,repel=TRUE)
```



函数说明： `fviz_ca_biplot` 主要有以下参数：

- **X**: 同理，该参数是一个 CA 对象，即 CA 函数的返回值 ca；
- **axes**: 该参数指定画图的两个维度，是一个长度为 2 的向量，默认值为 `c(1,2)`，指的是第一个维度和第二个维度；
- **geom**: 该参数的功能是指定图中显示的几何对象。有以下基本三种几何对象：
  - **point**：表示在图中画散点；
  - **arrow**：表示在图中画从 (0,0) 到每个点坐标的箭头；
  - **text**：表示在图中标注每个点的文本标签，即属性名和产品名；
  - 默认值是 `c("point","text")` 表示在图中显示散点和散点文本标签。上述三种几何对象可任意组合，所以该参数的值有 $2^3 - 1 = 7$ 种可能。
- **geom.row**: 该参数与 `geom` 参数类似，唯一不同在于该参数只指定行变量 (产品) 的几何对象显示；
- **geom.col**: 与 `geom.row` 类似，指定列变量 (属性) 的几何对象显示
- **repel**: 该参数为布尔值参数，只有 `TRUE` 和 `FALSE` 两种选择。`TRUE` 表示避免文本标签将散点覆盖 (当图

中散点太多，标签太多时可以使用)；FALSE 为默认参数，表示不去避免覆盖；

由于篇幅问题，以上是主要参数。当然还有很多有趣的参数，可以通过?fviz_ca_biplot 去查看该函数所有参数。

4.4.2 方法二

第二种方法效果整体与第一种方法相似，但是在第一种方法中，如果产品数与属性数过多，会导致文本标签要么太多看不清，要么太少最多只有 10 个文本标签。所以在第二种方法中，我自定义函数，当散点标签互相重叠覆盖时，设置了排除重叠太多内容的文本标签参数。

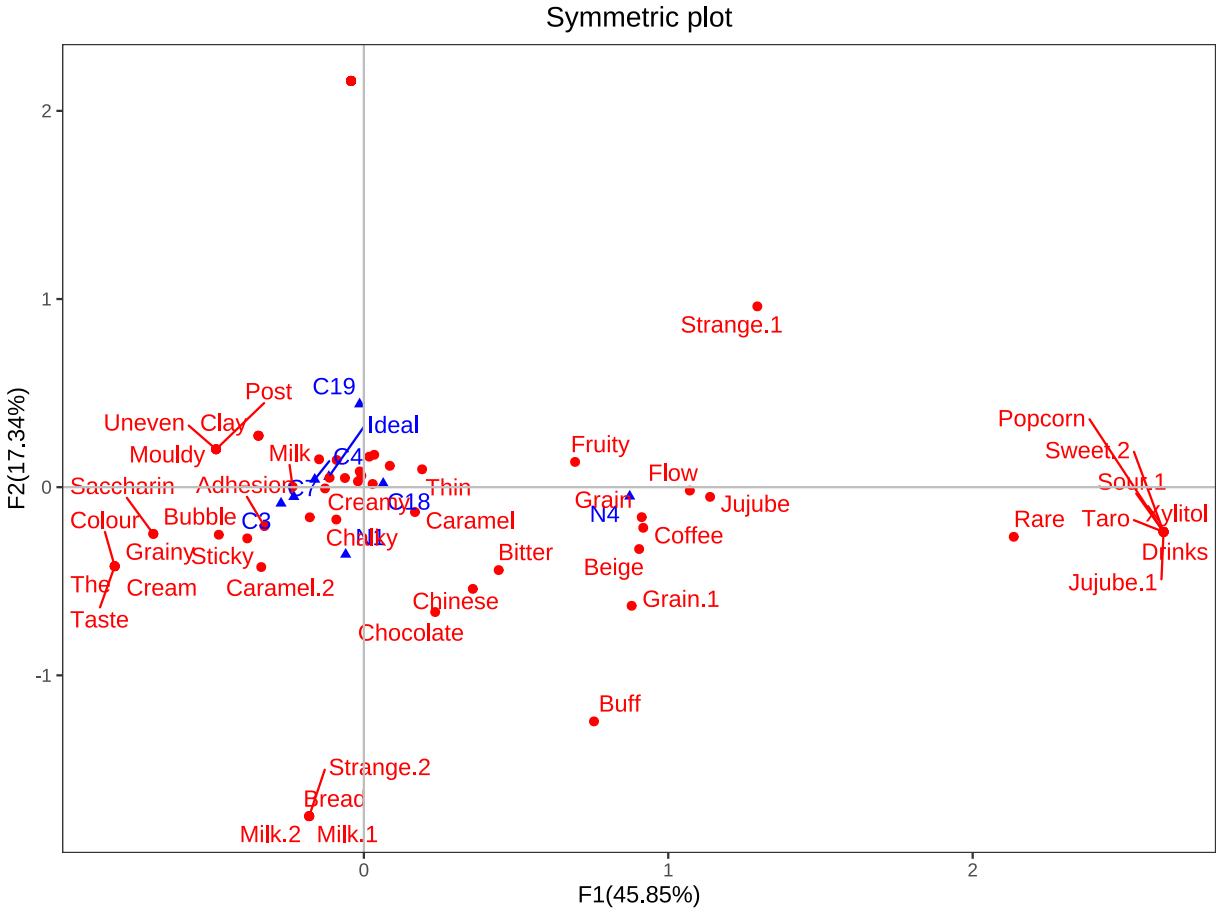
```
# 注意：此函数需要用到两个 R 包 ggplot2 与 ggrepel
# 如果未安装这两个包，请按前面方法安装
# 未安装的话，此函数将报错

Ca_cata_biplot<-function(X,dim=c(1,2),max.overlaps=10,
                         color=c("blue","red")){
  # 取出需要绘图的两个维度的数据
  row_dim<-data.frame(X$row$coord[,dim])
  col_dim<-data.frame(X$col$coord[,dim])
  Dim<-rbind(row_dim,col_dim)
  text<-c(rownames(row_dim),rownames(col_dim))
  class<-c(rep("1",nrow(row_dim)),rep("2",nrow(col_dim)))
  dim_data<-data.frame(Dim,text=text,class=class)
  # 绘图
  library(ggplot2)
  library(ggrepel)
  ggplot(data=dim_data,aes(x=dim_data[,1],y=dim_data[,2]))+
    geom_point(aes(shape=class,color=class))+
    geom_text_repel(data=row_dim,aes(x=row_dim[,1],y=row_dim[,2],
                                     label=rownames(row_dim)),color=color[1],
                    max.overlaps=max.overlaps)+
    geom_text_repel(data=col_dim,aes(x=col_dim[,1],y=col_dim[,2],
                                     label=rownames(col_dim)),max.overlaps=max.overlaps,
                    color=color[2])+
    geom_hline(yintercept=0,color="grey")+
    geom_vline(xintercept=0,color="grey")+
    scale_shape_manual(values=c(17,19))+
    scale_colour_manual(values=c("blue","red"))+
    guides(color="none",shape="none")+
    labs(title="Symmetric plot",
         x=paste("F",dim[1],"(",round(X$eig[dim[1],2],2),"%)",sep=""),
         y=paste("F",dim[2],"(",round(X$eig[dim[2],2],2),"%)",sep=""))+
    theme_bw()+
    theme(panel.grid=element_blank(),
          plot.title=element_text(hjust=0.5))
}
```

此函数有四个参数：

- **X**: 同理，该参数是一个 CA 对象，即 CA 函数的返回值 ca；
- **dim**: 该参数指定画图的两个维度，是一个长度为 2 的向量，默认值为 `c(1,2)`，指的是第一个维度和第二个维度；
- **max.overlaps**: 排除重叠太多内容的文本标签，显示在最大允许标签重叠数以下的文本标签 (相比方法一的改进)，默认为 10，即图中呈现文本标签重叠在 10 个以下的所有产品或者属性。参数值越大，允许显示的文本标签越多。
- **color**: 指定产品与属性的颜色，默认值为 `c("blue","red")`。

同理，我将其打包成一个 R 文件，调用方法以及注意事项前面已有示例。下面代码展示效果：

```
# 运行 Ca_cata_biplot 函数 R 文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Ca_cata_biplot.R")

# 效果展示
# 带有默认值的参数可以不必指定
# 例如不想修改默认值的话,dim,max.overlaps 与 color 参数可不必写出
# 下面代码等价于  Ca_cata_biplot(ca)
Ca_cata_biplot(ca,dim=c(1,2),max.overlaps=15,color=c("blue","red"))
```



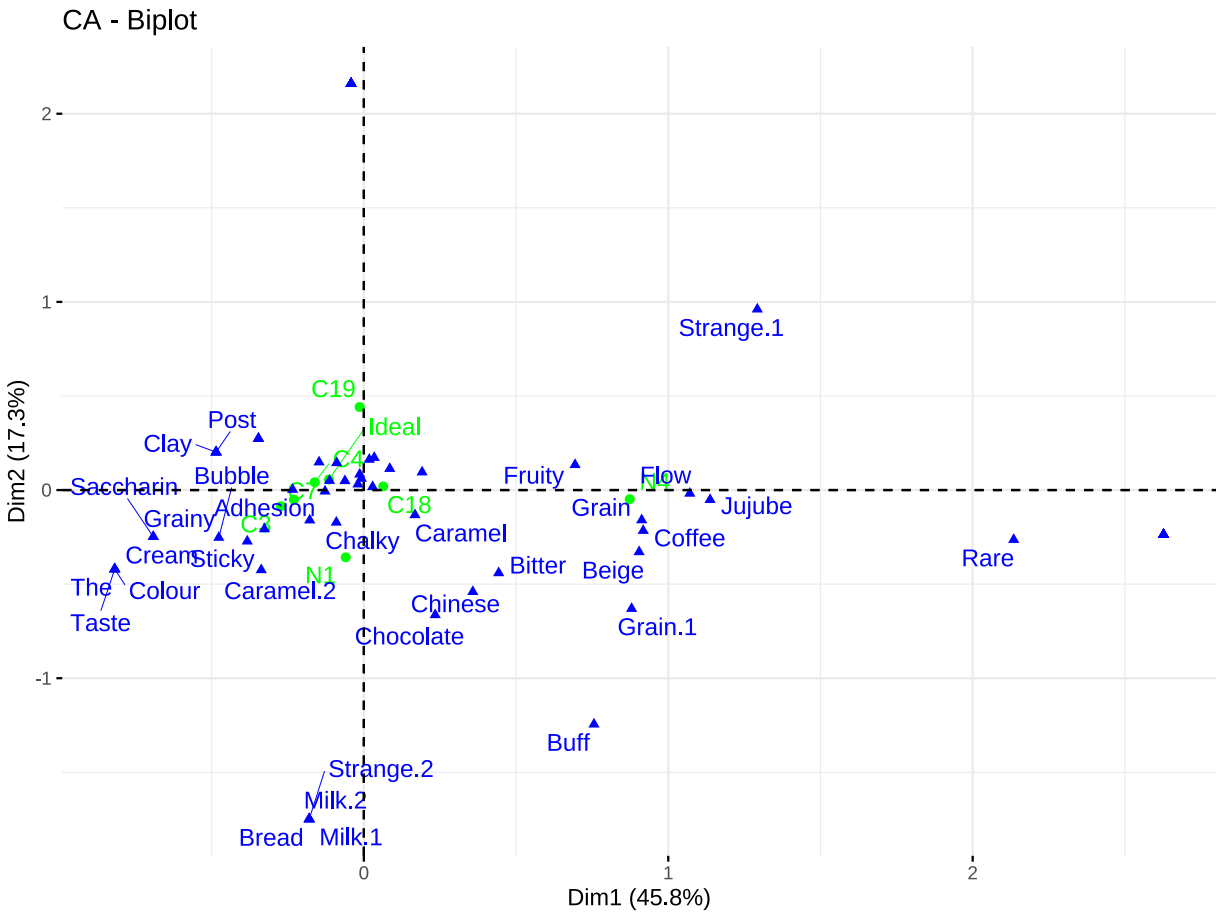
4.5 R 包 factoextra 简单介绍

R 包 `factoextra` 是一个专门对各种降维统计方法如对应分析 (CA), 主成分分析 (PCA), 多重因子分析 (MFA) 等进行可视化的一个很强大的包。下面我以对应分析 (CA) 为例，将其中的几个主要的可视化函数进行简单介绍。

4.5.1 fviz_ca_biplot()

前面已经介绍过了这个函数的主要参数。这个函数是专门对 CA 分析的变量 (属性) 与观测值 (产品) 进行两个维度的可视化。下面进行简单示例：

```
fviz_ca_biplot(ca,                          #CA 对象
    geom=c("point","text"),      # 显示散点和文本标签
    repel=TRUE,                   # 避免文本重叠
    col.col="blue",               # 指定列变量颜色，也可以指定分类变量进行映射
    col.row="green")              # 指定行变量颜色，也可以指定分类变量进行映射
```

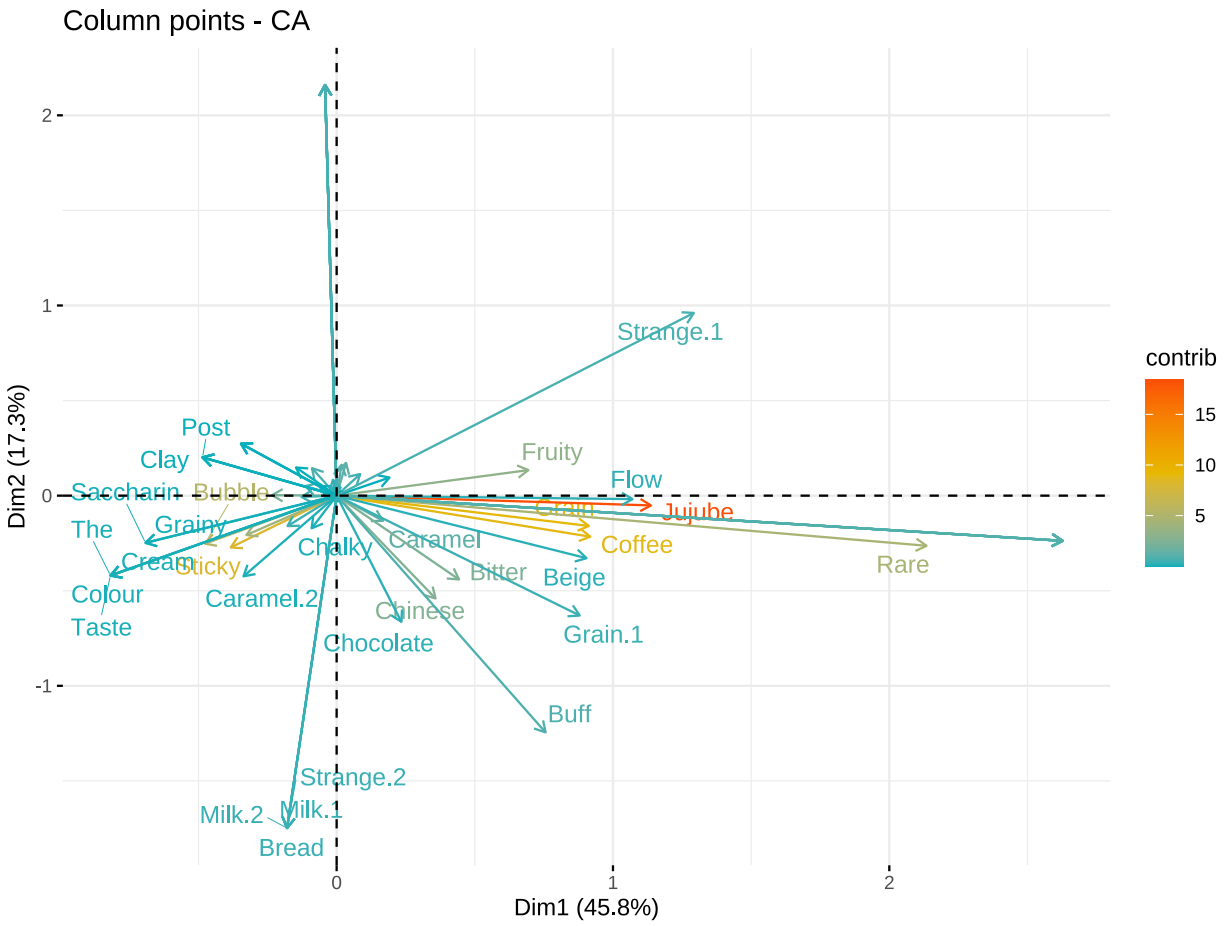



```
    arrows=c(TRUE,FALSE)        # 行变量显示箭头，列变量不显示
```

4.5.2 fviz_ca_col()

该函数是专门只对列变量 (属性) 进行二维可视化的函数, 其参数与 `fviz_ca_biplot` 基本相同。见如下代码示例：

```
fviz_ca_col(ca,
      # 依据每个列变量 (属性) 的贡献值大小填充颜色
      col.col = "contrib",
      # 只显示箭头和文本
      geom.col=c("arrow","text"),
      # 引号内部是十六进制颜色码，以 # 号开头，该参数表示按照这三种颜色渐进映射
      gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
      repel = TRUE
            )
```

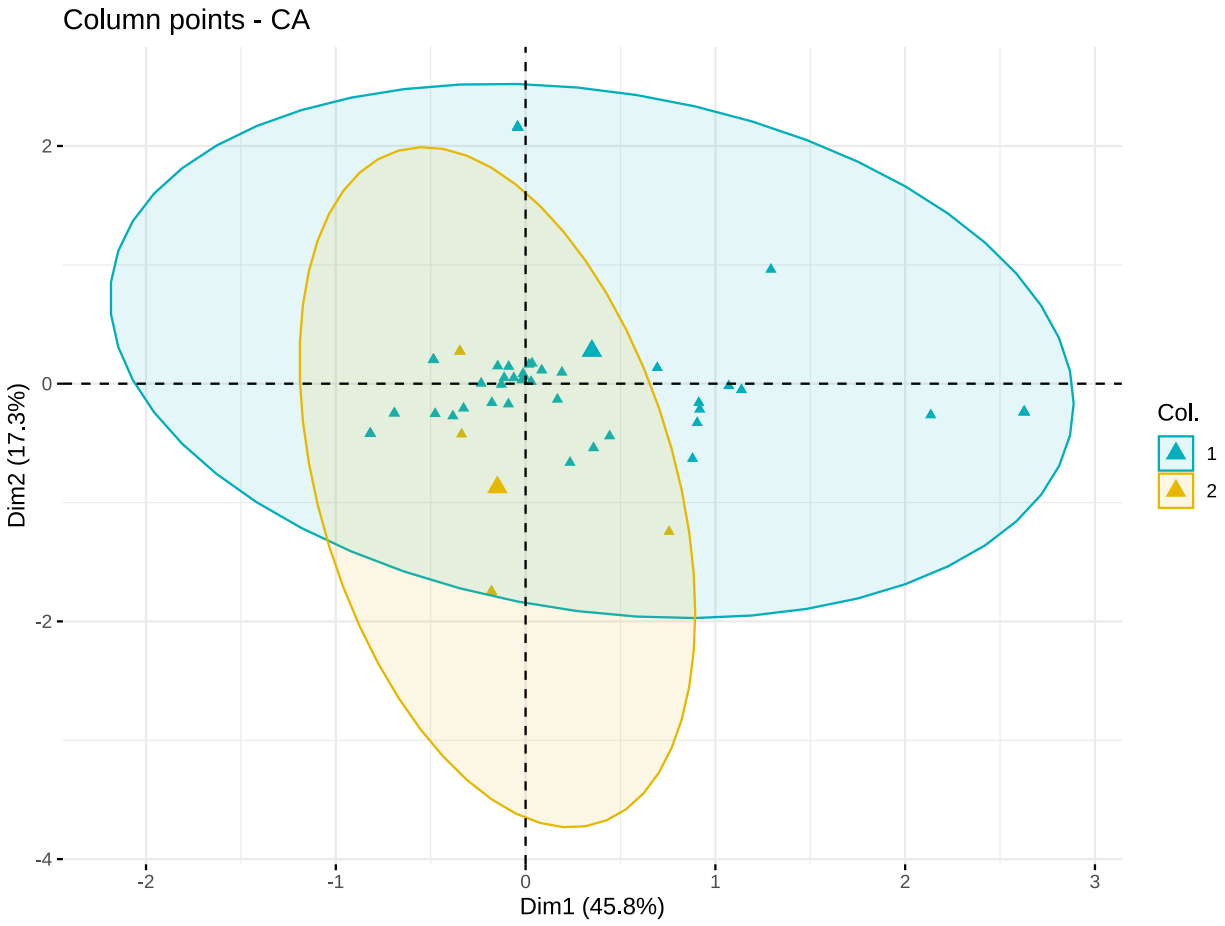



```
# 通过执行代码"?fviz_ca_col"，可查看全部参数。
```

由于文本重叠数太多 (属性数太多)，为了避免文本重叠，所以才没有显示文本。接下来，我们通过对列变量 (属性) 的二维坐标进行聚类，根据聚类结果进行可视化：

```
# 随机数种子
# 由于 kmeans 聚类利用了随机数
# 为了保持每次运行代码聚类结果相同，需要设定随机数种子
set.seed(1)

# 进行 kmean 聚类
```

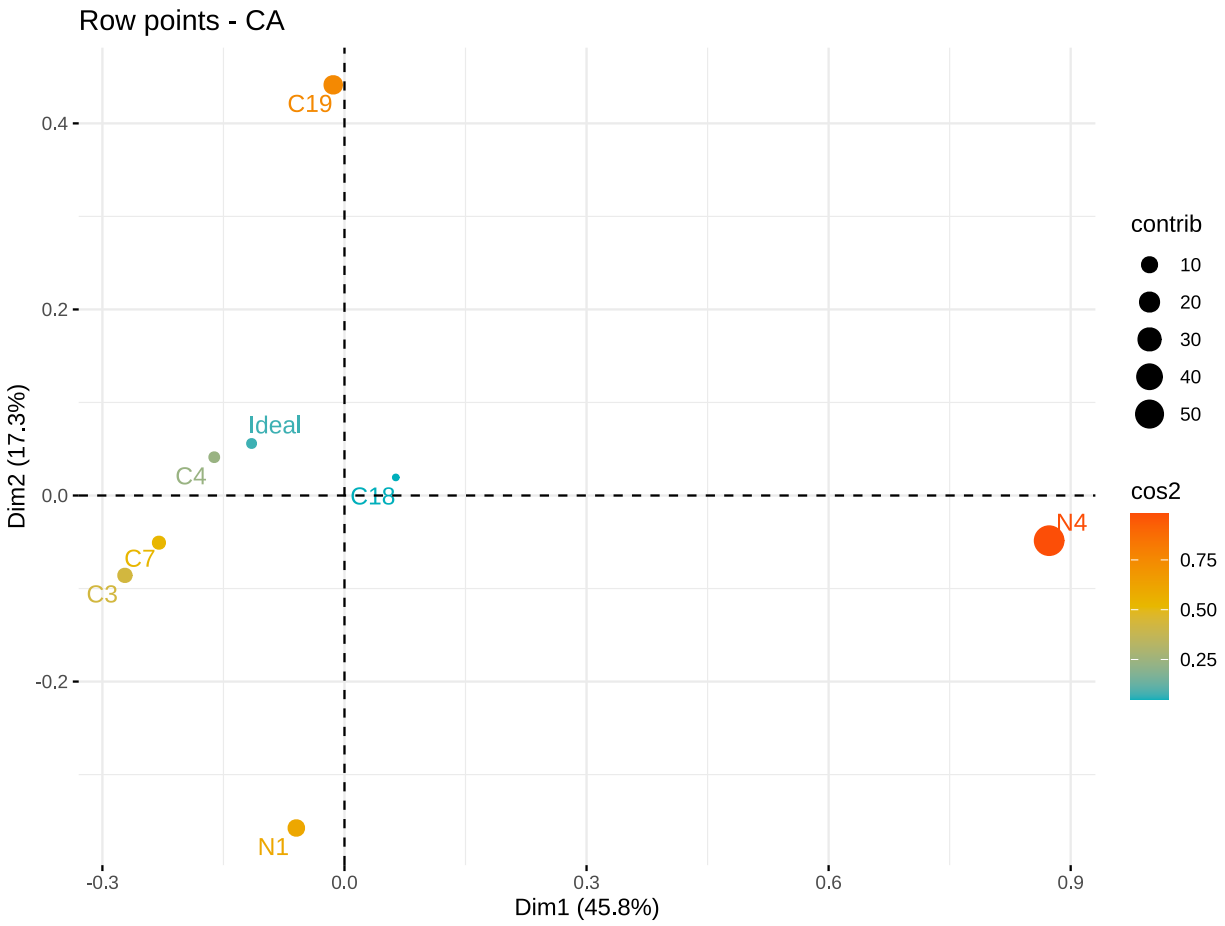
```r
#ca$col$coord 是属性的两个维度的坐标
#centers 是要聚成几类
c<-kmeans(ca$col$coord,centers=2)

# 可视化
fviz_ca_col(ca,
      # 依据每个列变量 (属性) 的聚类结果填充颜色
      col.col = as.factor(c$cluster),
      # 只显示散点
      geom.col="point",
      # 引号内部是十六进制颜色码，以 # 号开头，该参数表示按照这三种颜色渐进映射
      palette = c("#00AFBB", "#E7B800"),
      # 在每一类上添加椭圆
      addEllipses = TRUE
            )
```



4.5.3 fviz_ca_row()

该函数是专门只对行变量 (产品) 进行二维可视化的函数, 其参数与 `fviz_ca_biplot` 基本相同。见如下代码示例:

```r
fviz_ca_row(ca,
            # 按照各行变量（产品）的 cos2 值大小来映射颜色
            col.row = "cos2",
            # 按照各行变量（产品）的 contrib 值大小来映射散点大小
            pointsize = "contrib",
            gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
            repel = TRUE
             )
```



5 主坐标分析 (Principal Coordinate Analysis)

注意：此方法分析数据对是否含有理想点数据无要求。下面以不含有理想点数据为例，并且必须包含 Liking 得分数据。

5.1 使用数据说明

由于老师所给数据属性太多，且多数属性基本都是 0，不太适合做主坐标分析，所以我选择使用 XLSTAT 的关于 CATA 分析的 demo 数据去做主坐标分析。执行代码如下：

```
# 导入数据
data<-read.csv("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\demo.csv")

# 去除理想点数据
data<-data[data$Product!="Ideal",]
```

5.2 R 包的准备

我们需要用到一个内置主坐标分析函数的 R 包 ape, 使用其内置 pcoa 函数进行主坐标分析。第一次运行请按前面方法安装这个 R 包。

5.3 相关系数表 (Correlations)

我们需要求属性与属性之间的四分相关系数 (tetrachoric correlation)，属性与喜好得分 (Liking) 之间的点二列相关系数。

5.3.1 四分相关系数 (tetrachoric correlation)

四分相关系数是针对于两个二分变量计算它们之间的相关程度，其计算公式如下：

$$r = \cos\left[\frac{\sqrt{bc}}{\sqrt{ad}+\sqrt{bc}}\pi\right],$$

其中 a,b,c,d 见下表，π 为圆周率：

	0	1	合计
0	a	b	a+b
1	c	d	c+d
合计	a+c	b+d	n

表 5: 四格表

所以我自定义函数去计算四分相关系数，代码如下。这个函数是一个中间函数，主要是为了后面的函数调用，可以不用管。

```
## 对各属性两两计算四分相关系数 (tetrachoric correlation)
# 定义计算四分相关系数公式
TetrCor<-function(x){#x 为两列属性变量的数据框
  n11<-sum(x[1]==1 & x[2]==1)
  n12<-sum(x[1]==1 & x[2]==0)
  n21<-sum(x[1]==0 & x[2]==1)
  n22<-sum(x[1]==0 & x[2]==0)
  tab<-matrix(c(n11,n21,n12,n22),nrow=2)
```

```
  r<-cos(sqrt(tab[1,2]*tab[2,1])/(sqrt(tab[1,1]*tab[2,2])+
                                    sqrt(tab[1,2]*tab[2,1]))*pi)
  return(r)
}
```

5.3.2 点二列相关系数 (tetrachoric correlation)

点二列相关系数是用于计算连续变量与二分变量的相关程度，其计算公式如下：

$$R = \frac{\overline{X}_p - \overline{X}_q}{\sigma}\sqrt{pq}$$

其中

- p 表示二分变量中某一类别频率，q 表示另一类别频率；
- $\overline{X}_p$ 表示与二分变量 p 类别对应的连续变量的均值，$\overline{X}_q$ 同理；
- σ 是连续变量 X 的标准差。

自定义计算点二列相关系数函数代码如下，同样这是一个中间函数，可以不用理会。

```
# 定义计算点二列相关系数函数
#x 为两列变量的数据框，第一列为 Liking 得分，第二列为二分属性变量
BiseCor<-function(x){
  p<-sum(x[2])/nrow(x)
  q<-1-p
  x_p<-mean(x[x[2]==1,1])
  x_q<-mean(x[x[2]==0,1])
  std<-sqrt(var(x[1]))
  r<-(x_p-x_q)/std*sqrt(p*q)
  return(r[1,1])    #r[1,1] 是为了不返回行名与列名
}
```

5.3.3 相关系数全表

自定义 Corr_table 函数，并调用上述定义的计算相关系数函数，最后得到等同于 XLSTAT 插件上的的属性与喜好得分的相关系数全表。函数代码如下：

```
# 建立相关系数表 (数据不包括理想点)
#index:Liking 得分索引
Corr_table<-function(data,index){
  attr_all<- colnames(data)[(index+1):ncol(data)]# 保存所有属性名
  # 所有属性个数 +1 是因为还有 Liking 得分
  Corr_tab<-matrix(nrow=length(attr_all)+1,ncol=length(attr_all)+1)
  for(i in 1:(length(attr_all)+1)){
    for(j in i:(length(attr_all)+1)){
      if(i==j){Corr_tab[i,j]<-1}
      else if(j==length(attr_all)+1){
        Corr_tab[i,j]<-round(BiseCor(data[,c(index,i+index)]),3)
        Corr_tab[j,i]<-round(Corr_tab[i,j],3)  # 由于表格是对称的
```

```
      }
      else{
        Corr_tab[i,j]<-round(TetrCor(data[,c(i+index,j+index)]),3)
        Corr_tab[j,i]<-round(Corr_tab[i,j],3)  # 由于表格是对称的
      }
    }
  }
  # 从矩阵格式转化为数据框格式
  Corr_tab<-data.frame(Corr_tab)
  Corr_tab<-cbind(a=c(attr_all,"Liking"),Corr_tab)
  colnames(Corr_tab)<-c(" ",attr_all,"Liking")
  Corr_tab[is.na(Corr_tab)]<-""

  return(Corr_tab)
}
```

函数说明：该函数只有两个可修改参数：

- **data**: 所分析的数据；
- **index**:Liking 得分的列索引, 注意前面方法中的 `index` 参数是分析数据中从左到右第一个属性的列索引，与前面有所不同。

同样，为了方便直接调用，我将上述函数打包为一个 R 文件, 直接用 `source` 函数调用即可。下面是结果示例：

```
# 运行上述的自定义函数
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Corr_table.R")

#index 参数根据不同 CATA 数据的 Liking 得分列索引不同而进行修改
Corr_tab<-Corr_table(data=data,index=3)

# 结果展示
# 由于数据中属性列太多，我们只展示前 8 行与前 8 列
Corr_tab[1:8,1:8]
```

	Firm	Juicy	Sweet	Bitter	Sour	Crispy	Tasty
Firm	1.000	0.471	-0.005	-0.156	0.431	0.607	0.337
Juicy	0.471	1.000	0.326	-0.437	-0.152	0.395	0.490
Sweet	-0.005	0.326	1.000	-0.641	-0.615	0.011	0.445
Bitter	-0.156	-0.437	-0.641	1.000	0.228	-0.108	-0.640
Sour	0.431	-0.152	-0.615	0.228	1.000	0.280	-0.168
Crispy	0.607	0.395	0.011	-0.108	0.280	1.000	0.335
Tasty	0.337	0.490	0.445	-0.640	-0.168	0.335	1.000
Grainy	-0.693	-0.519	-0.074	-0.187	-0.690	-0.833	-0.587

可以发现，上述结果与 XLSTAT 插件结果不是完全一致，这是因为上述相关系数的计算用到了圆周率 π，而不同软件程序，对用来参与计算的 π 的小数保留位数可能不一样。总体上，只有小数点后几位的差别，基本一致。

5.4 主坐标分析可视化

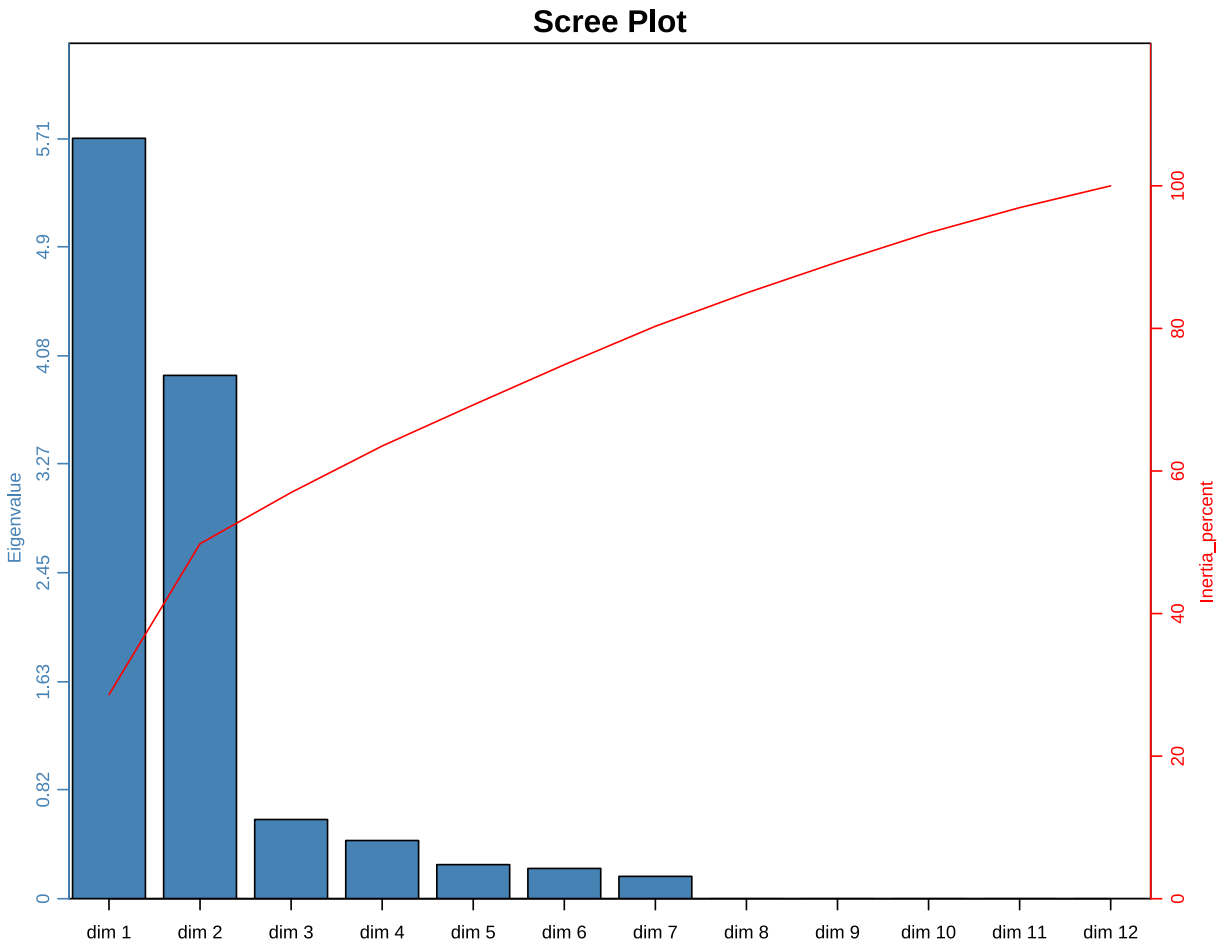
我们不妨用前面的相关系数表来定义属性与喜好得分之间的两两距离，定义距离矩阵 $D = 1 - Corr\_tab$，然后执行下面代码得到 pcoa 对象。

```
# 加载 R 包，调用主坐标分析函数 pcoa
library(ape)

# 构造距离矩阵 D
Corr_tab<-Corr_tab[,-1]
Corr_tab<-as.data.frame(lapply(Corr_tab,as.numeric))
D=1-Corr_tab

#pcoa 函数的参数对象是一个距离矩阵
PCOA<-pcoa(D)
```

5.4.1 碎石图 (Scree plot)

与对应分析类似，只需要 pcoa 对象即可实现碎石图，下面代码只需针对不同的 pcoa 对象，替换变量 PCOA 即可。

```
# 前期处理
row_num<-sum(abs(PCOA$values[,5]-1)>1e-3)
eig<-PCOA$values[1:(row_num+1),]

# 绘图
twoord.plot(lx=1:nrow(eig),ly=eig[,1],
            rx=1:nrow(eig),ry=eig[,5]*100,
            type=c("bar","l"),xlab="axis",ylab="Eigenvalue",
            rylab="Inertia_percent",rylim=c(0,120),
            lylim=c(0,max(Eigenvalue=eig[,1])+max(Eigenvalue=eig[,1])/8),
            ylab.at=(max(Eigenvalue=eig[,1])+0.01)/2,
            rylab.at=50,lcol="steelblue",rcol="red",main="Scree Plot",
            lytickpos=round(seq(0,max(Eigenvalue=eig[,1]),length.out=8),2),
            rytickpos=seq(0,100,by=20),rpch=1,
            xticklab=paste("dim",rownames(eig)),
            mar=c(2,4,2,4),    # 调整边距
            axislab.cex=0.7,
            do.first="plot_bg(col='white')")
```



关于所有特征值大小情况，可以通过代码 `PCOA$values` 获得。

5.4.2 对称图 (Symmetric plot)

与对应分析类似，根据维度坐标可以画出对称图。代码如下：

```
# 注意：此函数需要用到两个 R 包 ggplot2 与 ggrepel
# 如果未安装这两个包，请按前面方法安装
# 未安装的话，此函数将报错

Pcoa_cata_biplot<-function(X,dim=c(1,2),corr_tab,max.overlaps=10,color=c("blue","red")){
  # 取出需要绘图的两个维度的数据
  Dim<-data.frame(X$vectors[,dim])
  text<-colnames(corr_tab)
  class<-c(rep("1",nrow(Dim)-1),"2")
  dim_data<-data.frame(Dim,text=text,class=class)
  # 绘图
  library(ggplot2)
  library(ggrepel)
  ggplot(data=dim_data,aes(x=dim_data[,1],y=dim_data[,2]))+
    geom_point(aes(shape=class,color=class))+
    geom_text_repel(aes(label=text,color=class),
```

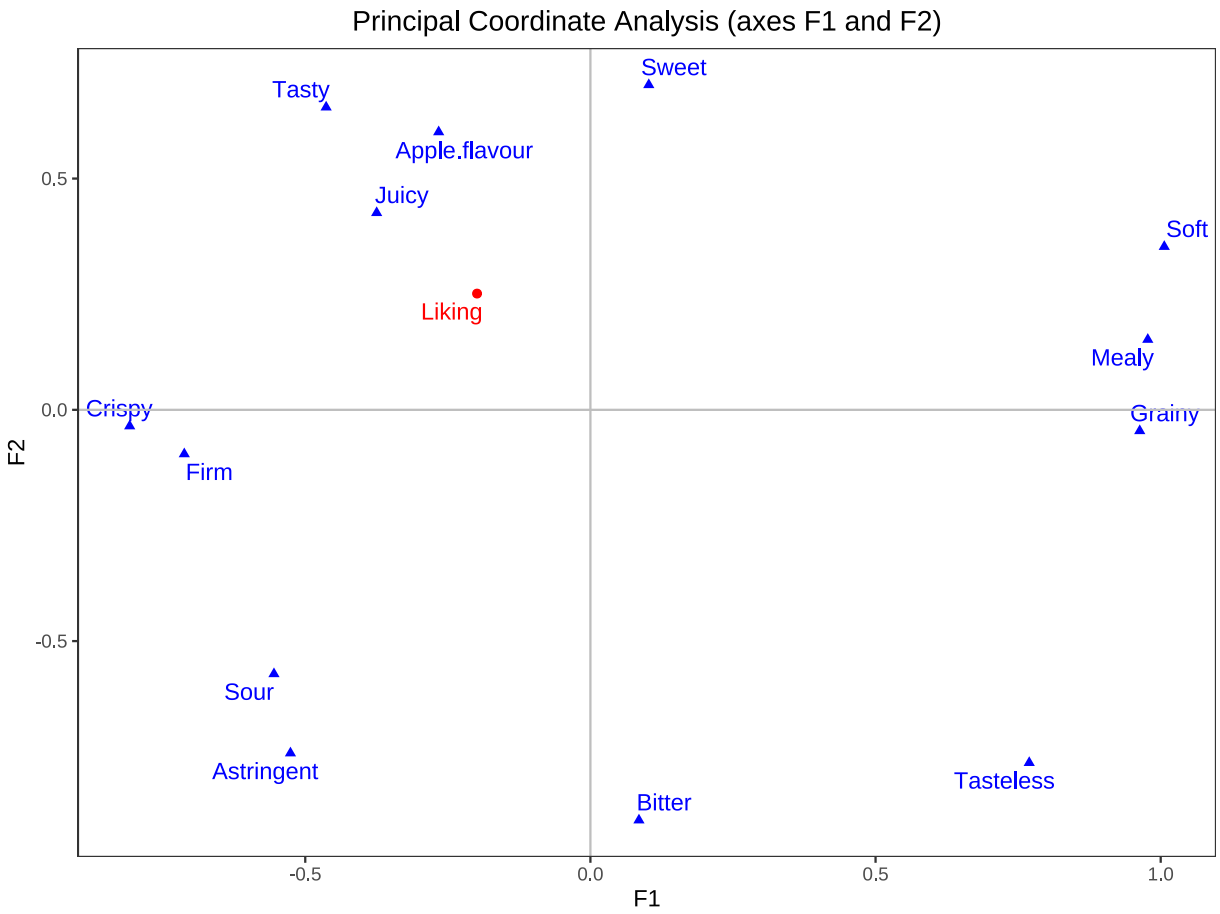
```
                    max.overlaps=max.overlaps)+
    geom_hline(yintercept=0,color="grey")+
    geom_vline(xintercept=0,color="grey")+
    scale_shape_manual(values=c(17,19))+
    scale_colour_manual(values=color)+
    guides(color="none",shape="none")+
    labs(title=paste("Principal Coordinate Analysis (axes F",dim[1],
                     " and F",dim[2],")",sep=""),
         x=paste("F",dim[1],sep=""),
         y=paste("F",dim[2],sep=""))+
    theme_bw()+
    theme(panel.grid=element_blank(),
          plot.title=element_text(hjust=0.5))
}
```

此函数有五个参数：

- **X**: 同理，该参数是一个 PCOA 对象，即 pcoa 函数的返回值 PCOA；
- **dim**: 该参数指定画图的两个维度，是一个长度为 2 的向量，默认值为 `c(1,2)`，指的是第一个维度和第二个维度；
- **corr_tab**: 该参数为主坐标分析的相关系数表；
- **max.overlaps**: 排除重叠太多内容的文本标签，显示在最大允许标签重叠数以下的文本标签 (相比方法一的改进)，默认为 10，即图中呈现文本标签重叠在 10 个以下的所有产品或者属性。参数值越大，允许显示的文本标签越多。
- **color**: 指定产品与属性的颜色，默认值为 `c("blue","red")`。

同样我也将其打包成一个 R 文件，效果示例如下。

```
# 运行函数源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Pcoa_cata_biplot.R")

# 效果展示
# 注意此函数的参数 X 对象是 pcoa 对象，即 pcoa 函数的返回值
Pcoa_cata_biplot(PCOA,corr_tab=Corr_tab)
```



6 惩戒分析 (Penalty analysis)

注意：此方法分析数据应包含 Liking 喜好得分数据，但是对于数据是否有理想点，输出图表有所不同，所以下面我们单独讨论。

6.1 R 包准备

惩戒分析绘图中会频繁用到 R 包 `ggplot2` 与 R 包 `ggrepel`，请执行代码之前，确保自己安装了这两个包，否则会报错。

6.2 有理想点数据

由于老师的数据属性太多，不易展示绘图效果，所以我继续沿用上节主坐标分析的 demo 数据, 但是上节数据无理想点数据，所以我们重新导入。

```
# 导入数据
data <- read.csv("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\demo.csv")
```

```r

# 由于 XLSTAT 示例中去掉了两个属性，为了保持一致，我们也去掉这两个属性
# 去掉第 8 列与第 14 列
data<-data[,-c(8,14)]
```

6.2.1 所有属性分析 (Analysis of attributes)

6.2.1.1 必读说明 需要说明的是，在 XLSTAT 分析的示例中，作者分别对 “the must have attributes” 以及 “the nice to have and negative attributes” 进行单独分析以及可视化。但是可以明显发现，两者的图表构成都是一致的，所以在以下小节中，每个图表都同时适用两种属性类别，只要修改 `ideal` 参数即可。

6.2.1.2 汇总表 (Summary table) 我定义了 `Summary_tab` 函数，该函数直接返回汇总表。函数源代码如下:

```r
# 定义求 Summary Table 函数
#data 为原始分析数据
#index 为数据中从左到右第一个属性的索引
#ideal 为 0 或 1

Summary_tab<-function(data,ideal=1,index){
  cosum<-unique(data$Consumer)
  N_Y_num<-0   #N_T_num 为 P(NO|Yes) 的数量
  Y_Y_num<-0   ##N_T_num 为 P(Yes|Yes) 的数量
  # 定义存储数据的空矩阵
  sum_table<-matrix(nrow=2,ncol=length(index:ncol(data)))
  for(i in index:ncol(data)){
    for(item in cosum){
      cosum_data<-data[data$Consumer==item,]
      # 计算 N_Y_num
      if(cosum_data[cosum_data$Product=="Ideal",][,i]==ideal){
        N_Y_num<-N_Y_num+sum(cosum_data[cosum_data$Product!="Ideal",][,i]==1-ideal)
      }
      # 计算 Y_Y_num
      if(cosum_data[cosum_data$Product=="Ideal",][,i]==ideal){
        Y_Y_num<-Y_Y_num+sum(cosum_data[cosum_data$Product!="Ideal",][,i]==ideal)
      }
    }
    sum_table[1,i-(index-1)]<-N_Y_num
    sum_table[2,i-(index-1)]<-Y_Y_num
    # 每次计算完一个属性之后，存储数据的变量值要清零
    N_Y_num<-0
    Y_Y_num<-0
  }
  # 得到数据矩阵之后，将其转化为数据框格式，并赋予行名与列名
  if(ideal==1){rowname=c("P(No)|(Yes)","P(Yes)|(Yes)")}
  else{rowname=c("P(Yes)|(No)","P(No)|(No)")}
  sum_df<-data.frame(sum_table)
  sum_df<-cbind(Level=rowname,sum_df)
```

```
  colnames(sum_df)<-c("Level",colnames(data)[index:ncol(data)])
  if(ideal==1){return(sum_df)}
  else{
    sum_df<-sum_df[c(2,1),]
    row.names(sum_df)<-c(1,2)
    return(sum_df)}
}
```

函数说明：该函数有三个参数：

- **data**: 分析数据
- **index**: 分析数据中从左到右第一个属性的列索引
- **ideal**: 该参数有两种取值：
  - **1**: 该参数为 1 时表示属于 “Analysis of the must have attributes”;
  - **0**: 该参数为 0 时表示属于 “Analysis of the nice to have and negative attributes”;

同理，将该函数打包成一个 R 文件，便于直接调用。下面是代码效果示例：

```
# 运行代码源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Summary_tab.R")

#the must have attributes
sum1_df<-Summary_tab(data=data,ideal=1,index=4)
# 由于属性过多，我们只展示前 10 个属性
sum1_df[,1:10]
```

Level	Firm	Juicy	Sweet	Bitter	Sour	Crispy	Tasty	Grainy	Soft
P(No)|(Yes)	225	226	290	6	79	202	281	13	24
P(Yes)|(Yes)	245	324	170	4	51	178	169	2	11

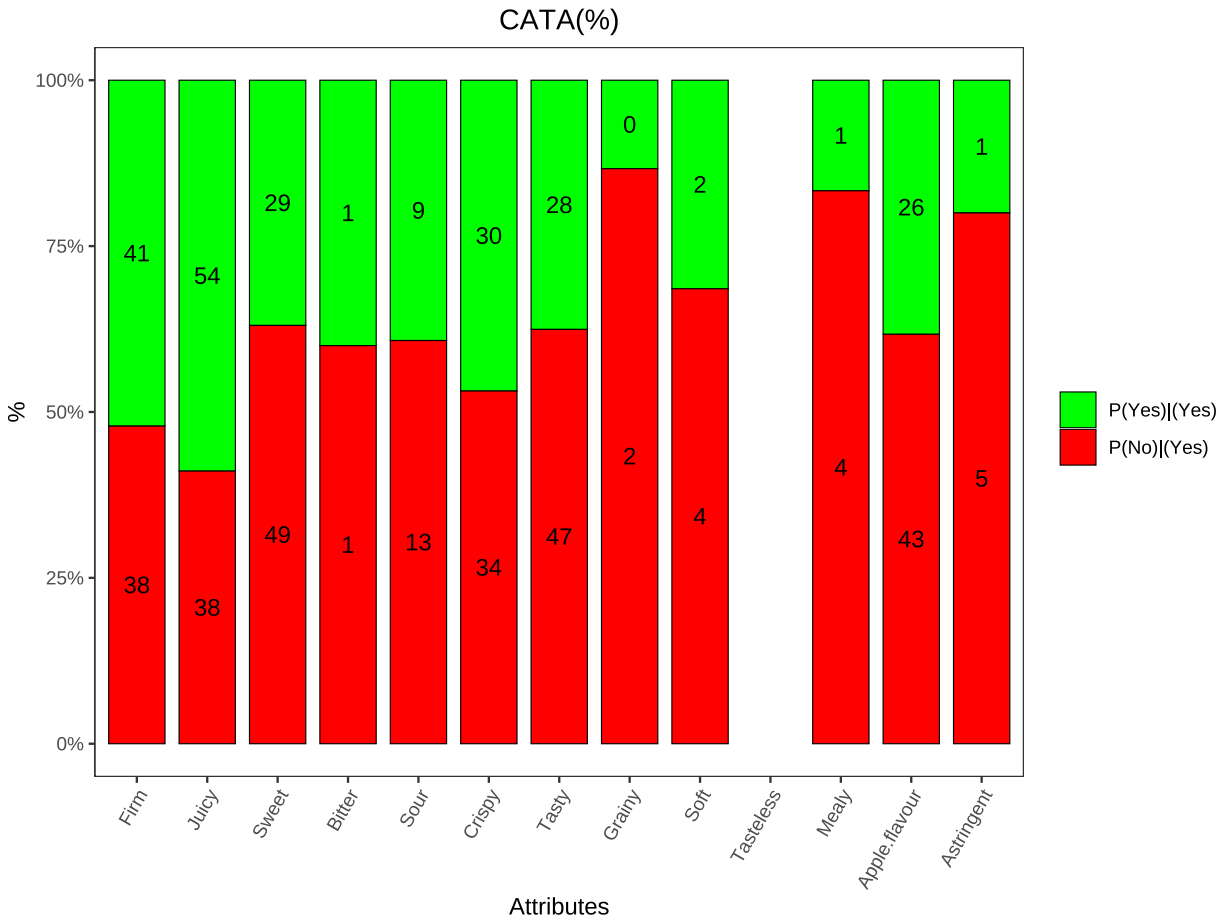
```
#the nice to have and negative attributes
sum0_df<-Summary_tab(data=data,ideal=0,index=4)
# 由于属性过多，我们只展示前 10 个属性
sum0_df[,1:10]
```

Level	Firm	Juicy	Sweet	Bitter	Sour	Crispy	Tasty	Grainy	Soft
P(No)|(No)	83	27	103	538	333	160	113	529	454
P(Yes)|(No)	42	18	32	47	132	55	32	51	106

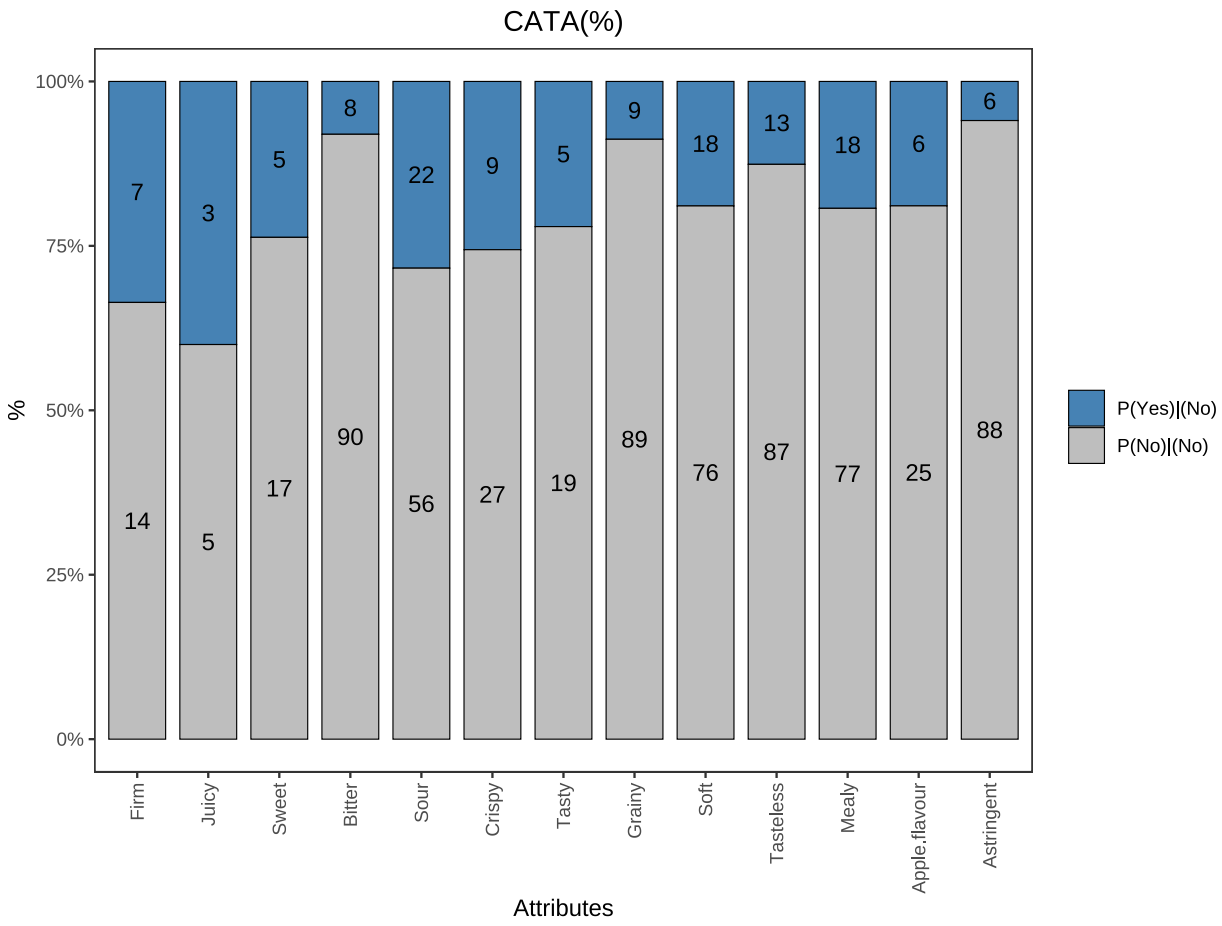
6.2.1.3 百分比柱形图 根据上小节得到的汇总表 `sum1_df` 与 `sum0_df`，用百分比柱形图进行可视化。同样，我自定义了百分比柱形图函数，由于函数源代码太长，限于篇幅，就不展示在此处，同时我将其打包成一个 R 文件，命名为 `Percent_bar.R`。执行以下代码可直接得出百分比柱形图：

```
# 运行函数源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Percent_bar.R")

```

```
#the must have attributes
Percent_bar(sum_df=sum1_df,ideal=TRUE)
```




```
#the nice to have and negative attributes
Percent_bar(sum_df=sum0_df,ideal=FALSE,color=c("steelblue","grey"),angle=90)
```



函数说明：`Percent_bar` 函数有三个参数：

- **sum_df**: 该参数为上一小节的汇总表结果；
- **color**: 该参数可以修改图形配色，默认值为 `c("green","red")`；
- **angle**: 该参数用于控制 x 轴标签的显示角度 (当 x 轴属性个数很多时很有用)，默认值为 60，即 x 轴标签倾斜 60 度。

6.2.1.4 对比表 (Comparison table)

同样由于构造对比表函数源代码太长，限于篇幅，就不在此处展示。我将其打包成一个 R 文件 'Comprison_table.R', 执行以下代码，输出对比表：

```
# 运行函数源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Comprison_table.R")

# 调用该函数，输出 Comprison table
#the must have attributes
comprison1_table<-Comprison_table(data,sum_df=sum1_df,ideal=1,index=3)
# 由于 A4 纸张宽度容不下表格所有列数，所以只展示前 7 列
comprison1_table[,1:7]
```

Variable	Level	Frequencies	Percent	Sum_Liking	Mean_Liking	Mean_drops
Firm	P(No)|(Yes)	225	37.82%	1261	5.604	1.514
Firm	P(Yes)|(Yes)	245	41.18%	1744	7.118	

Variable	Level	Frequencies	Percent	Sum_Liking	Mean_Liking	Mean_drops
Juicy	P(No)|(Yes)	226	37.98%	1141	5.049	2.161
Juicy	P(Yes)|(Yes)	324	54.45%	2336	7.21	
Sweet	P(No)|(Yes)	290	48.74%	1607	5.541	2.024
Sweet	P(Yes)|(Yes)	170	28.57%	1286	7.565	
Bitter	P(No)|(Yes)	6	1.01%	33	5.5	0.75
Bitter	P(Yes)|(Yes)	4	0.67%	25	6.25	
Sour	P(No)|(Yes)	79	13.28%	464	5.873	1.343
Sour	P(Yes)|(Yes)	51	8.57%	368	7.216	
Crispy	P(No)|(Yes)	202	33.95%	1146	5.673	1.569
Crispy	P(Yes)|(Yes)	178	29.92%	1289	7.242	
Tasty	P(No)|(Yes)	281	47.23%	1544	5.495	2.322
Tasty	P(Yes)|(Yes)	169	28.4%	1321	7.817	
Grainy	P(No)|(Yes)	13	2.18%	95	7.308	1.192
Grainy	P(Yes)|(Yes)	2	0.34%	17	8.5	
Soft	P(No)|(Yes)	24	4.03%	118	4.917	3.447
Soft	P(Yes)|(Yes)	11	1.85%	92	8.364	
Tasteless	P(No)|(Yes)	0	0%	0		
Tasteless	P(Yes)|(Yes)	0	0%	0		
Mealy	P(No)|(Yes)	25	4.2%	180	7.2	0
Mealy	P(Yes)|(Yes)	5	0.84%	36	7.2	
Apple.flavour	P(No)|(Yes)	253	42.52%	1414	5.589	1.806
Apple.flavour	P(Yes)|(Yes)	157	26.39%	1161	7.395	
Astringent	P(No)|(Yes)	32	5.38%	181	5.656	1.594
Astringent	P(Yes)|(Yes)	8	1.34%	58	7.25	

```
#the nice to have and negative attributes
comprison0_table<-Comprison_table(data,sum_df=sum0_df,ideal=0,index=3)
# 由于 A4 纸张宽度容不下表格所有列数，所以只展示前 7 列
comprison1_table[,1:7]
```

Variable	Level	Frequencies	Percent	Sum_Liking	Mean_Liking	Mean_drops
Firm	P(No)|(Yes)	225	37.82%	1261	5.604	1.514
Firm	P(Yes)|(Yes)	245	41.18%	1744	7.118	
Juicy	P(No)|(Yes)	226	37.98%	1141	5.049	2.161
Juicy	P(Yes)|(Yes)	324	54.45%	2336	7.21	
Sweet	P(No)|(Yes)	290	48.74%	1607	5.541	2.024
Sweet	P(Yes)|(Yes)	170	28.57%	1286	7.565	
Bitter	P(No)|(Yes)	6	1.01%	33	5.5	0.75
Bitter	P(Yes)|(Yes)	4	0.67%	25	6.25	
Sour	P(No)|(Yes)	79	13.28%	464	5.873	1.343
Sour	P(Yes)|(Yes)	51	8.57%	368	7.216	
Crispy	P(No)|(Yes)	202	33.95%	1146	5.673	1.569
Crispy	P(Yes)|(Yes)	178	29.92%	1289	7.242	
Tasty	P(No)|(Yes)	281	47.23%	1544	5.495	2.322
Tasty	P(Yes)|(Yes)	169	28.4%	1321	7.817	

Variable	Level	Frequencies	Percent	Sum_Liking	Mean_Liking	Mean_drops
Grainy	P(No)|(Yes)	13	2.18%	95	7.308	1.192
Grainy	P(Yes)|(Yes)	2	0.34%	17	8.5	
Soft	P(No)|(Yes)	24	4.03%	118	4.917	3.447
Soft	P(Yes)|(Yes)	11	1.85%	92	8.364	
Tasteless	P(No)|(Yes)	0	0%	0		
Tasteless	P(Yes)|(Yes)	0	0%	0		
Mealy	P(No)|(Yes)	25	4.2%	180	7.2	0
Mealy	P(Yes)|(Yes)	5	0.84%	36	7.2	
Apple.flavour	P(No)|(Yes)	253	42.52%	1414	5.589	1.806
Apple.flavour	P(Yes)|(Yes)	157	26.39%	1161	7.395	
Astringent	P(No)|(Yes)	32	5.38%	181	5.656	1.594
Astringent	P(Yes)|(Yes)	8	1.34%	58	7.25	

值得说明的是，被检验属性的显著性 p 值可能与 XLSTAT 所给示例不完全相同，这是因为我查不到其所用检验方法，所以依据我个人的一些基于统计学的知识，用 LSD 检验去检验属性的显著性。

函数说明：`Comprison_table` 函数有三个参数：

- **data**: 该参数为原始数据;
- **sum_df**: 该参数为上一小节的汇总表结果;
- **ideal**: 该参数有两种取值：
  - **1**: 该参数为 1 时表示属于 “Analysis of the must have attributes”;
  - **0**: 该参数为 0 时表示属于 “Analysis of the nice to have and negative attributes”;
- **index**: 该参数为数据中 **Liking** 列的列索引;

6.2.1.5 平均影响显示 (Mean impact display)

利用前面构造的 comprison_table，可以画出平均影响图。源代码如下所示：

```
# 利用 Comprison_table 绘图
Mean_impact<-function(comprison_table,color="blue"){
  # 获取有限值数据的索引
  ind_nona<-which(comprison_table$Mean_drops!=" "&comprison_table$Mean_drops>0)
  # 获取 Mean_drops 显著数据的索引
  ind_sign<-ind_nona[which(comprison_table[ind_nona,"P_value"]=="<0.0001")]
  # 返回排序后的索引
  ind_sort<-ind_sign[sort(comprison_table$Mean_drops[ind_sign],index.return=TRUE)$ix]
  # 将 Variable 变量转化为因子，便于图形排序
  meandrop_df<-comprison_table[ind_sort,]
  meandrop_df$Variable<-factor(meandrop_df$Variable,levels=meandrop_df$Variable)
  meandrop_df$Mean_drops<-as.numeric(meandrop_df$Mean_drops)
  # 绘图
  library(ggplot2)
  ggplot(data=meandrop_df,aes(x=Variable,y=Mean_drops))+
    geom_bar(stat="identity",fill=color)+
    labs(title="Mean impact",x="Attributes")+
    coord_flip()+  # 将坐标轴互换
```

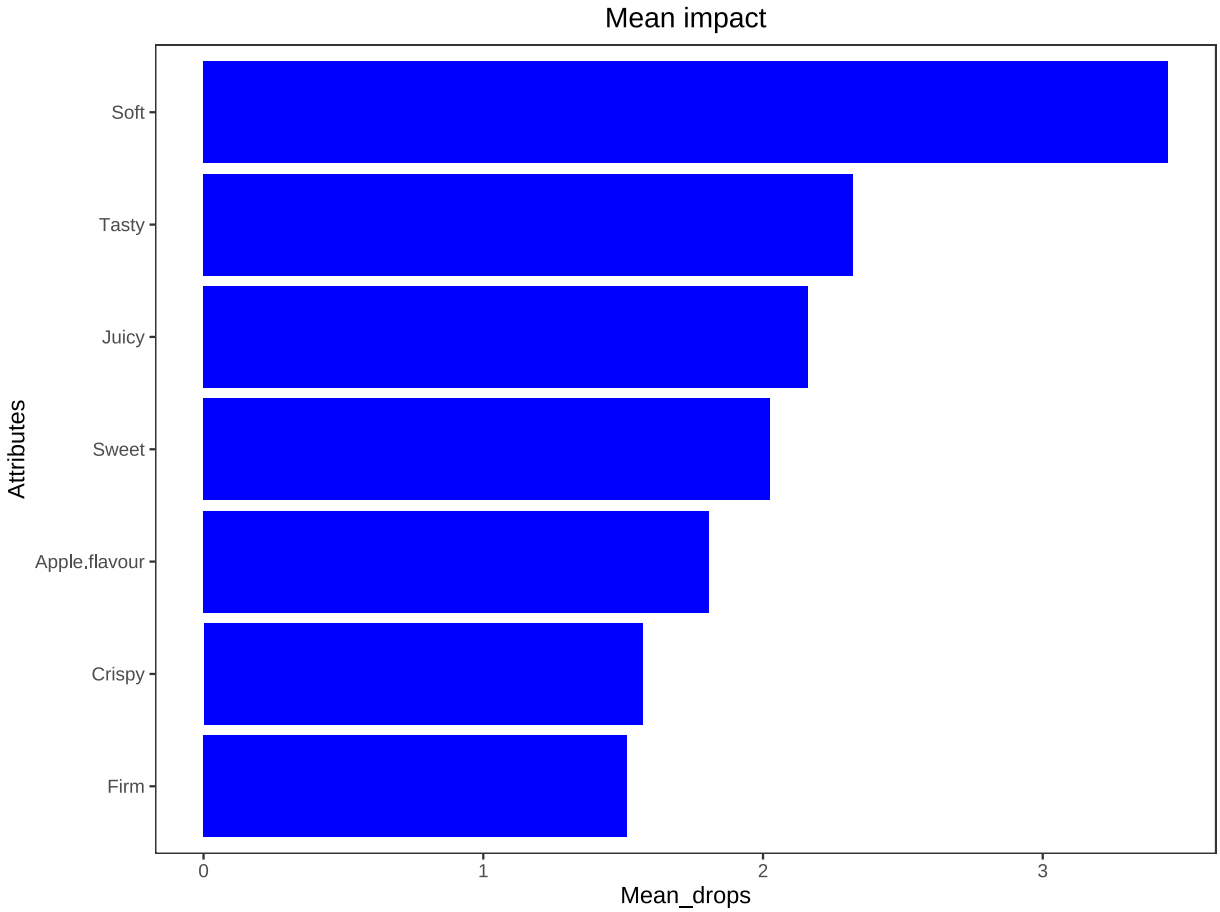
```
    theme(plot.title=element_text(hjust=0.5)) # 使坐标轴居中显示
}
```

函数说明：该函数有两个参数：

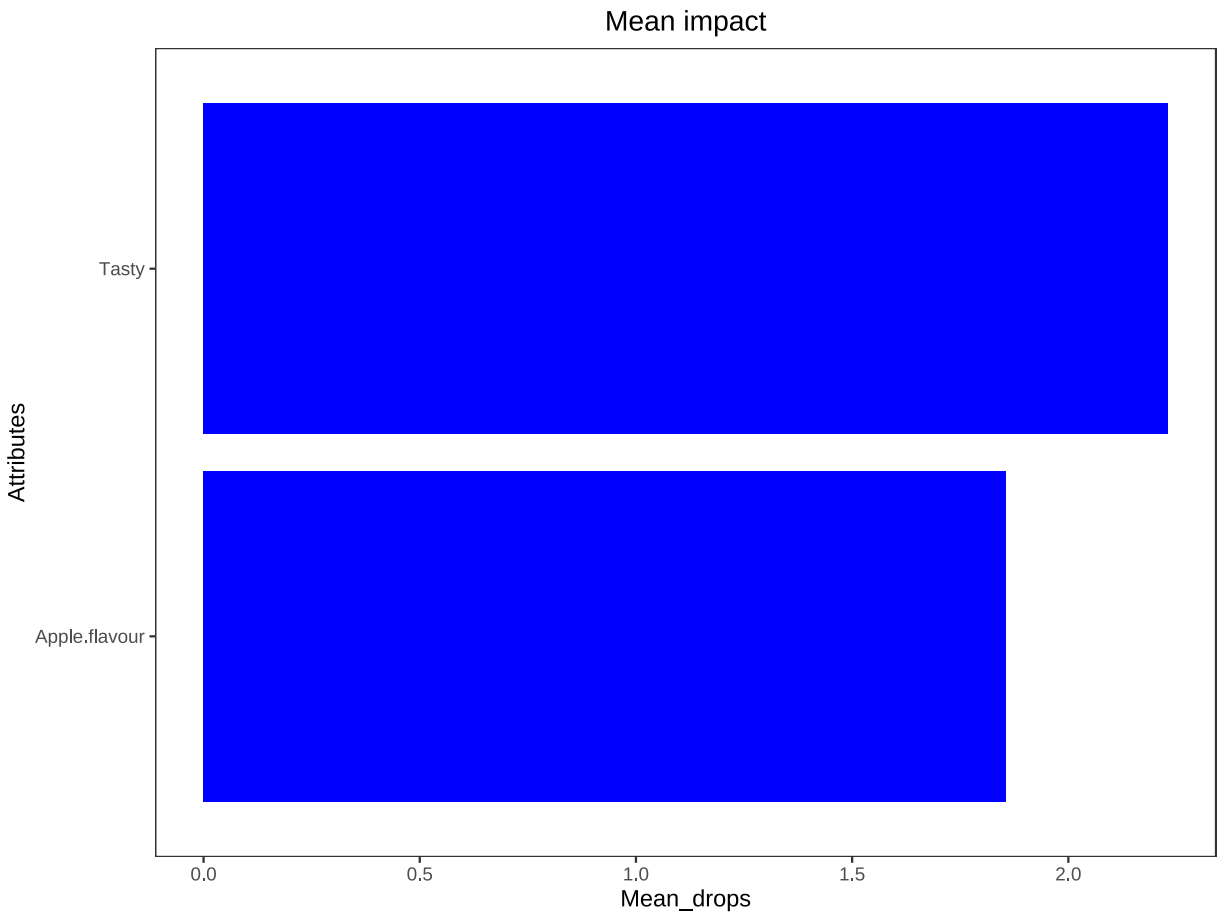
- **comprison_table**: 该参数为上一小节的对比表结果 (comprison table);
- **color**: 该参数指定条形图的颜色，默认颜色为蓝色 `blue`;

最后我们执行以下代码，查看绘图效果：

```
# 运行函数源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Mean_impact.R")

#the must have attributes
Mean_impact(comprison1_table)
```

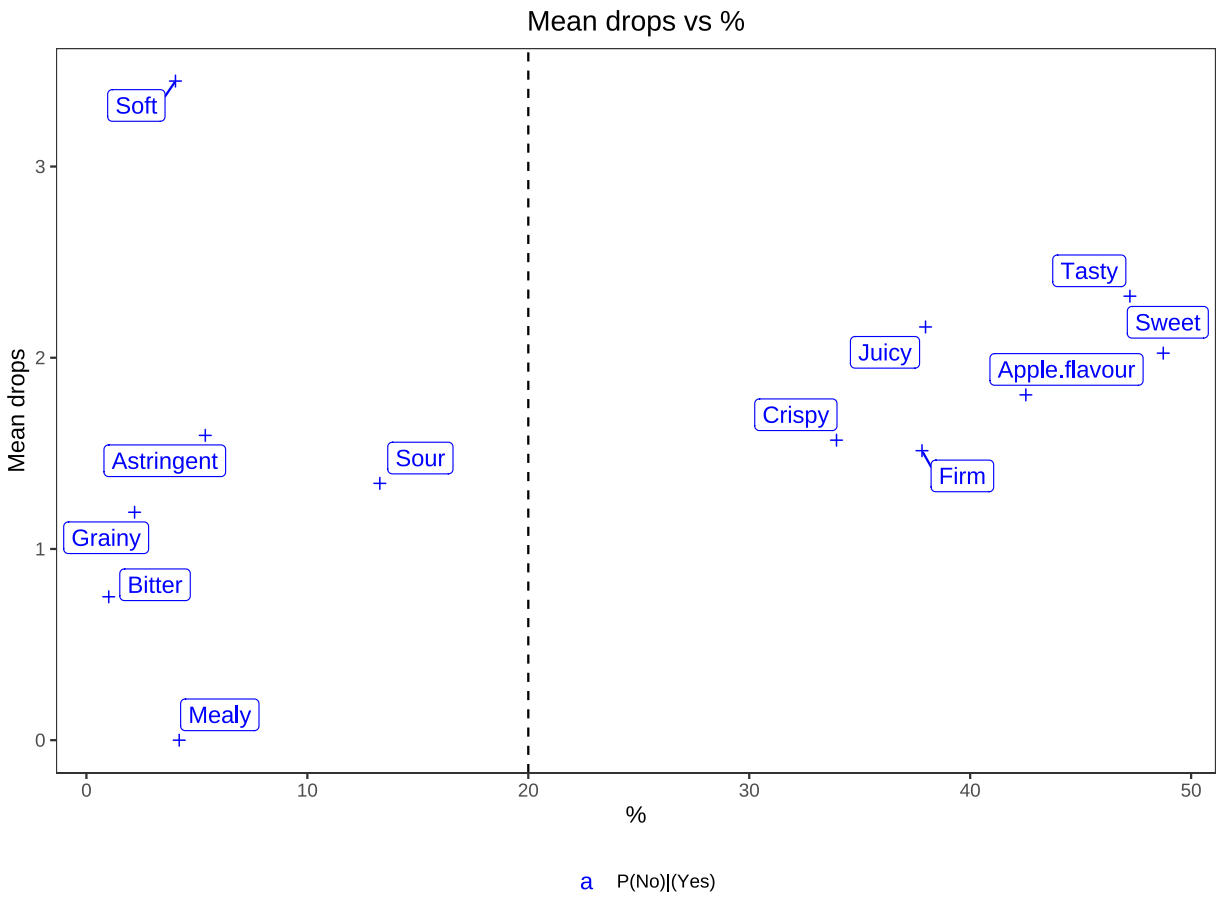



```
#the nice to have and negative attributes
Mean_impact(comprison0_table)
```



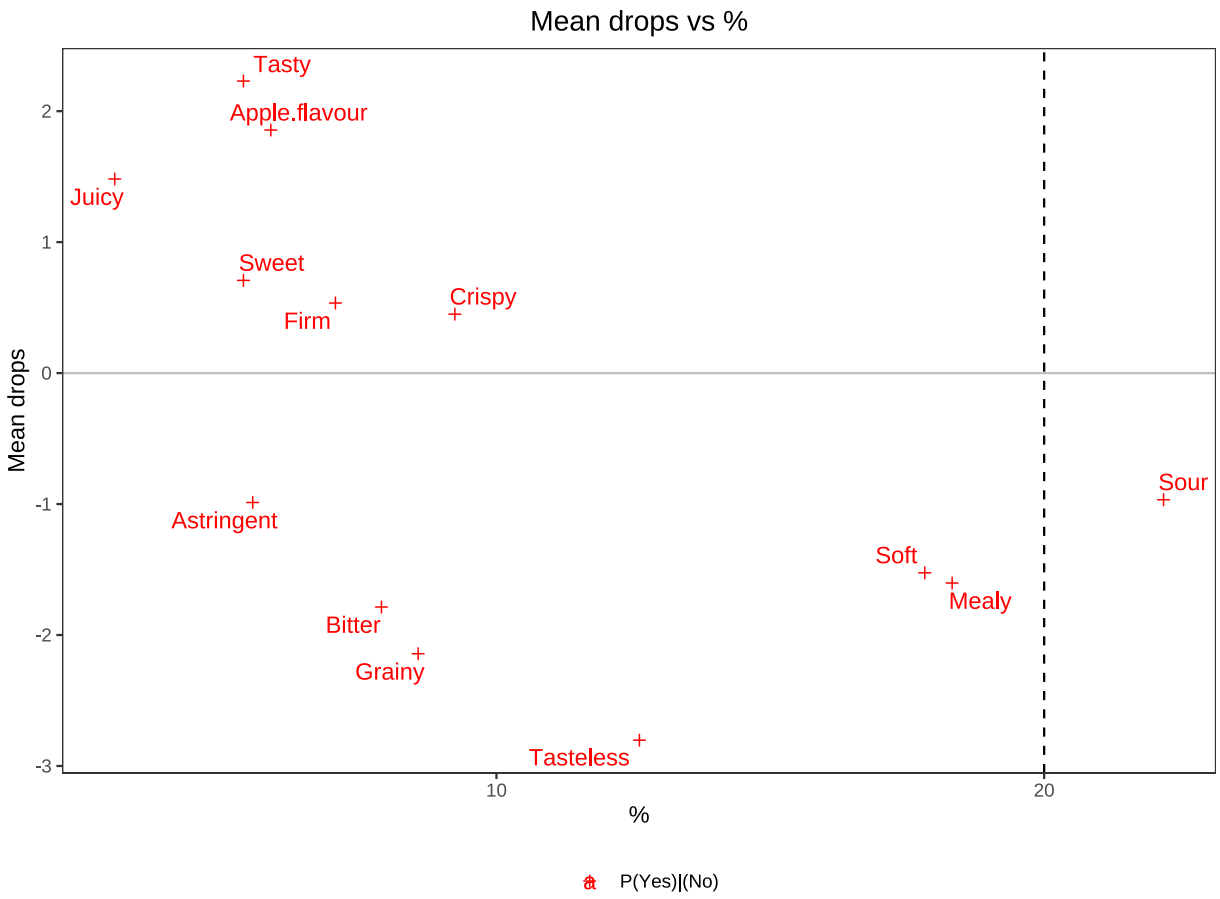
6.2.1.6 均值与百分比散点图 (Mean drops vs %) 同样我自定义了 `Meandrop_percent` 函数去绘制该图，该函数以对比表 (comprison_table) 为主要参数。我将其打包成一个 R 文件，下面请看代码示例：

```
# 运行该函数源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Meandrop_percent.R")

#the must have attributes
Meandrop_percent(comprison_table=comprison1_table)
```



a P(No)|(Yes)

```
#the nice to have and negative attributes
Meandrop_percent(comprison_table=comprison0_table,
                 color="red",label=FALSE)
```



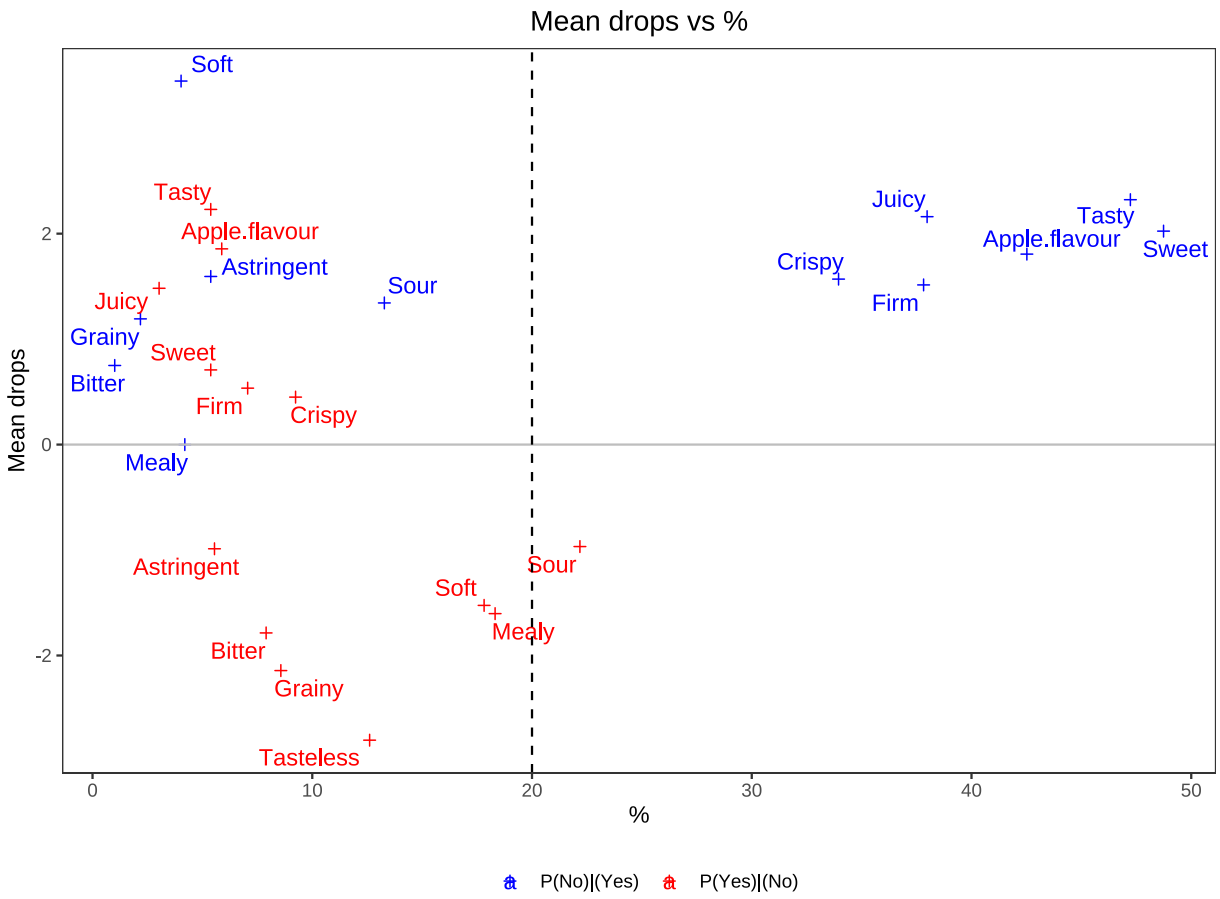
函数说明：该函数有三个可修改参数：

- **comprison_table**: 该参数为对比表 (comprison table);
- **color**: 该参数指定散点以及文本标签颜色，默认值为蓝色 `blue`;
- **label**: 该参数为逻辑参数，只有 `TRUE` 和 `FALSE` 两个值:
  - **TRUE**：为真时，代表给散点添加文本框标签;
  - **FALSE**：为假时，代表给散点添加文本标签，没有文本框。

6.2.2 汇总分析 (Penalty analysis summary)

自定义函数 `Meandrop_both` 可以实现汇总分析图，具体源代码请见 R 文件 `Meandrop_both.R`。**值得注意的是**，该函数适用了之前定义的 `Summary_tab` 函数和 `Comprision_table` 函数，所以适用该函数之前，要先运行上述两个函数的源文件，完整代码如下所示：

```
# 运行函数源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Summary_tab.R")
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Comprison_table.R")
# 注意上面两个函数要在 Meandrop_both 函数之前先运行，否则会报错
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Meandrop_both.R")

# 结果示例
Meandrop_both(data=data,index=4,label=FALSE)
```



函数说明：该函数有四个可修改参数：

- **data**: 该参数为用于分析的数据;
- **index**: 该参数为 data 中从左到右第一个属性所在列的列索引;
- **color**: 该参数指定散点以及文本标签颜色，默认值为 `c("red","blue")`;
- **label**: 该参数为逻辑参数，只有 `TRUE` 和 `FALSE` 两个值:
  - **TRUE**：为真时，代表给散点添加文本框标签;
  - **FALSE**：为假时，代表给散点添加文本标签，没有文本框。

6.2.3 属性分析 (Attributes analysis:)

6.2.3.1 属性四格表 自定义函数 `Attr_ana` 可实现根据分析数据与属性名称，输出属性四格表，函数源代码保存在 R 文 `Attr_ana.R` 中。实现示例如下：

```
# 运行函数源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Attr_ana.R")

# 输出属性名为 Mealy 四格表
Attr_ana(data=data,attr="Mealy")
```

```
## $table
##     Mealy Product_0 Product_1
## 1 Ideal_0  6.6(77%)    5(18%)
## 2 Ideal_1   7.2(4%)   7.2(1%)
```

```
##
## $discrible
## [1] "This attribute is a must not have"
```

```
# 输出属性名为 Sweet 四格表
Attr_ana(data=data,attr="Sweet")
```

```
## $table
##     Sweet Product_0 Product_1
## 1 Ideal_0  6.4(17%)   7.1(5%)
## 2 Ideal_1  5.5(49%)  7.6(29%)
##
## $discrible
## [1] "This attribute is a must have"
```

函数说明：该函数只有两个可修改参数：

- **data**: 该参数为用于分析的数据;
- **attr**: 该参数为需要分析的属性名称;

可以发现该函数的返回值有两个部分，第一个部分是属性的汇总四格表, 第二个是属性的分类情况。需要说明的是，属性分类可能不完全准确，这是因为我能够查阅到的分类边界很模糊，无法找到更清晰的分类依据。

6.2.3.2 属性分类汇总结果

在自定义函数 `Attr_ana` 的基础上, 自定义函数 `Summary_attr` 可以很容易实现属性分类汇总。同样，在使用此函数之前一定要保证 `Attr_ana` 函数已经运行过，否则会报错。该函数保存在名为 `Summary_attr.R` 的 R 文件中。函数示例如下：

```
# 运行函数源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Attr_ana.R")
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Summary_attr.R")

# 输出分类汇总结果
Summ_attr_df<-Summary_attr(data=data,index=4)
Summ_attr_df
```

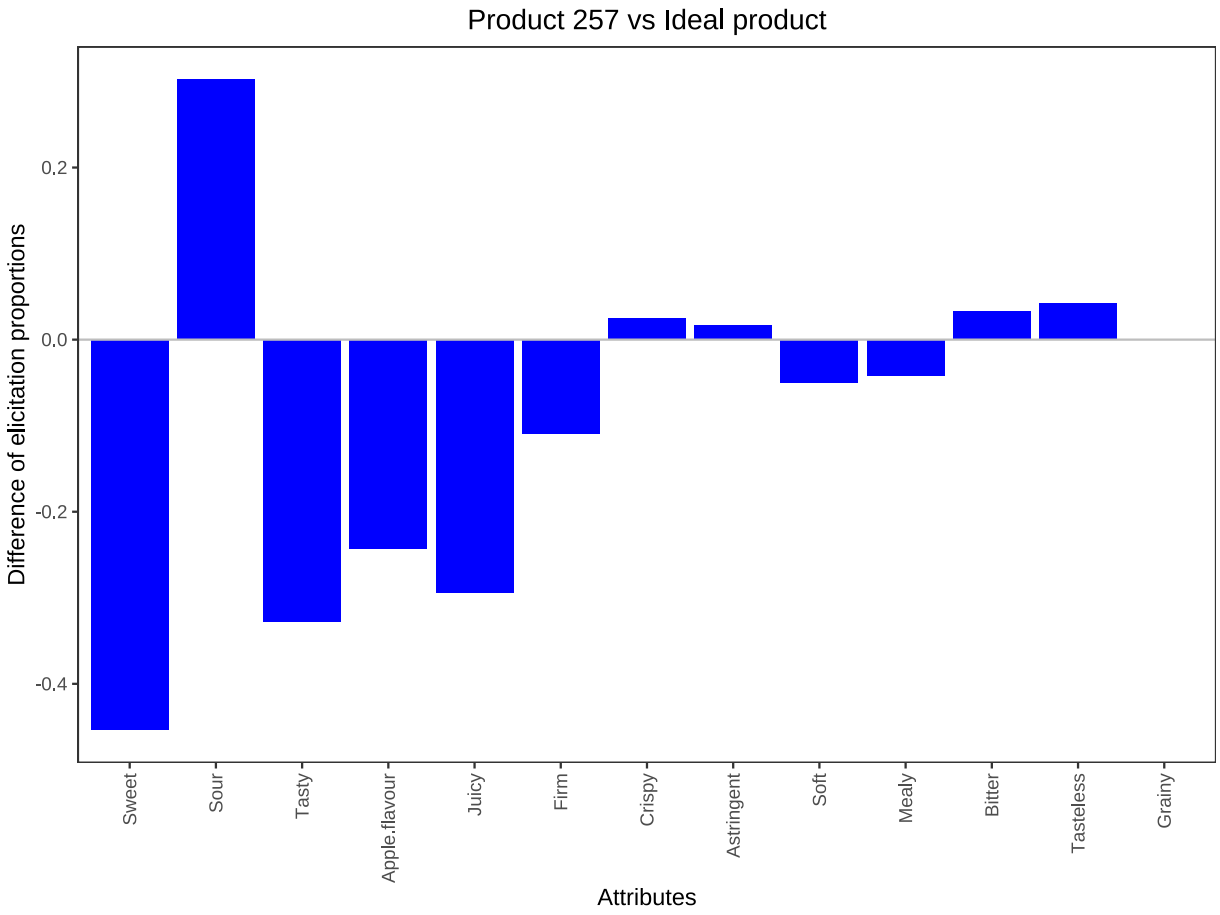
Must_have	Does_not_harm	Must_not_have
Firm	Sour	Bitter
Juicy	Astringent	Grainy
Sweet		Soft
Crispy		Tasteless
Tasty		Mealy
Apple.flavour		

函数说明：该函数只有两个可修改参数：

- **data**: 该参数为用于分析的数据;
- **index**: 该参数为 data 中从左到右第一个属性所在列的列索引。

6.2.3.3 诱发率比较 (Comparison of elicitation rates) 自定义函数 Comp_product_ideal 可以实现理想产品与实际产品的诱发率比较条形图。**但是由于无法知晓计算诱发率差异的置信区间的方法，所以没有如 XLSTAT 上所给出的两条虚线。**同时，该函数保存在名为 Comp_product_ideal.R 的 R 文件中。代码运行示例如下：

```
# 运行函数源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Comp_product_ideal.R")

# 调用函数输出绘图
Comp_product_ideal(data=data,product="257",index=4,color="blue",angle=90)
```



函数说明：该函数有五个可修改参数：

- **data**: 该参数为用于分析的数据;
- **index**: 该参数为 data 中从左到右第一个属性所在列的列索引;
- **product**: 该参数指定与理想产品作比较的产品名称;
- **color**: 指定条形图绘图颜色;
- **angle**: 指定 x 轴标签的倾斜角度，当属性过多时很有用。

6.3 无理想点数据

6.3.1 数据导入

我使用去除理想点数据的 CATA 实例文件数据，并只分析前面 24 个属性。执行代码如下：

```
# 导入数据
# 由于路径太长，显示文本容易溢出，所以使用 paste 函数将字符串格式路径分割一下
data<-read.csv(paste("C:\\Users\\里高房价\\Desktop\\",
      "CATA&TCATA\\CATA\\CATA 实例（无理想点）.csv",sep=""))

data<-data[,1:28]  # 只分析前面 24 个属性，一共 28 列
```

6.3.2 汇总表 (Summary table)

我定义了 Summary_tab_N 函数，该函数直接返回汇总表。同理，我将该函数打包成一个 R 文件，便于直接调用。下面是代码效果示例：

```
# 运行代码源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Summary_tab_N.R")

tab_N<-Summary_tab_N(data,index=5)

# 由于纸张限制，我们只输出 10 列进行显示
tab_N[,1:10]
```

Level	Sticky	Caramel	Brown	Bubble	Sweet	Jujube	Milk	Grain	Acid
Absent	257	309	260	340	303	376	237	404	160
Present	212	160	209	129	166	93	232	65	309

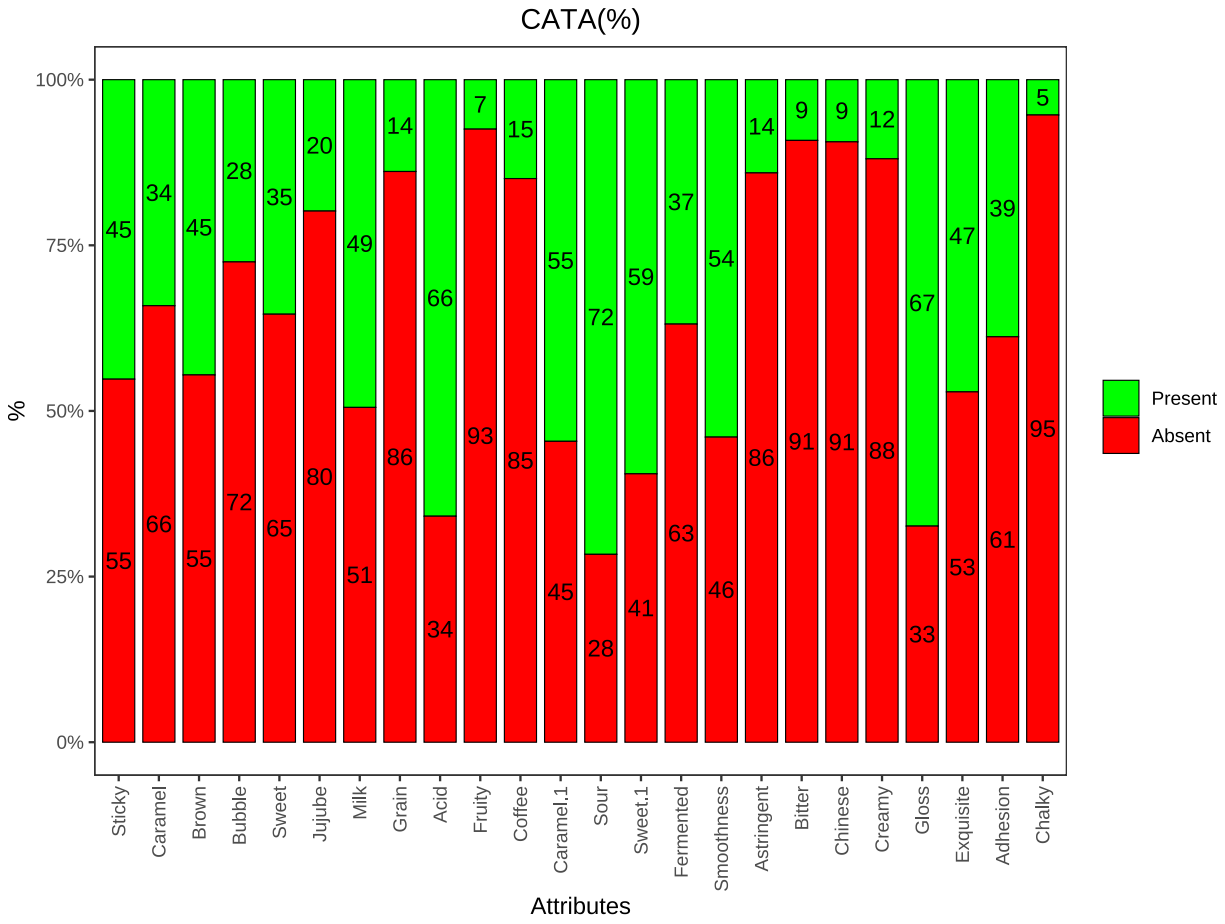
函数说明：该函数有三个参数：

- **data**: 分析数据
- **index**: 分析数据中从左到右第一个属性的列索引

6.3.3 百分比柱形图

根据上小节得到的汇总表 tab_N，用百分比柱形图进行可视化。同样，我自定义了百分比柱形图函数，由于函数源代码太长，限于篇幅，就不展示在此处，同时我将其打包成一个 R 文件，命名为 Percent_bar_N.R。执行以下代码可直接得出百分比柱形图：

```
# 运行函数源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Percent_bar_N.R")

Percent_bar_N(table=tab_N,angle=90)
```



函数说明： `Percent_bar_N` 函数有三个参数：

- **table**: 该参数为上一小节的汇总表结果；
- **color**: 该参数可以修改图形配色，默认值为 `c("green","red")`；
- **angle**: 该参数用于控制 x 轴标签的显示角度 (当 x 轴属性个数很多时很有用)，默认值为 60，即 x 轴标签倾斜 60 度。

6.3.4 对比表 (Comparison table)

同样由于构造对比表函数源代码太长，限于篇幅，就不在此处展示。我将其打包成一个 R 文件 'Comprison_table_N.R', 执行以下代码，输出对比表：

```
# 运行函数源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Comprison_table_N.R")

# 调用该函数，输出 Comprison table
comprison_table<-Comprison_table_N(data=data,table=tab_N,index=4)

# 由于 A4 纸张宽度容不下表格所有列数，所以只展示前 7 列
comprison_table[,1:7]
```

Variable	Level	Frequencies	Percent	Sum_Liking	Mean_Liking	Mean_drops
Sticky	Absent	257	54.8%	1308	5.089	
Sticky	Present	212	45.2%	1299	6.127	1.038
Caramel	Absent	309	65.88%	1640	5.307	
Caramel	Present	160	34.12%	967	6.044	0.737
Brown	Absent	260	55.44%	1419	5.458	
Brown	Present	209	44.56%	1188	5.684	0.226
Bubble	Absent	340	72.49%	1870	5.500	
Bubble	Present	129	27.51%	737	5.713	0.213
Sweet	Absent	303	64.61%	1594	5.261	
Sweet	Present	166	35.39%	1013	6.102	0.841
Jujube	Absent	376	80.17%	2166	5.761	
Jujube	Present	93	19.83%	441	4.742	-1.019
Milk	Absent	237	50.53%	1178	4.970	
Milk	Present	232	49.47%	1429	6.159	1.189
Grain	Absent	404	86.14%	2269	5.616	
Grain	Present	65	13.86%	338	5.200	-0.416
Acid	Absent	160	34.12%	782	4.888	
Acid	Present	309	65.88%	1825	5.906	1.018
Fruity	Absent	434	92.54%	2432	5.604	
Fruity	Present	35	7.46%	175	5.000	-0.604
Coffee	Absent	399	85.07%	2247	5.632	
Coffee	Present	70	14.93%	360	5.143	-0.489
Caramel.1	Absent	213	45.42%	1114	5.230	
Caramel.1	Present	256	54.58%	1493	5.832	0.602
Sour	Absent	133	28.36%	705	5.301	
Sour	Present	336	71.64%	1902	5.661	0.36
Sweet.1	Absent	190	40.51%	952	5.011	
Sweet.1	Present	279	59.49%	1655	5.932	0.921
Fermented	Absent	296	63.11%	1617	5.463	
Fermented	Present	173	36.89%	990	5.723	0.26
Smoothness	Absent	216	46.06%	1097	5.079	
Smoothness	Present	253	53.94%	1510	5.968	0.889
Astringent	Absent	403	85.93%	2294	5.692	
Astringent	Present	66	14.07%	313	4.742	-0.95
Bitter	Absent	426	90.83%	2428	5.700	
Bitter	Present	43	9.17%	179	4.163	-1.537
Chinese	Absent	425	90.62%	2445	5.753	
Chinese	Present	44	9.38%	162	3.682	-2.071
Creamy	Absent	413	88.06%	2255	5.460	
Creamy	Present	56	11.94%	352	6.286	0.826
Gloss	Absent	153	32.62%	765	5.000	
Gloss	Present	316	67.38%	1842	5.829	0.829
Exquisite	Absent	248	52.88%	1213	4.891	
Exquisite	Present	221	47.12%	1394	6.308	1.417
Adhesion	Absent	287	61.19%	1551	5.404	
Adhesion	Present	182	38.81%	1056	5.802	0.398

Variable	Level	Frequencies	Percent	Sum_Liking	Mean_Liking	Mean_drops
Chalky	Absent	444	94.67%	2498	5.626	
Chalky	Present	25	5.33%	109	4.360	-1.266

值得说明的是，被检验属性的显著性 p 值可能与 XLSTAT 所给示例不完全相同，这是因为我查不到其所用检验方法，所以依据我个人的一些基于统计学的知识，用 LSD 检验去检验属性的显著性。

函数说明： Comprison_table_N 函数有三个参数：

- **data**: 该参数为原始数据;
- **table**: 该参数为上一小节的汇总表结果;
- **index**: 该参数为数据中 **Liking** 列的列索引;

6.3.5 平均影响显示 (Mean impact display)

利用前面构造的 comprison_table，可以画出平均影响图。源代码如下所示：

```
# 利用 Comprison_table 绘图
Mean_impact<-function(comprison_table,color="blue"){
  # 获取有限值数据的索引
  ind_nona<-which(comprison_table$Mean_drops!=" "&comprison_table$Mean_drops>0)
  # 获取 Mean_drops 显著数据的索引
  ind_sign<-ind_nona[which(comprison_table[ind_nona,"P_value"]=="<0.0001")]
  # 返回排序后的索引
  ind_sort<-ind_sign[sort(comprison_table$Mean_drops[ind_sign],index.return=TRUE)$ix]
  # 将 Variable 变量转化为因子，便于图形排序
  meandrop_df<-comprison_table[ind_sort,]
  meandrop_df$Variable<-factor(meandrop_df$Variable,levels=meandrop_df$Variable)
  meandrop_df$Mean_drops<-as.numeric(meandrop_df$Mean_drops)
  # 绘图
  library(ggplot2)
  ggplot(data=meandrop_df,aes(x=Variable,y=Mean_drops))+
    geom_bar(stat="identity",fill=color)+
    labs(title="Mean impact",x="Attributes")+
    coord_flip()+  # 将坐标轴互换
    theme_bw()+
    theme(panel.grid=element_blank(),
          plot.title=element_text(hjust=0.5))
}
```

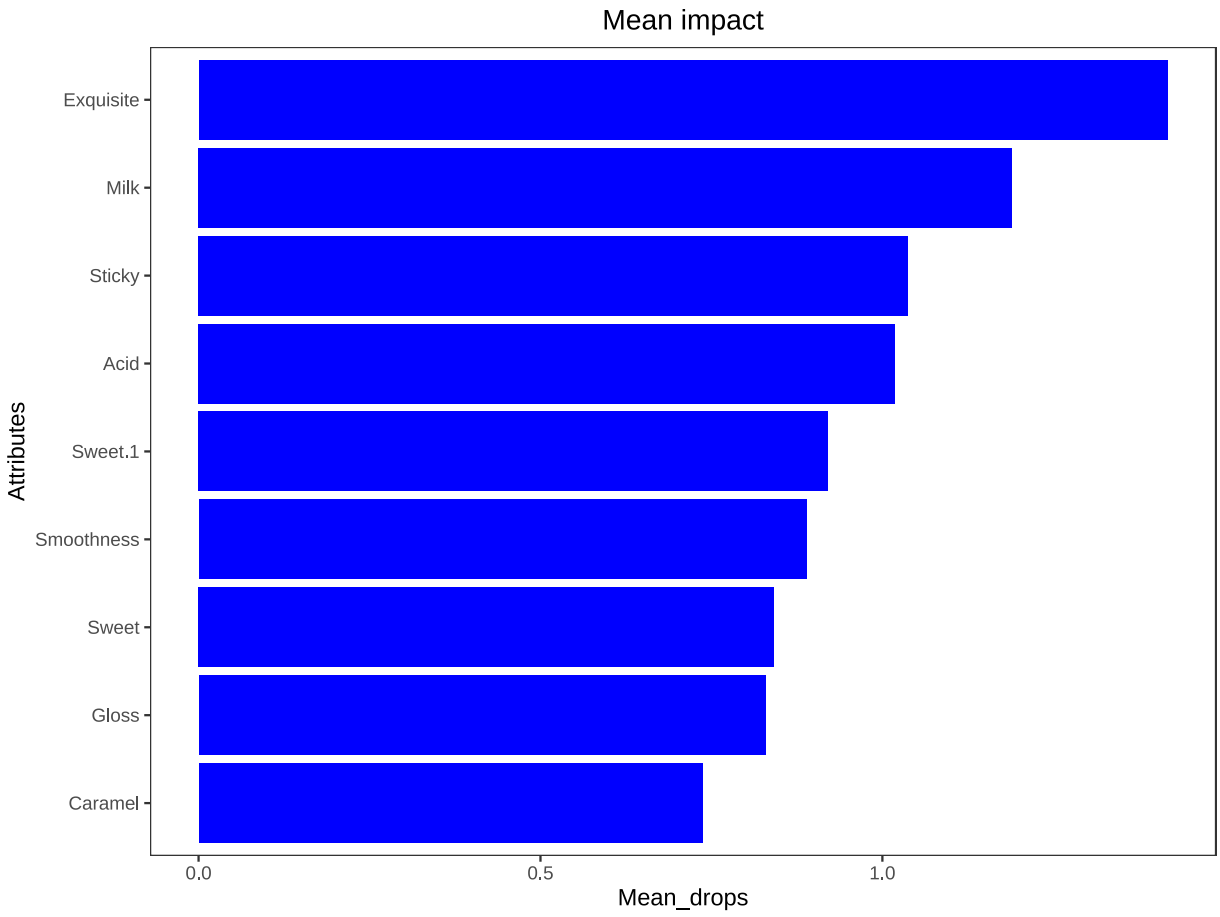
函数说明： 该函数有两个参数：

- **comprison_table**: 该参数为上一小节的对比表结果 (comprison table);
- **color**: 该参数指定条形图的颜色，默认颜色为蓝色 blue;

最后我们执行以下代码，查看绘图效果：

```
# 运行函数源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Mean_impact.R")
```

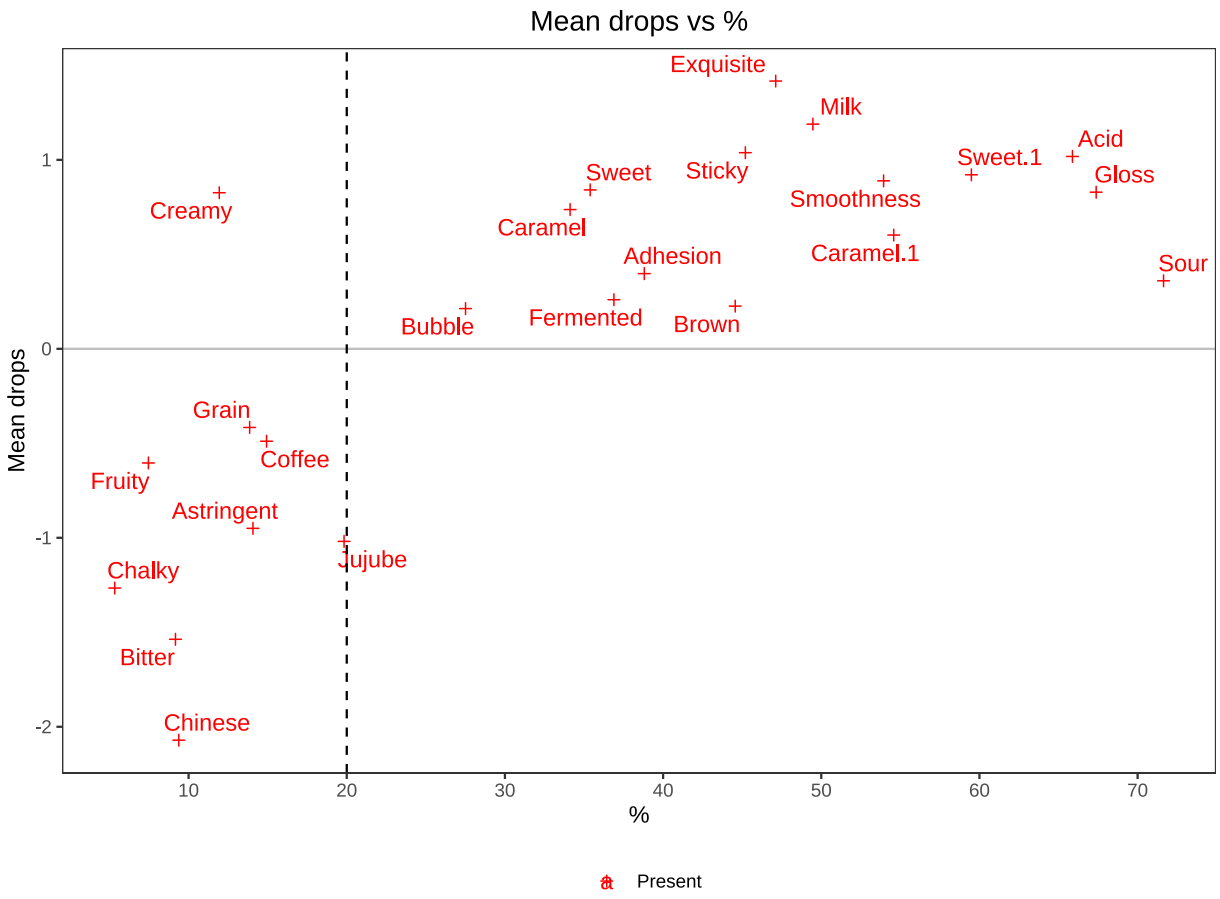
```
Mean_impact(comprison_table=comprison_table)
```



6.3.6 均值与百分比散点图 (Mean drops vs %)

同样我自定义了 Meandrop_percent 函数去绘制该图, 该函数以对比表 comprison_table 为主要参数。我将其打包成一个 R 文件，下面请看代码示例:

```
# 运行该函数源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\CATA\\Meandrop_percent.R")

Meandrop_percent(comprison_table=comprison_table,color="red",label=FALSE)
```



函数说明：该函数有三个可修改参数：

- **comprison_table**: 该参数为对比表 (comprison table);
- **color**: 该参数指定散点以及文本标签颜色，默认值为蓝色 `blue`;
- **label**: 该参数为逻辑参数，只有 `TRUE` 和 `FALSE` 两个值:
  - **TRUE**：为真时，代表给散点添加文本框标签;
  - **FALSE**：为假时，代表给散点添加文本标签，没有文本框。

7 文献分析方法补充

7.1 凝聚层次聚类 (Agglomerative Hierarchical Clustering)

我使用的数据为上述方法的 CATAdemo 数据。

7.1.1 R 包的准备

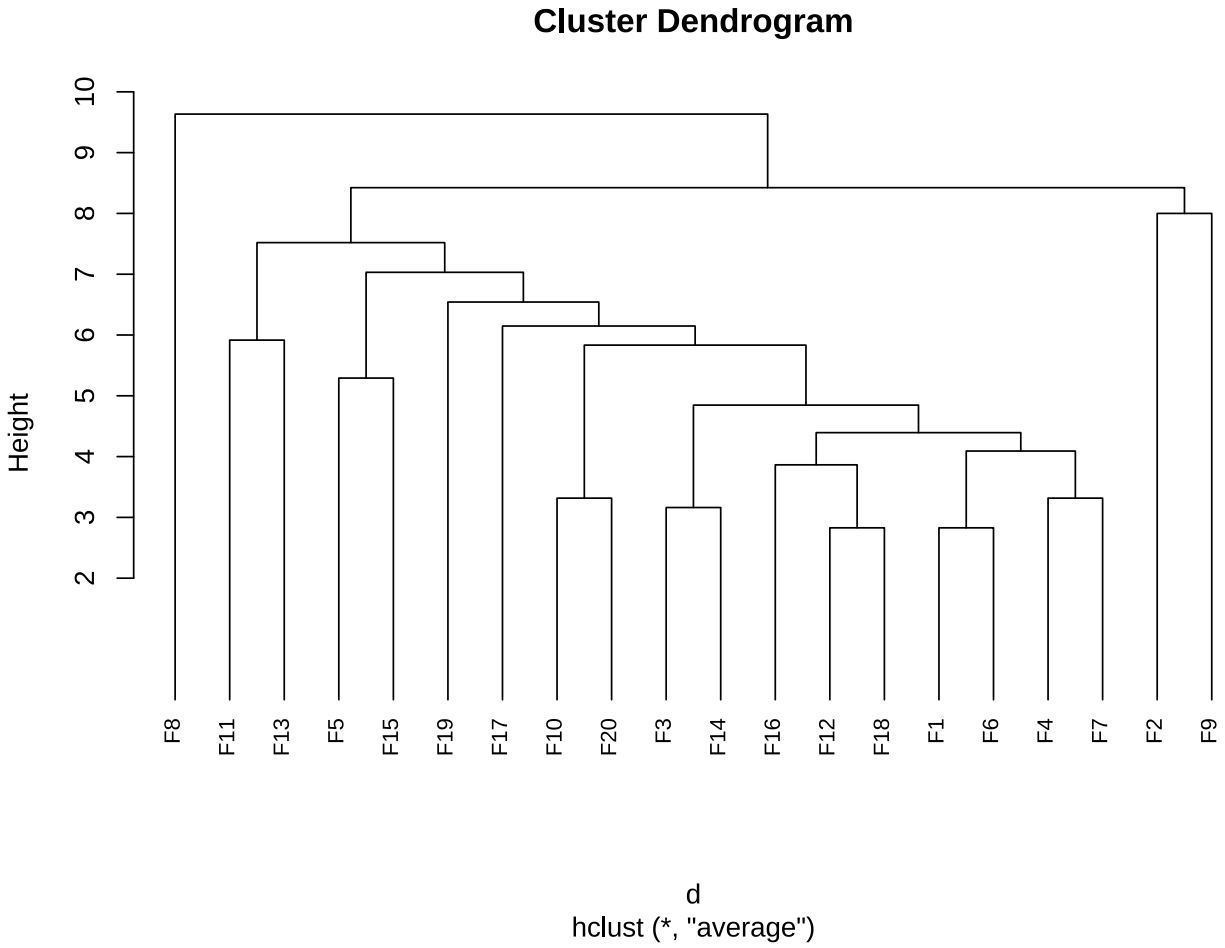
我们需要使用到两个 R 包:

- dendextend: 用于系统树图可视化的扩展包
- reshape2: 用于数据整合与重构

7.1.2 聚类可视化

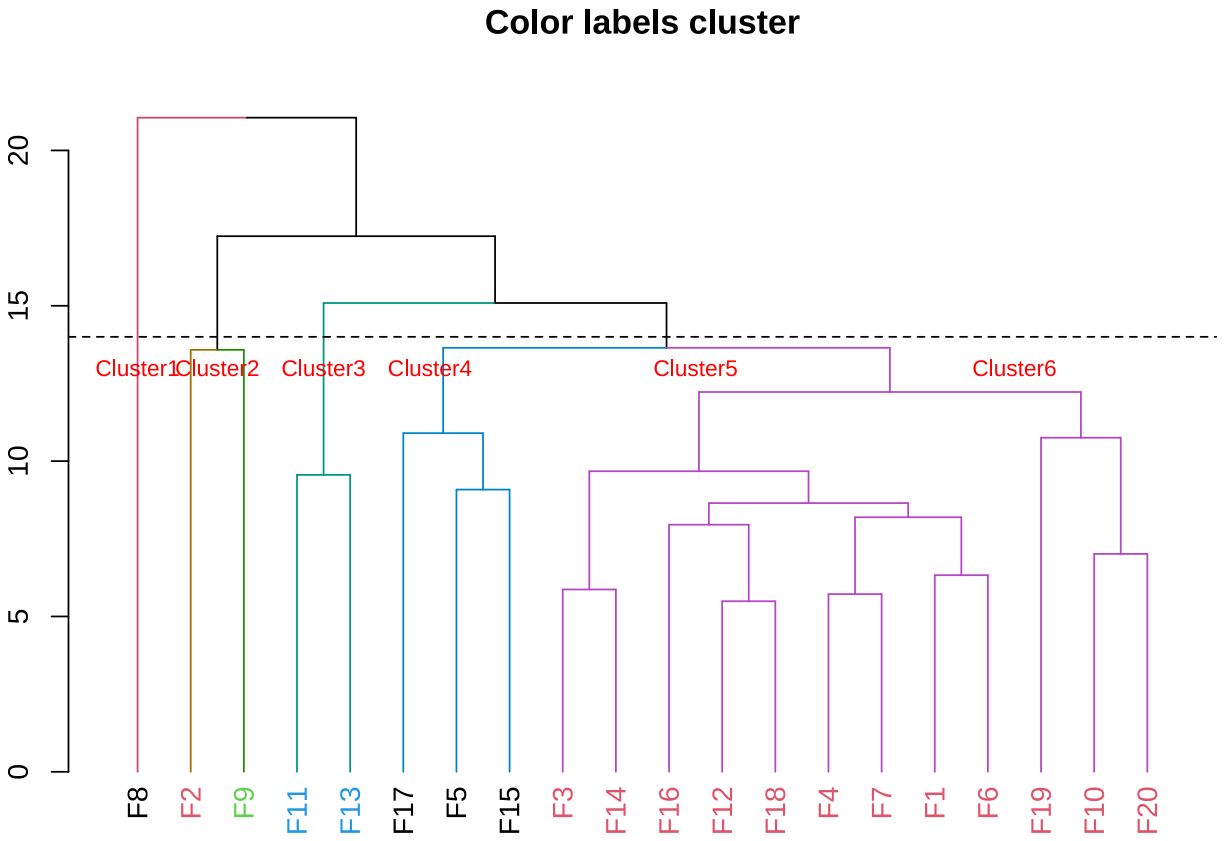
适用 R 自带函数 hclust() 进行层次聚类，并利用 plot() 函数绘制层次聚类图。注意传递给 hclust() 的聚类参数是一个距离矩阵，利用 R 自带函数 dist() 就可以根据数据计算出距离矩阵。下面是代码示例：

```
library(reshape2)

# 数据整合与重构
m_data=melt(data[data$Product!="Ideal",],id=c("Consumer","Product"),
            measure=c('Liking'))
d_data=dcast(m_data,Consumer~Product+variable)

# 选择部分数据用于聚类
AHC<-d_data[1:20,2:ncol(d_data)]

# 赋予行名
rownames(AHC)=paste("F",1:20,sep="")

# 计算聚类数据的欧几里得距离
d<-dist(AHC)

# 传递距离参数，进行层次聚类，使用平均法
h_average<-hclust(d,method="average")
# 可视化聚类结果
plot(h_average,hang=-1,cex=0.8,frame.plot=FALSE)
```



可以发现，R 自带绘图函数 `plot()` 的绘图不够灵活，而且不够美观。所以使用 R 包 `dendextend` 树图可视化扩展包绘制层次聚类图：

```
# 扩展绘图
#%>% 是一种管道符
library(dendextend)  # 系统树图的扩展绘图包

dend <- d %>% dist %>% hclust(method = "average") %>% as.dendrogram

# 根据聚类结果将其分为 6 类，利用颜色区分
dend %>% set("labels_cex", 1) %>% set("labels_col", value = c(1:4), k=6) %>%
  set("branches_k_color", k = 6) %>%
  #set("branches_lwd", c(1,2,4)) %>%
  plot(main = "Color labels cluster",horiz=FALSE,leaflab="perpendicular")
abline(h = 14, lty = 2)

# 添加分类标签
label_x<-c(1,2.5,4.5,6.5,11.5,17.5)
label_y<-rep(13,6)
label_text<-paste("Cluster",1:6,sep="")
text(x=label_x,y=label_y,labels=label_text,col="red",cex=0.8)
```



7.2 组合条形图

7.2.1 数据整理

我使用数据为 XLSTAT 的 demo 数据。下面代码是将原始数据整理为我需要的绘图数据。

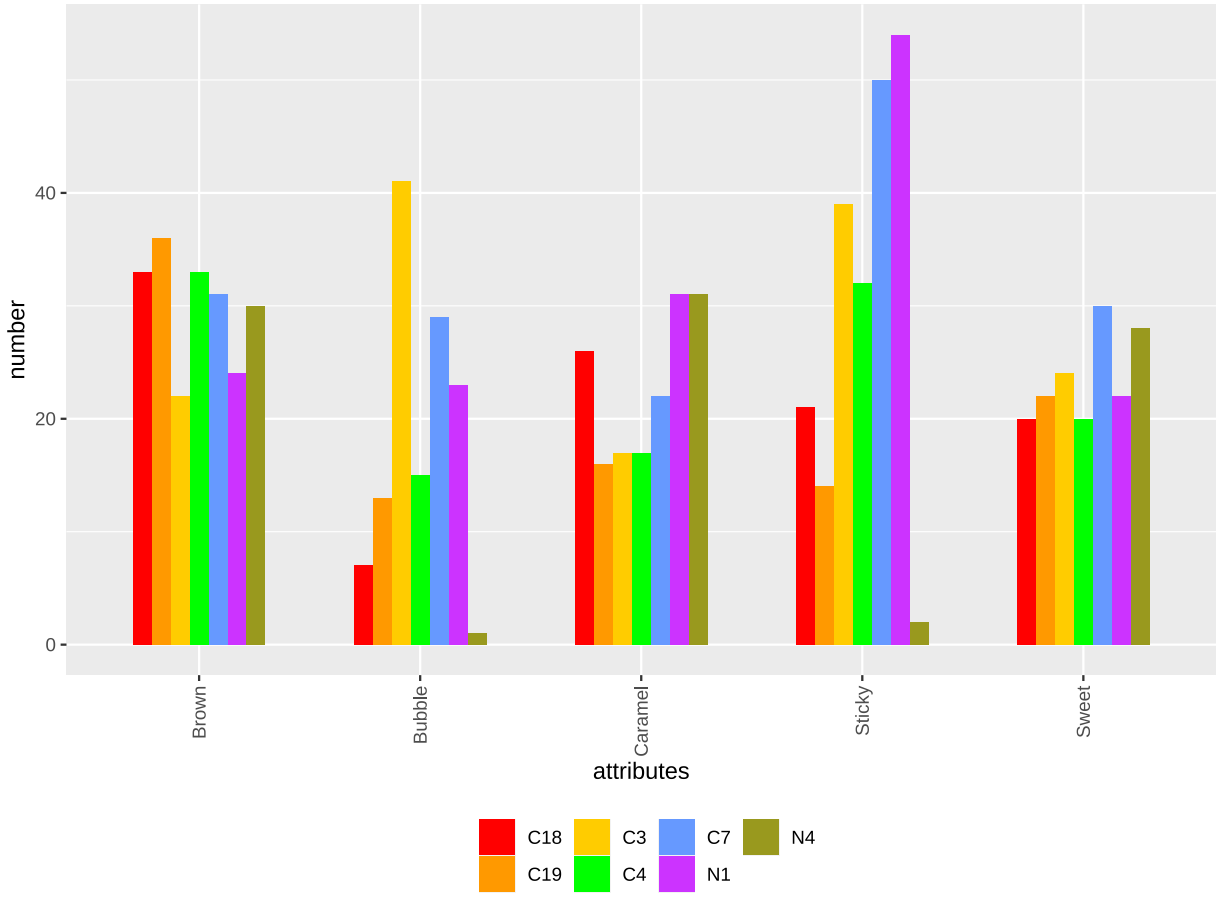
```
index=4
# 调用之前的构建列联表函数
Contingency.table <-aggregate(data[,-c(1:index)],
                  by=list(Product=data[,"Product"]),sum,na.rm=TRUE)
```

7.2.2 绘图

根据列联表数据，相同属性的产品为一组，绘制分组条形图：

```
## 条形图
library(ggplot2)
library(ggsci)
# 绘图数据处理
variables <- colnames(data[,-c(1:4)])
plot.variables <- variables[1:5]

# 提取处用来绘图的数据
temp<-Contingency.table[,c("Product",plot.variables)]
bar_data<-melt(temp,id=c("Product"),variable.name="attributes",
```

```
                value.name="number")


# 由于属性名太长，不易显示，我使用以下代码只截取属性英文
Attr<-c()
for(attr in as.character(bar_data$attributes)){
  # 使用正则表达式提取
  a<-strsplit(attr,split="[.]",perl=TRUE)[[1]][1]
  Attr<-c(Attr,a)
}
bar_data$attributes<-Attr

# 绘图
ggplot(data=bar_data,aes(x=attributes,y=number,fill=Product))+
  # 同一属性不同产品并排显示
  geom_bar(stat="identity",position="dodge",width=0.6,size=0.25)+
  scale_fill_ucscgb()+
  guides(x = guide_axis(angle = 90))+
  theme(legend.position="bottom",        # 调整图例位置与方向
        legend.direction = "horizontal",
        legend.title=element_blank())
```



7.3 惩罚提升分析 (Penalty-Lift Analysis)

应用该分析的目的是绘制正负条形图，参考论文 ***11 Check-All-That-Apply Questions*** 中的 **Figure 16**。

7.3.1 数据说明

我使用的数据是 `Dataset_Chapter11.txt` 数据，该数据是从上述论文中获得。首先需要将其导入到 R 中：

```
# 导入数据
data<- read.table(file="E:\\Dataset_Chapter11.txt",header=T,sep="\t",fill=T)

# 填充缺失值
data[is.na(data[,ncol(data)]),ncol(data)]<-0
```

7.3.2 数据整理

将原始数据进行整理，计算提升度 (Lift)：

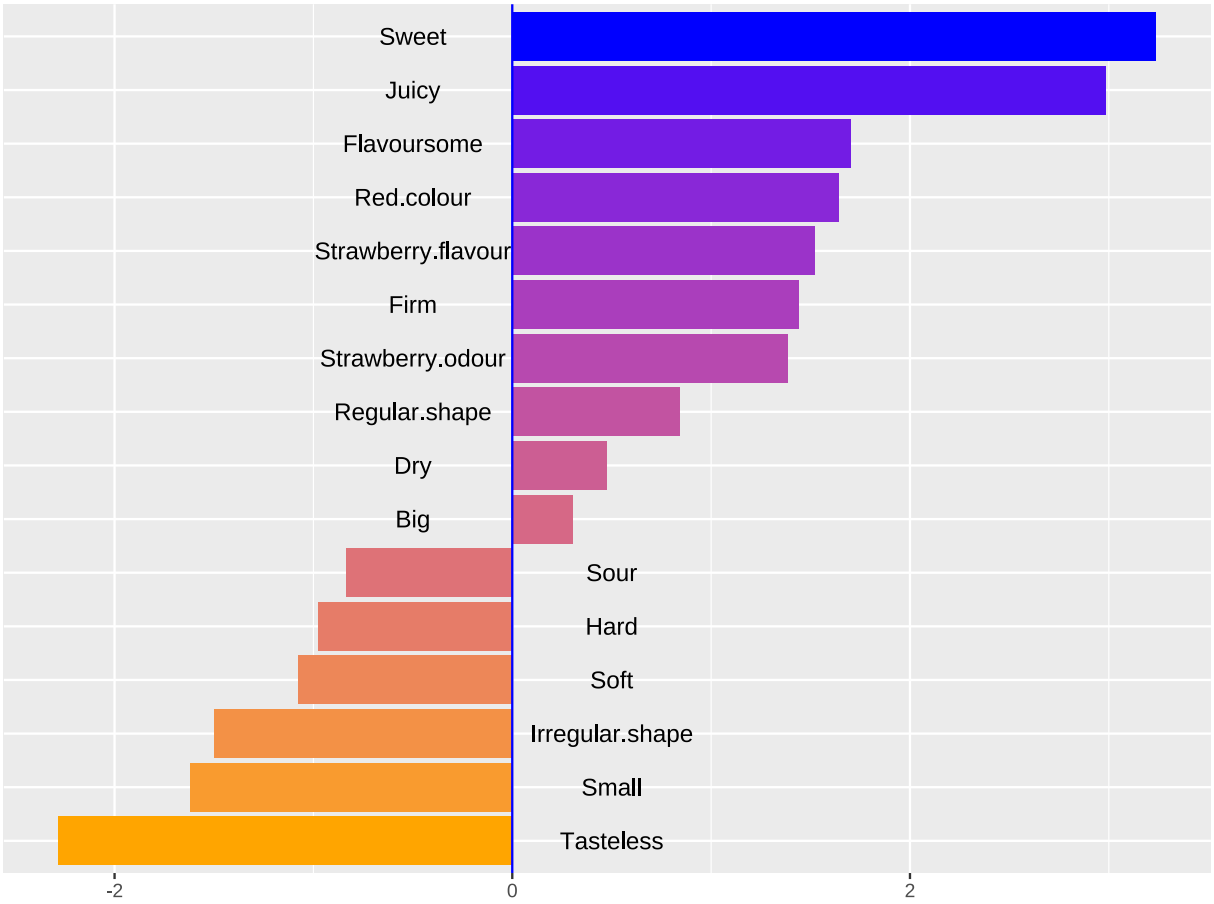
```
# 计算提升度 Lift
attr_all<-colnames(data[4:ncol(data)])
lift<-c()
for(attr in attr_all){
  pos<-mean(data[data[,attr]==1,]$Liking,na.rm=TRUE)
  neg<-mean(data[data[,attr]==0,]$Liking,na.rm=TRUE)
  lift<-c(lift,pos-neg)
}
# 属性标签数据
text<-c()
for(i in lift){
  if(i>0){text_y<--0.5}
  else{text_y<-0.5}
  text<-c(text,text_y)
}
lift_df<-data.frame(labels=attr_all,lift=lift,text=text)
# 为了便于图形展示，将数据排序处理
index_sort<-sort(lift_df$lift,index.return=TRUE)$ix
lift_df$labels<-factor(lift_df$labels,levels=lift_df$labels[index_sort])
```

7.3.3 绘图

绘制基于 Lift 的正负条形图：

```
# 绘图
library(ggplot2)
library(scales)
# 设置配色
colour <- seq_gradient_pal("orange", "blue")(seq(0,1,length.out=16))

ggplot(data=lift_df,aes(x=labels,y=lift,fill=labels))+
```

```
  geom_bar(stat="identity")+
  geom_hline(yintercept=0,color="blue")+
  geom_text(aes(x=labels,y=text,label=labels))+
  scale_fill_manual(values=colour)+
  guides(fill="none")+  # 移除图例
  theme(axis.text.y=element_blank(),
        axis.ticks.y=element_blank())+
  labs(y="",x="")+
  coord_flip()
```



7.4 相关矩阵图

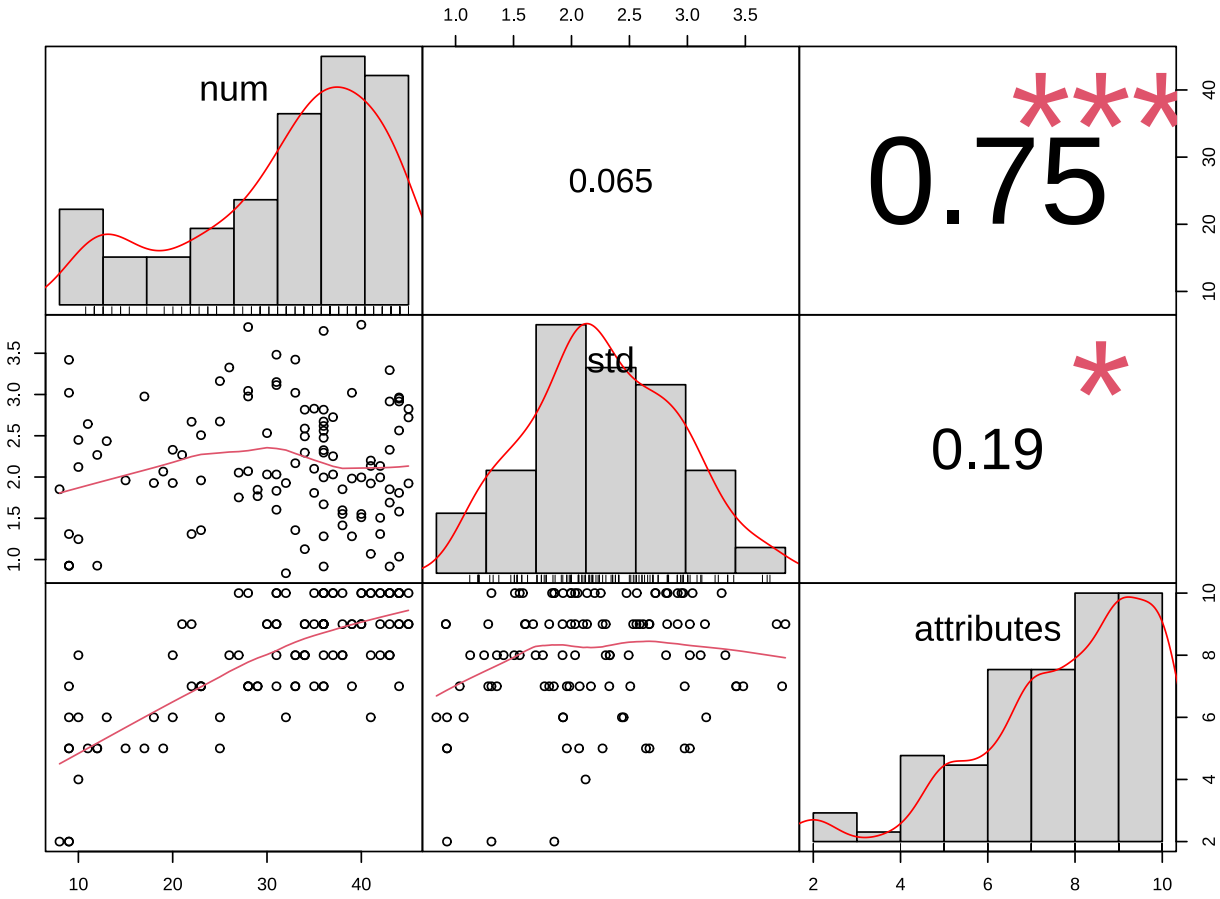
7.4.1 数据说明

我使用的是论文 ***How children approach a CATA test influences the outcome*** 的数据 `Bread.csv`，我将统计每一位评估者所有的打勾数量、不同产品之间打勾数量的标准差以及打勾属性的数量，最后绘制它们的相关矩阵图 (参考该论文的 **Fig.1**。

7.4.2 数据处理于绘图

注意绘制相关矩阵图要用到 R 包 `PerformanceAnalytics` 中的 `chart.Correlation` 函数，请提前安装好。

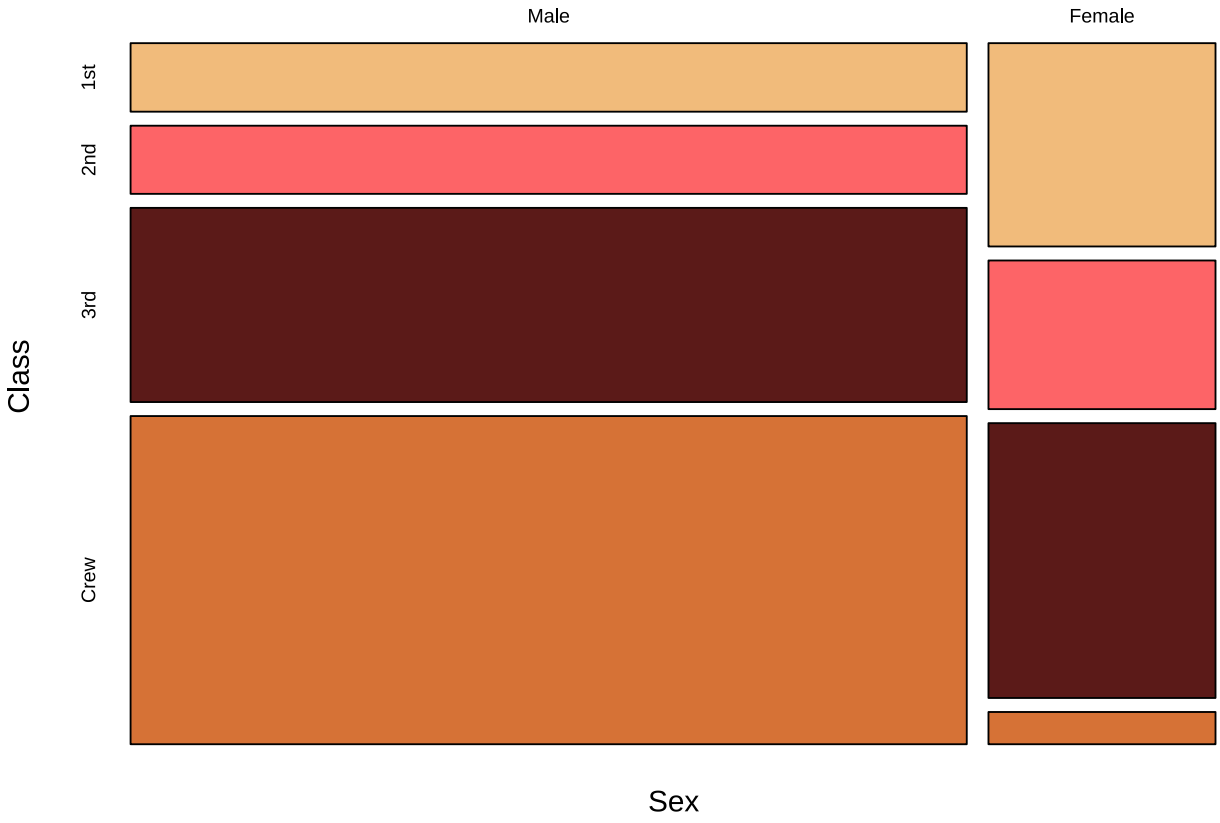
```r
# 导入数据
library(reshape2)
data<-read.csv("E:\\Bread.csv")

# 去除为空值的最后两列
data<-data[,-c(ncol(data),ncol(data)-1)]

# 将字符串格式转化为数值格式
data<-data.frame(lapply(data,as.numeric))

#number 变量
attr_data=melt(data[,c(1,9:18)],id=("No"))
num=aggregate(attr_data[,c(1,3)],by=list(attr_data$No),sum,na.rm=TRUE)$value

#std 变量
prd_data=melt(data[,c(1,4:18)],id=c("No","prd"))
temp=aggregate(prd_data[,c(1,2,4)],
            by=list(No_=prd_data$No,prd_=prd_data$prd),sum,na.rm=TRUE)
std=aggregate(temp[,c(1,5)],by=list(temp$No_),sd)$value

#attributes 变量
temp=aggregate(attr_data[,c(1,3)],
               by=list(No_=attr_data$No,prd_=attr_data$variable),
               sum,na.rm=TRUE)
attributes<-aggregate(temp[,c(1,4)],by=list(temp$No_),
                      function(x){return(sum(x>0))})$value
# 数据整合
matrix_data<-data.frame(num=num,std=std,attributes=attributes)

# 绘图
library(PerformanceAnalytics)
chart.Correlation(matrix_data, histogram=TRUE, pch=19)
```



7.5 马赛克图

马赛克图是分析两个分类变量之间最常用的图形。可以使用 R 包 `graphics` 中的 `mosaicplot()` 函数绘制马赛克图。R 包 `wesanderson` 中内置许多配色方案，主要用于图形配色。以下是我基 R 于内置 **Titanic** 数据集，绘制存活人员性别与等级两个分类变量之间的马赛克图：

```
library(graphics)
library(wesanderson) # 颜色提取
mosaicplot( ~Sex+Class,Titanic, color = wes_palette("GrandBudapest1"),main = '')
```


Male
Female
1st
2nd
3rd
Crew
Class
Sex

第二部分　Temporal Check-All-That-Apply(TCATA)

8　数据准备

8.1　原始数据格式说明

基于 R 语言的 TCATA 分析原始数据格式要求如下：

- 原始数据整体与老师发的 **TCATA 实例**一致
- 在变量命名上：
  - 评估者列命名必须为 **Assessors**
  - 产品列命名必须为 **Products**
  - 属性列命名必须为 **Attributes**
  - 评估时间列必须从 **0** 开始，且列名为**数字**
  - 即与 **TCATA 实例**命名完全一致
- 在列的顺序排布上：
  - **Assessors,Products,Attributes** 三列必须为前三列，三者的顺序可以任意
  - 关于评估时间的列必须在前三列之后，按时间从小到大排布即可
- 同样，需要将 xls 或其 Excel 文件格式导出为 **csv** 格式，在用 R 代码导入为数据框格式，再做进一步处理。

8.2　数据导入

数据导入流程与 CATA 完全一致，仅仅不同在于 TCATA 数据不用讨论是否存在理想点数据。关于导入方法，见第一部分的**数据导入**那一节。以下是适用我自己的导入数据代码：

```
# 导入原始数据
data<-read.csv("E:\\TCATA 实例.csv")
```

9　汇总表 (Summary table)

使用我的自定义函数 `Summary_TCATA_table`，返回数据的汇总结果。我将函数源代码保存在 R 文件 `Summary_TCATA_table.R` 中。执行以下代码，输出结果：

```
# 运行函数源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\TCATA\\Summary_TCATA_table.R")

# 调用该函数输出结果
Summary_TCATA_table(data=data)
```

	value
Number of sessions	1
Number of assessors	50

	value
Number of products	3
Number of attributes	6
Start time	0
End time	20

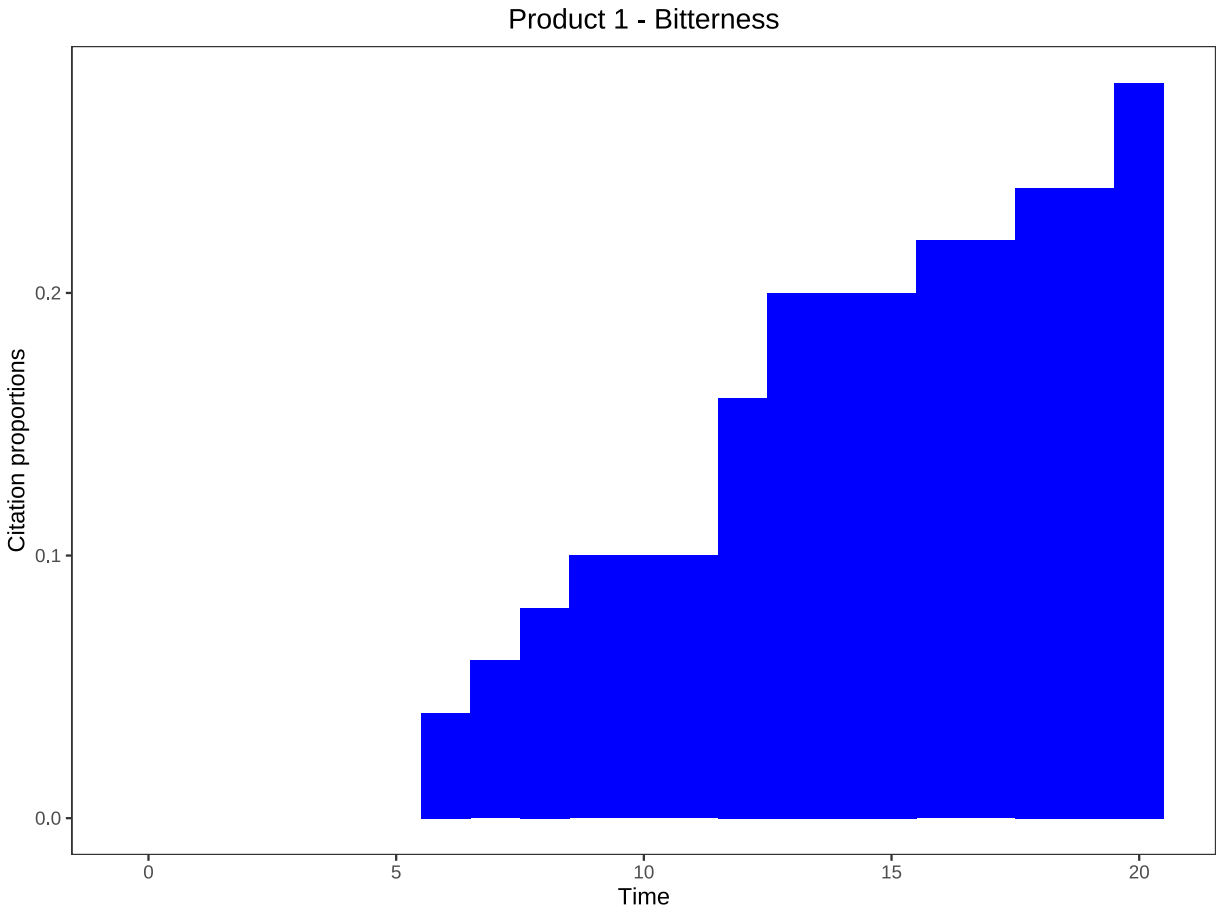
函数说明：该函数只有 data 一个参数，只需将所需要分析的数据传给该函数就可以返回汇总结果。

10 引用比例条形图 (Bar chart of citation proportions)

10.1 单图绘制

使用自定义函数 Barchart_prop 绘制产品与属性基于时间的比例条形图。该函数被保存在 R 文件 Barchart_prop.R 中。使用以下代码绘制：

```
# 运行函数源文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\TCATA\\Barchart_prop.R")

# 调用该函数输出结果
Barchart_prop(data=data,product="1",attr="Bitterness",index=4,color="blue")
```



函数说明：该函数有五个可修改参数：

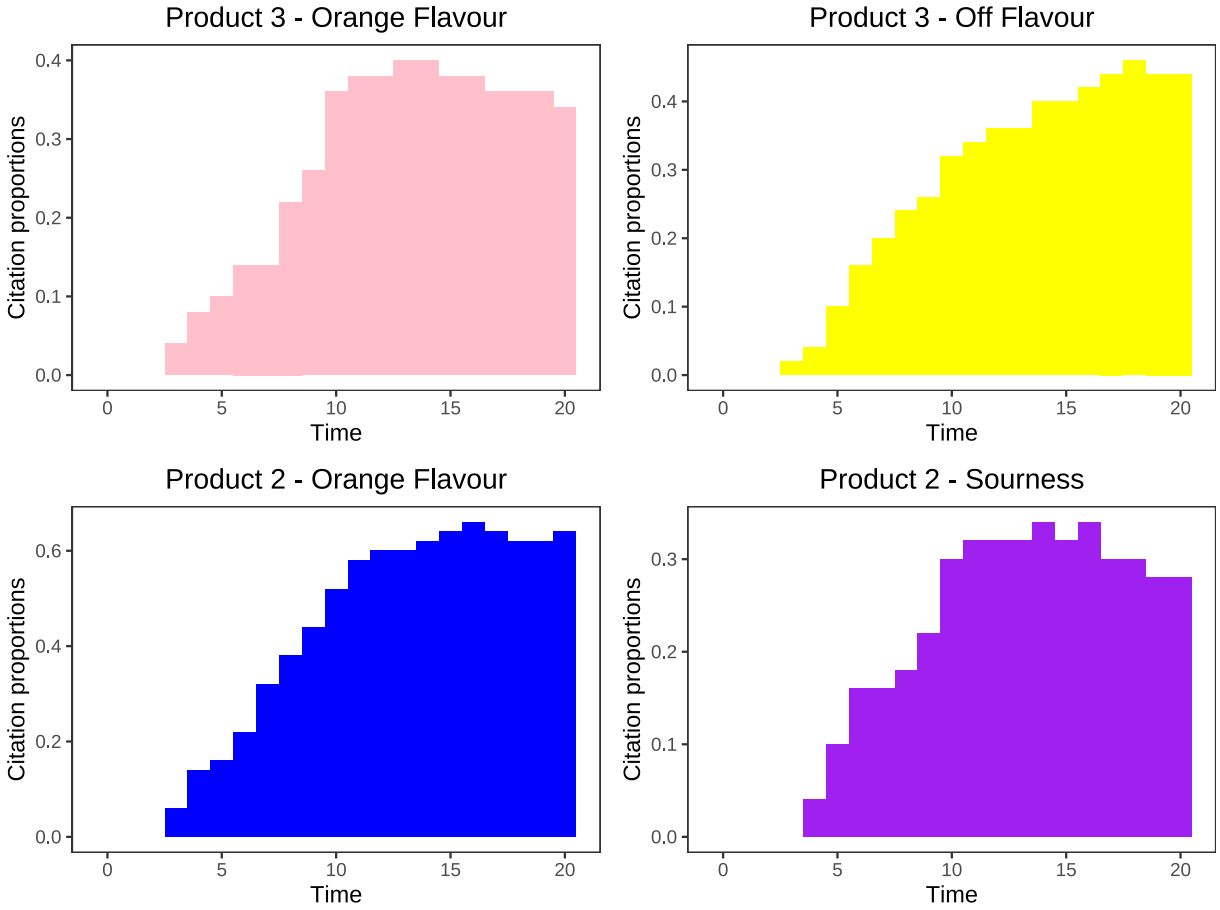
- **data**: 该参数为用于分析的 TCATA 数据;
- **index**: 该参数为 data 中评估时间为 **0** 的列索引;
- **product**: 该参数指定产品名称;
- **attr**: 该参数指定属性名称;
- **color**: 指定条形图绘图颜色，默认值为蓝色 (`blue`);

10.2 多图汇总

如果您需要一次性展示多图，可以执行以下代码。下列代码是通过多次调用上述绘图函数，然后利用 R 包 ggpubr 中的 `ggarrange` 函数将多图进行组合。下列代码以组合四张图为例 (2x2):

```
## 组合比例条形图
# 加载宏包，若该包未安装，请提前安装此包
library(ggpubr)

all_product<-unique(data$Products) # 获得所有产品名
all_attr<-unique(data$Attributes)  # 获得所有属性名
# 定义每张绘图所用颜色
all_color<-c("blue","green","pink","purple","yellow","grey")

# 多次调用绘图函数
```

```
p1<-Barchart_prop(data,all_product[3],all_attr[4],4,all_color[3])
p2<-Barchart_prop(data,all_product[3],all_attr[3],4,all_color[5])
p3<-Barchart_prop(data,all_product[2],all_attr[4],4,all_color[1])
p4<-Barchart_prop(data,all_product[2],all_attr[5],4,all_color[4])

# 调用组合函数
ggarrange(p1,p2,p3,p4,ncol=2,nrow=2)
```



组合函数说明：ggarrange 函数只需要设定两个参数:

- **ncol**: 指定组合图的列数;
- **nrow**: 指定组合图的行数;

然后其他的保存绘图函数的变量依次以逗号分隔放在上述两个参数之前即可。

11 产品曲线 (Curves by product)

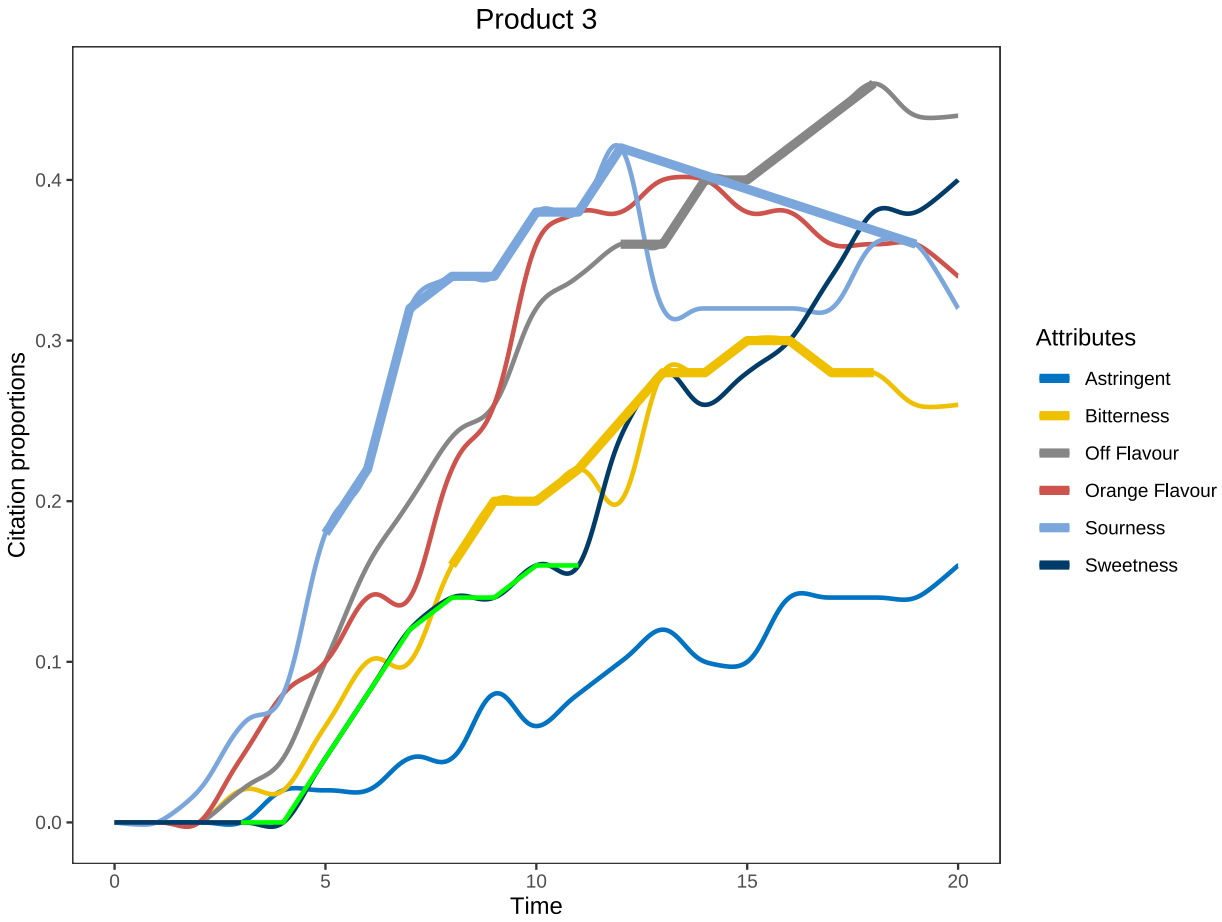
11.1 必读说明

需要说明的是，在 XLSTAT 所给示例中，产品曲线图中画了很多条曲线，曲线数量要多于属性数量。这是因为在曲线图中为了体现该产品的每个属性在每个时间点是否显著这个维度，XLSTAT 使用虚线，粗线等视觉体现来突出是否显著。虽然达到了体现显著性维度的目的，但是个人感觉视觉效果不是很好，曲线太多，会对视觉效果有一定的干扰。所以，我使用了新的规则去体现其想表达的显著性维度，具体内容，在下一小节会有说明。

11.2 曲线绘图

同样，自定义函数 `Curve_prop` 可以实现绘制曲线图，其函数源代码保存在 R 文件 `Curve_prop.R` 中。请看下面绘图示例：

```
# 运行源函数文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\TCATA\\Curve_prop.R")

# 调用函数绘图
# 该函数要使用到 R 包 ggplot2,ggalt 与 ggsci, 请提前安装好这三个包
Curve_prop(data=data,product="3",index=4)
```



函数说明：该函数有三个修改参数：

- **data**: 指定分析数据;
- **product**: 指定产品名称;
- **index**: 该参数为 data 中评估时间为 **0** 的列索引;

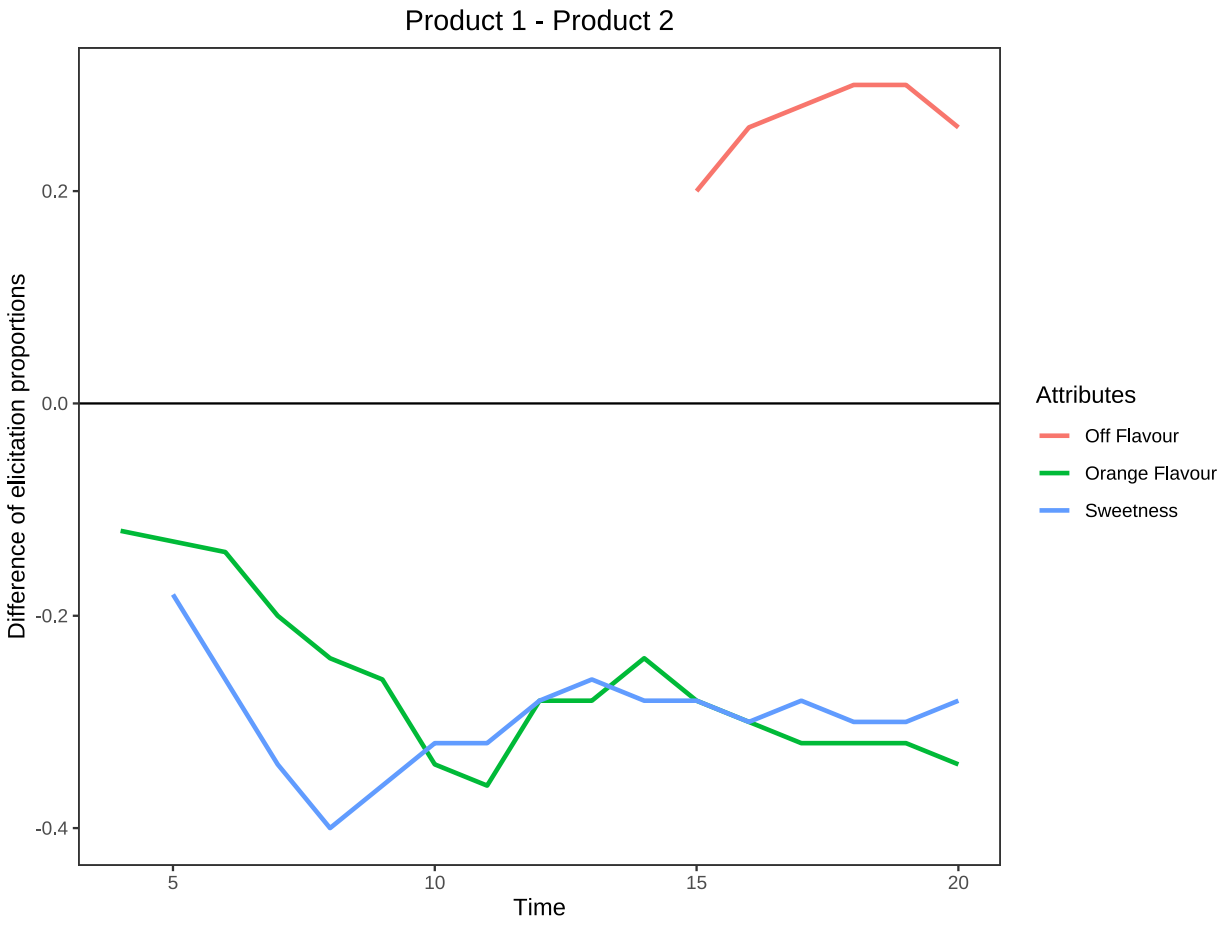
图形解读: 该函数有三种曲线:

- **细实线**: 代表该产品下该属性与所有产品下该属性的平均水平**无显著差异**;
- **绿色实线**: 代表该产品下该属性**显著低于**所有产品下该属性的平均水平;
- **粗实线**: 代表该产品下该属性**显著高于**所有产品下该属性的平均水平;

11.3 显著性差异 (Significative differences of citation proportions)

利用上一小节的显著性结果，我们可以绘制进行比较的两个产品下只有显著性差异的属性时间序列图。自定义函数 `Diff_prop` 可以实现此功能，其保存在 R 文件 `Diff_prop.R` 中。具体实现代码如下:

```
# 运行源函数文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\TCATA\\Diff_prop.R")

# 调用函数绘图
Diff_prop(product1="1",product2="2",data=data,index=4)
```



函数说明: 该函数有四个修改参数:

- **product1**: 指定进行比较的第一个产品名称;
- **product2**: 指定进行比较的第二个产品名称;
- **data**: 指定分析数据;
- **index**: 该参数为 data 中评估时间为 **0** 的列索引;

图形解读: 该折线图只显示两个产品中，基于某些时间点具有显著性差异的属性的折线图。不显著的属性以及不显著的时间点都不会显示在图中。同时，关于显著性检验我使用的是 **Wilcox 配对检验。**

12 对应分析 (Correspondence Analysis)

12.1 必读说明

TCATA 分析中的对应分析与 CATA 分析中的对应分析原理完全相同，但是由于两种方法所使用的数据有显著差别，所以 CATA 分析的 CA 分析 R 代码并不能适用于 TCATA 分析中。但是在使用 R 包 `factoextra` 中的可视化函数如 `fviz_ca_col()` 等函数时，绘图代码完全可以适用，仅仅是 ca 对象有所不同而已。

12.2 R 包的准备

我们需要用的三个 R 包:

- `FactoMineR`: 该包用于对应分析的统计计算
- `factoextra`: 该包用于对应分析可视化
- `reshape2`: 该包用于数据整合重构

12.3 数据整理

由于进行因子分析的数据不是原始数据，所以我们需要将分析数据整理出适合因子分析的数据。自定义函数 `Tidy_data` 可以对原始分析数据进行整理，返回出能够进行对应分析的数据对象。该函数保存在 R 文件 `Tidy_data.R` 中。见如下代码示例:

```
# 运行源函数文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\TCATA\\Tidy_data.R")

# 整理出数据
ca_data<-Tidy_data(data=data,index=4)
# 由于，列数太多，只显示前 10 列
ca_data[,1:10]
```

1_X0	1_X1	1_X2	1_X3	1_X4	1_X5	1_X6	1_X7	1_X8	1_X9
0	0	0	0.00	0.00	0.00	0.04	0.04	0.06	0.06
0	0	0	0.00	0.00	0.00	0.04	0.06	0.08	0.10
0	0	0	0.02	0.04	0.10	0.18	0.28	0.28	0.32
0	0	0	0.00	0.02	0.06	0.08	0.12	0.14	0.18

1_X0	1_X1	1_X2	1_X3	1_X4	1_X5	1_X6	1_X7	1_X8	1_X9
0	0	0	0.00	0.04	0.08	0.10	0.10	0.18	0.22
0	0	0	0.02	0.12	0.14	0.16	0.20	0.24	0.30

函数说明：该函数有两个修改参数：

- **data**: 指定分析数据；
- **index**: 该参数为 data 中评估时间为 **0** 的列索引；

12.4 特征值 (Eigenvalues)

使用 R 包 FactoMineR 进行对应分析，代码如下：

```
# 加载 R 包
library(FactoMineR)

# 返回对应分析对象
#CA 是专门进行对应分析的函数
ca<-CA(ca_data,graph=FALSE)

# 输出特征值数据，保留三位小数
round(ca$eig,3)
```

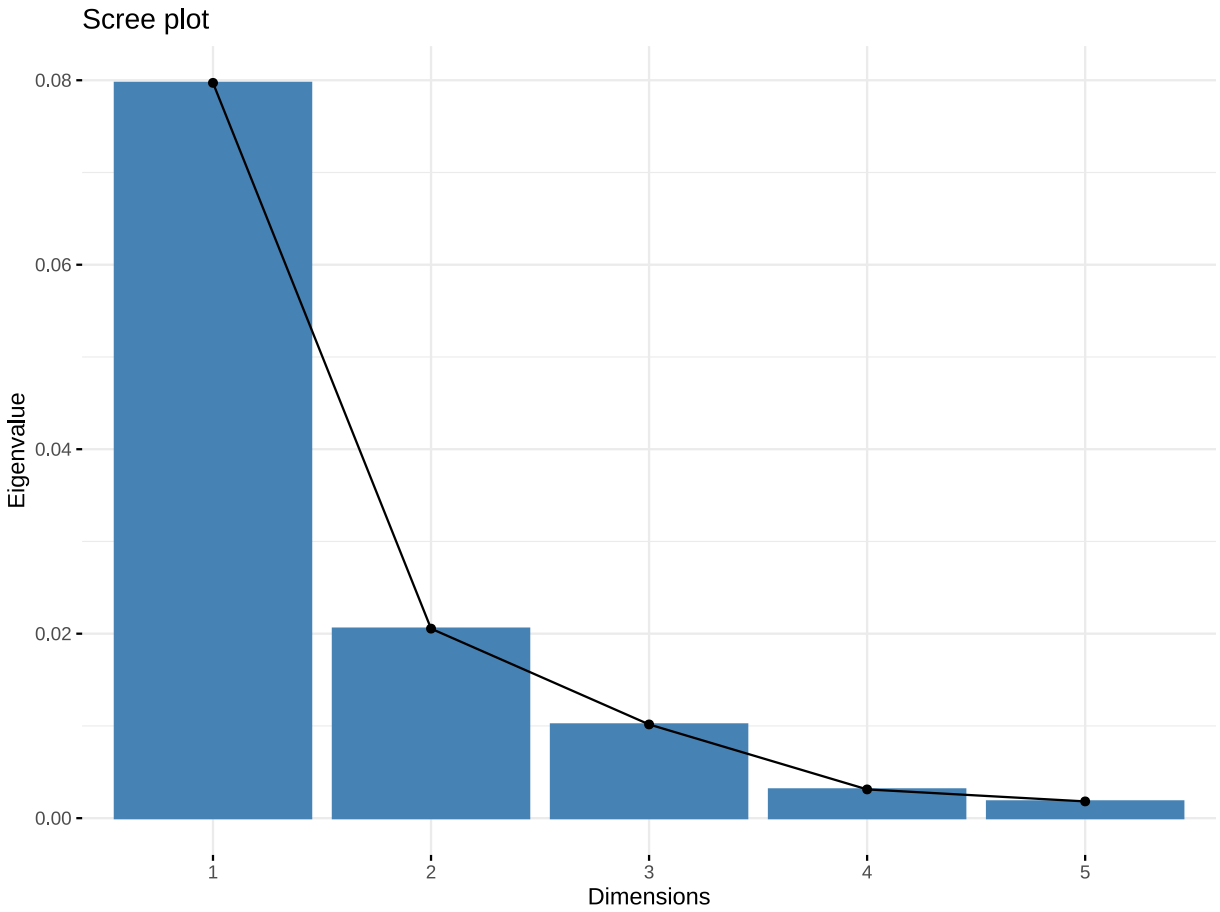
```
##        eigenvalue percentage of variance cumulative percentage of variance
## dim 1       0.080                 69.101                             69.101
## dim 2       0.021                 17.812                             86.913
## dim 3       0.010                  8.809                             95.722
## dim 4       0.003                  2.703                             98.424
## dim 5       0.002                  1.576                            100.000
```

12.5 碎石图 (Scree plot)

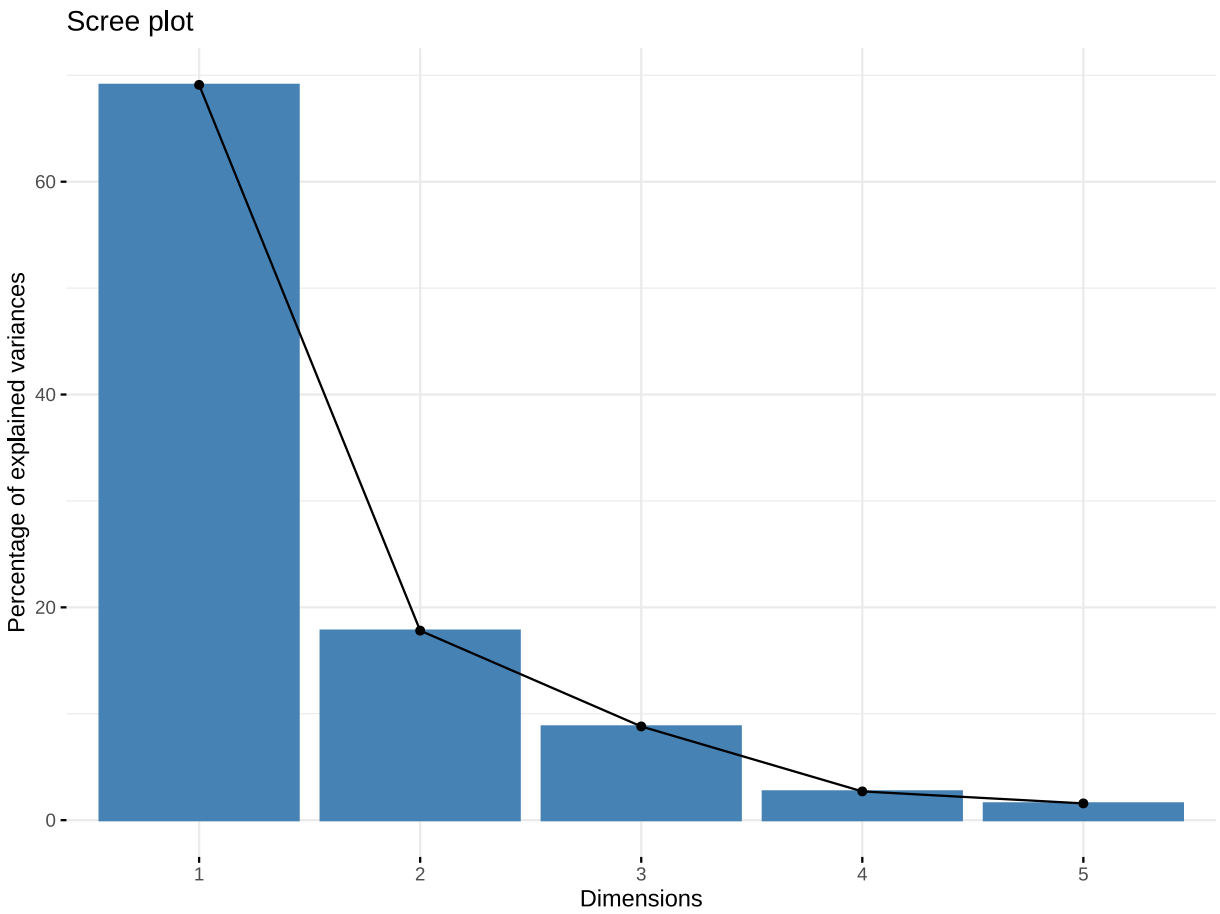
CATA 分析的绘制碎石图的两个方法，完全可以用于 TCATA 方法中，这是因为两个方法使用的都是 CA 函数返回的 ca 对象，具体函数使用方法请看 CATA 分析该部分。下面直接进行示例：

12.5.1 方法一

```
# 基于特征值的碎石图
fviz_eig(ca,choice="eigenvalue")
```

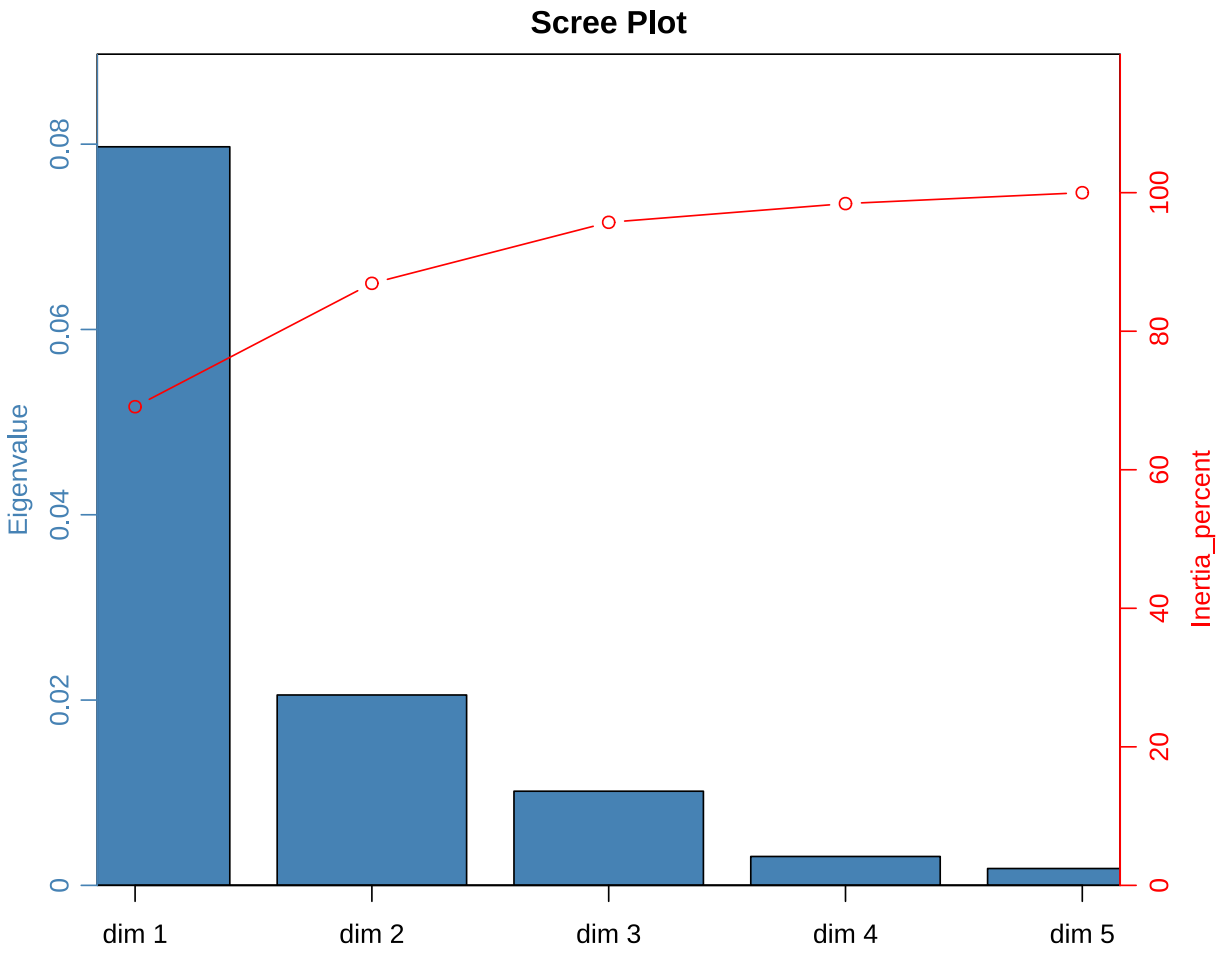



```
# 基于惯性百分比的碎石图
fviz_eig(ca,choice="variance")
```



12.5.2 方法二

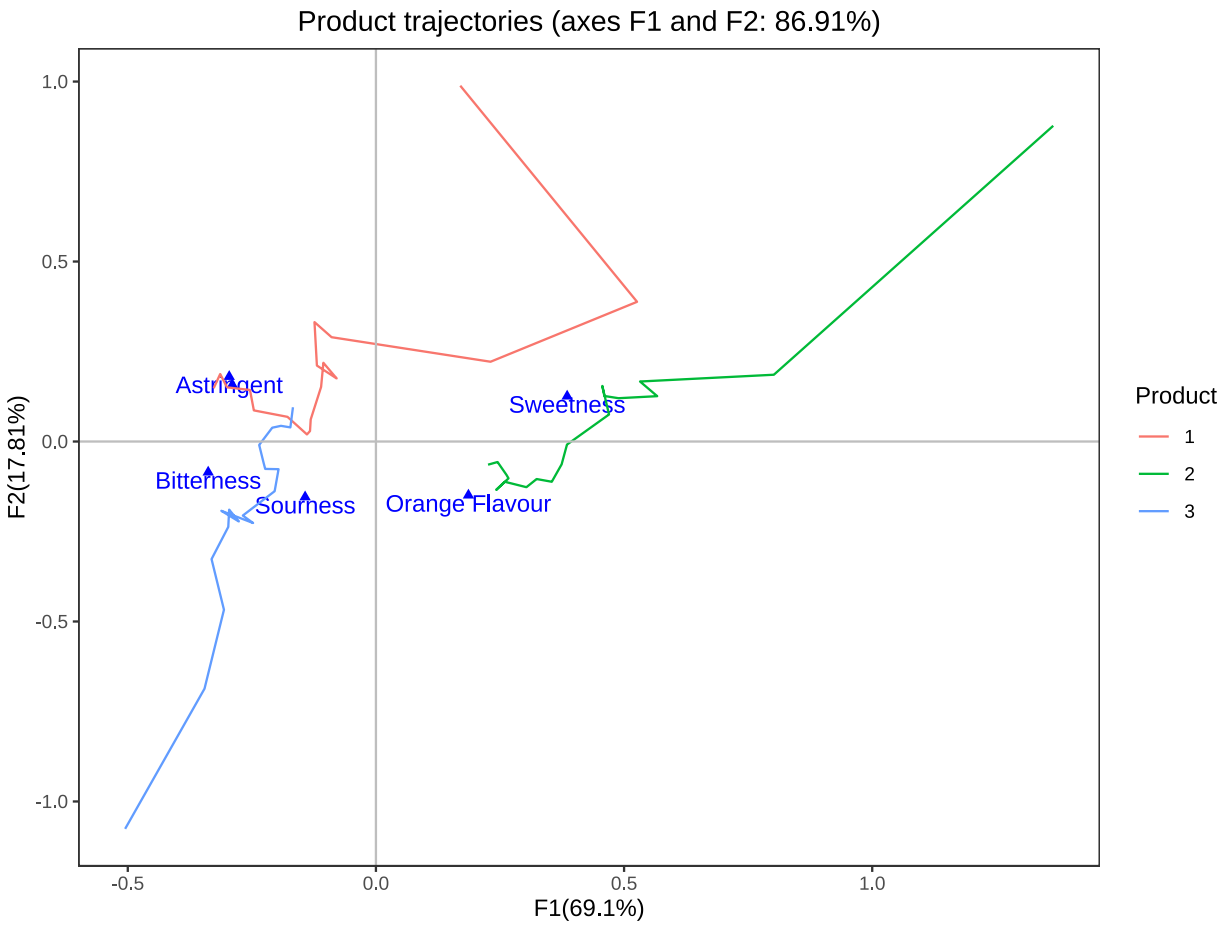
```
# 加载 R 包
library(plotrix)
# 绘图
# 该图可根据 CA 分析的 ca 变量直接运行得出，不需要修改其他参数
# 当然前提是 ca 变量存在
twoord.plot(lx=1:nrow(ca$eig),ly=ca$eig[,1],
            rx=1:nrow(ca$eig),ry=ca$eig[,3],
            type=c("bar","b"),xlab="axis",ylab="Eigenvalue",
            rylab="Inertia_percent",rylim=c(0,120),
            lylim=c(0,max(Eigenvalue=ca$eig[,1])+0.01),
            ylab.at=(max(Eigenvalue=ca$eig[,1])+0.01)/2,
            rylab.at=50,lcol="steelblue",rcol="red",main="Scree Plot",
            lytickpos=round(seq(0,max(Eigenvalue=ca$eig[,1]),length.out=5),2),
            rytickpos=seq(0,100,by=20),rpch=1,
            xticklab=rownames(ca$eig),
            mar=c(2,4,2,4),   # 调整边距
            do.first="plot_bg(col='white')")
```



12.6 产品轨迹图 (Product trajectories)

自定义函数 Ca_traj 实现产品轨迹图。R 源代码在 R 文件 Ca_traj 中。以下是代码示例:

```
# 运行源函数文件
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\TCATA\\Ca_traj.R")

# 输出产品轨迹图
Ca_traj(ca=ca,dim=c(1,2),data=data,color="blue")
```



函数说明：该函数有四个可修改参数：

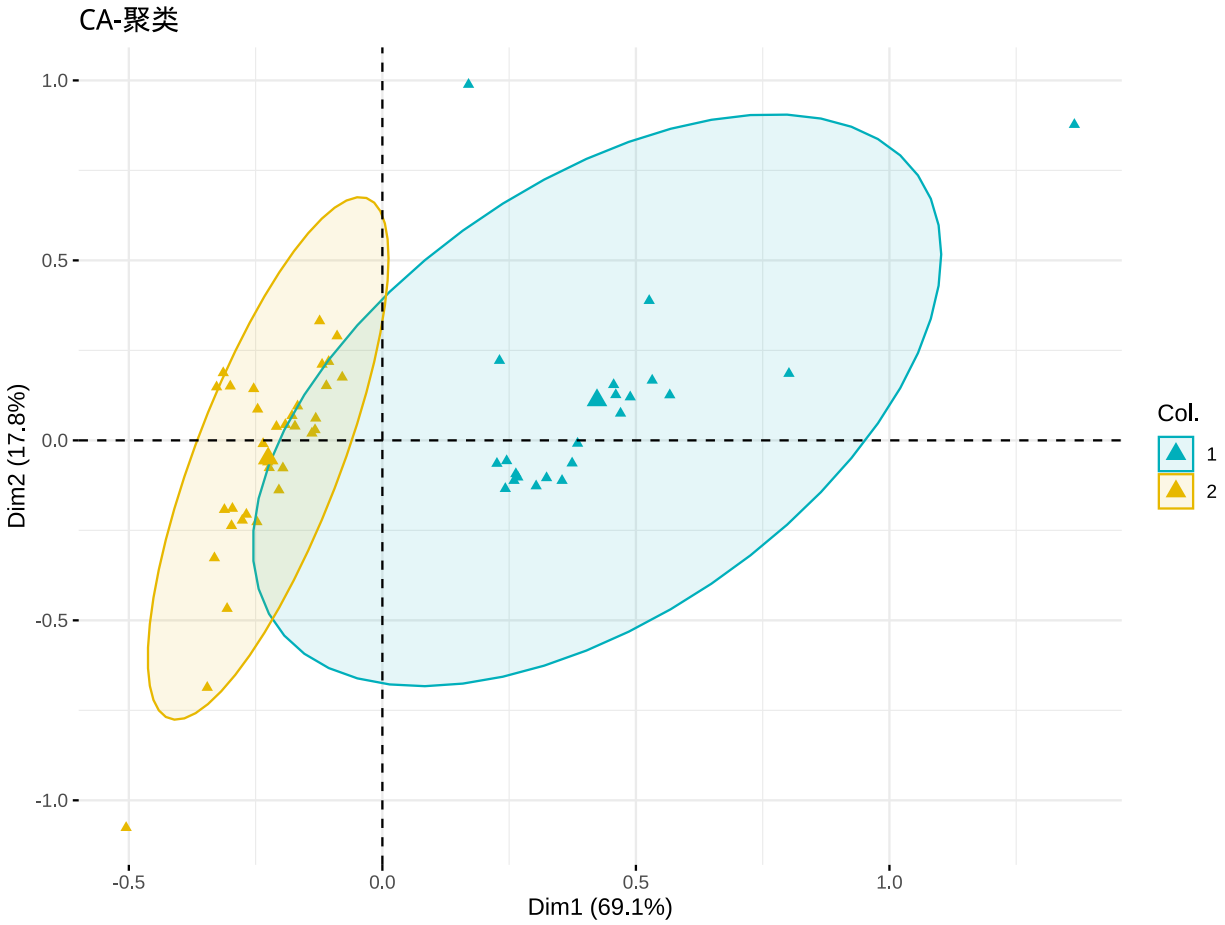
- **ca**: 指定 ca 对象，即 `CA` 函数的返回值；
- **dim**: 指定产品轨迹的维度，默认值为 `c(1,2)`，即第一维度和第二维度；
- **data**: 指定分析数据；
- **color**: 指定散点与文本标签颜色，默认值为红色 `red`；

12.7 R 包 factoextra 可视化简单示例

如 CATA 分析一样，TCATA 分析同样可以适用该 R 包可视化。下面是一个例子。

```
# 随机数种子
# 由于 kmeans 聚类利用了随机数
# 为了保持每次运行代码聚类结果相同，需要设定随机数种子
set.seed(2)

# 进行 kmean 聚类
#ca$col$coord 是属性的两个维度的坐标
#centers 是要聚成几类
c<-kmeans(ca$col$coord,centers=2)

# 可视化
fviz_ca_col(ca,
```

```
        # 依据每个列变量（属性）的聚类结果填充颜色
        col.col = as.factor(c$cluster),
        # 只显示散点
        geom.col="point",
        # 引号内部是十六进制颜色码，以 # 号开头，该参数表示按照这三种颜色渐进映射
        palette = c("#00AFBB", "#E7B800"),
        # 在每一类上添加椭圆
        addEllipses = TRUE,
        # 修改标题
        title="CA-聚类"
              )
```



13 文献分析方法补充

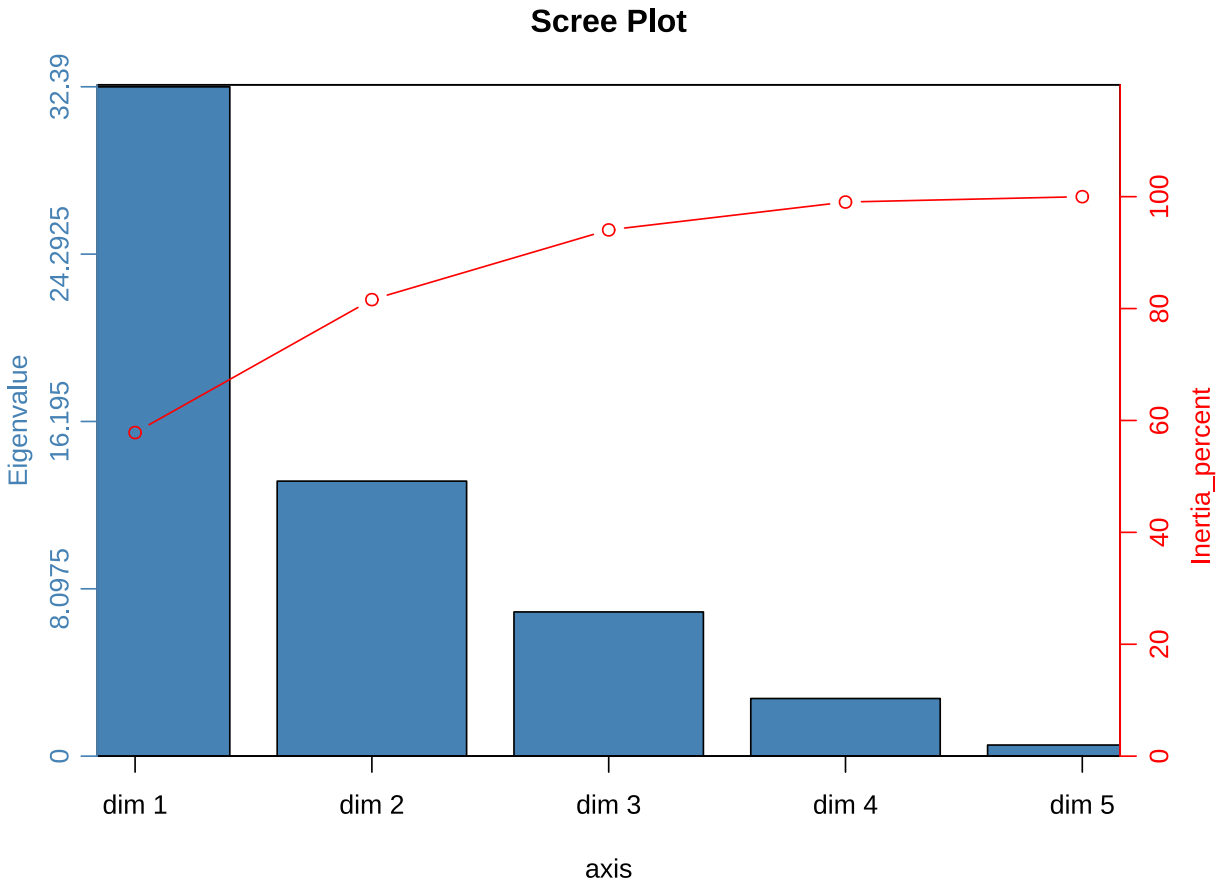
13.1 主成分分析 (PCA)

主成分分析依旧要使用到 R 包 FactoMineR 与 factoextra。与对应分析类似，除了使用的是 PCA 函数，而不是 CA 函数，其他流程完全相同。

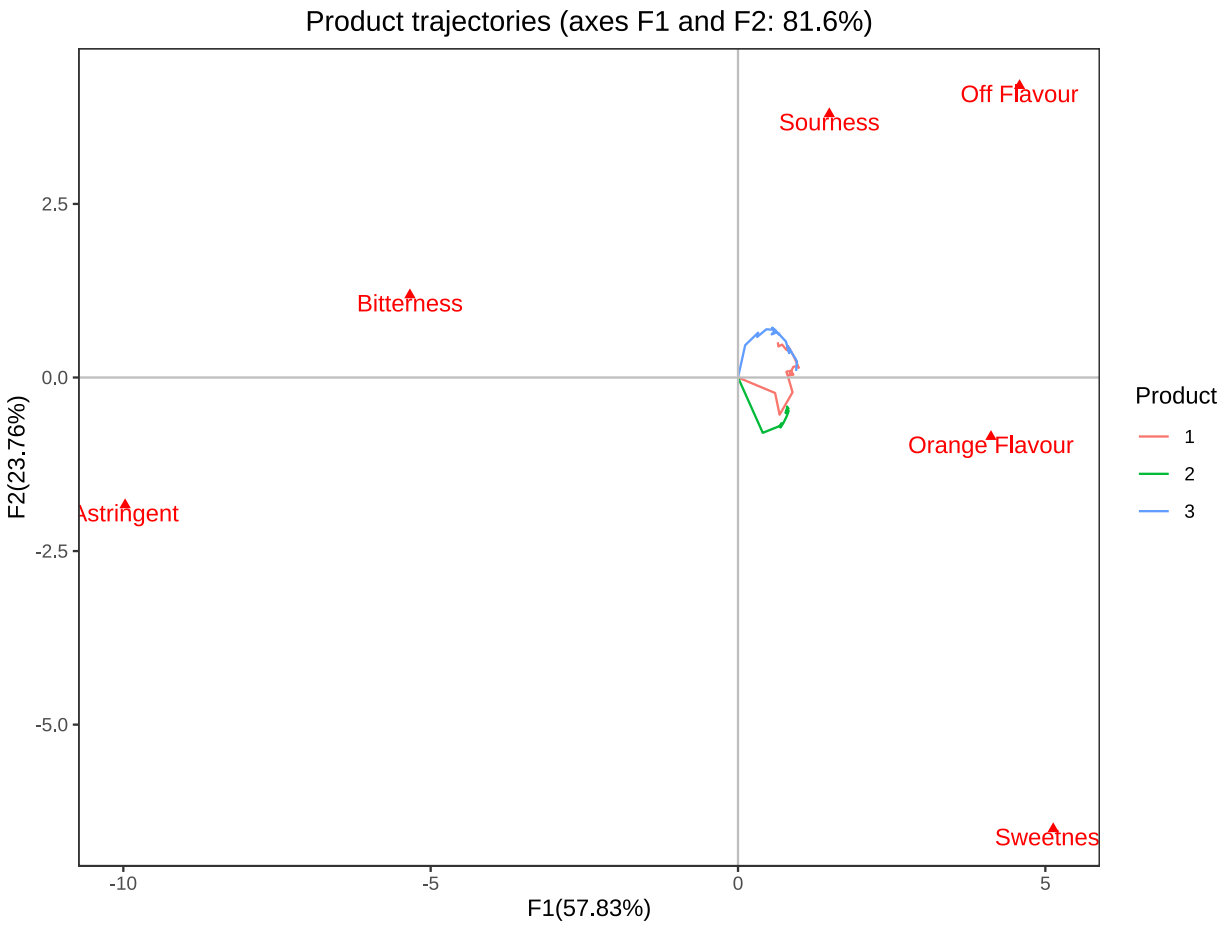
```
# 由于前面已经运行过函数源文件，所以下面的自定义函数可以不用 source 运行
# 数据整理
pca_data<-Tidy_data(data,index=4)
#FactoMineR 进行主成分分析的函数为 PCA()
pca<-PCA(pca_data,graph=FALSE)
# 特征值
round(pca$eig,3)
```

```
##        eigenvalue percentage of variance cumulative percentage of variance
## comp 1     32.387                 57.833                             57.833
## comp 2     13.308                 23.765                             81.598
## comp 3      6.978                 12.461                             94.059
## comp 4      2.789                  4.980                             99.039
## comp 5      0.538                  0.961                            100.000
```

```
# 碎石图
library(plotrix)
twoord.plot(lx=1:nrow(pca$eig),ly=pca$eig[,1],
            rx=1:nrow(pca$eig),ry=pca$eig[,3],
            type=c("bar","b"),xlab="axis",ylab="Eigenvalue",
            rylab="Inertia_percent",rylim=c(0,120),
            lylim=c(0,max(Eigenvalue=pca$eig[,1])+0.1),
            ylab.at=(max(Eigenvalue=pca$eig[,1])+0.1)/2,
            rylab.at=50,lcol="steelblue",rcol="red",main="Scree Plot",
            lytickpos=seq(0,round(max(Eigenvalue=pca$eig[,1]),2),
                          length.out=5),
            rytickpos=seq(0,100,by=20),rpch=1,
            xticklab=rownames(ca$eig),
            do.first="plot_bg(col='white')")
```

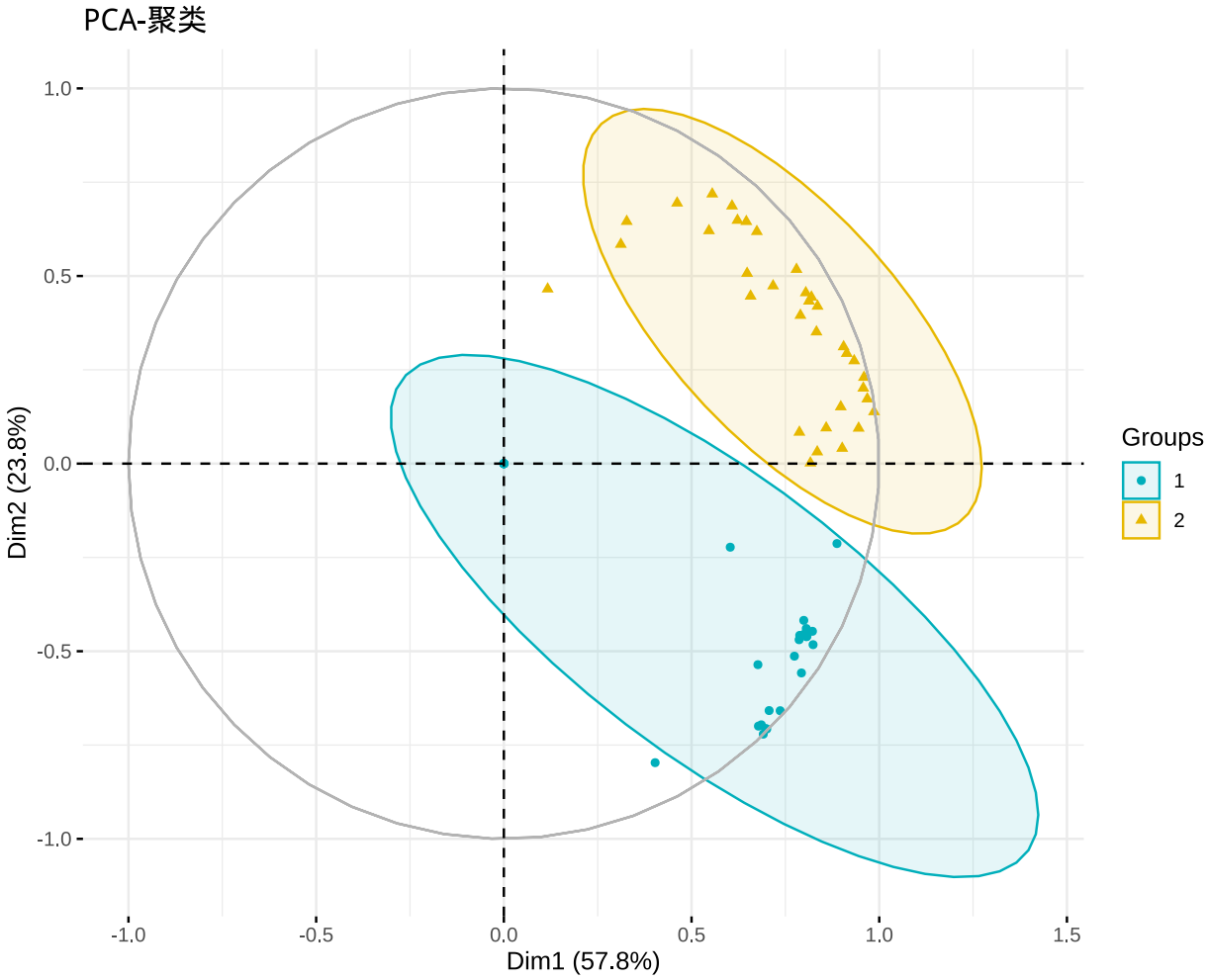



```
# 产品轨迹图
source("C:\\Users\\里高房价\\Desktop\\CATA&TCATA\\TCATA\\Pca_traj.R")
Pca_traj(pca=pca,dim=c(1,2),data=data)
```



R 包 factoextra 进行主成分分析 (PCA) 可视化的函数为 fviz_pca_biplot,fviz_pca_var,fviz_pca_ind, 其参数基本相同，限于篇幅，想要了解更详尽的函数用法，可以执行命令 “?+ 函数名”, 获取函数帮助。下面以聚类为例：

```
# 聚类示例
set.seed(3)
c<-kmeans(pca$var$coord,centers=2)
# 可视化
fviz_pca_var(pca,
      # 依据每个列变量 (属性) 的聚类结果填充颜色
      col.var = as.factor(c$cluster),
      # 只显示散点
      geom.var="point",
      # 用类别映射散点形状
      habillage = as.factor(c$cluster),
      # 引号内部是十六进制颜色码，以 # 号开头，该参数表示按照这三种颜色渐进映射
      palette = c("#00AFBB", "#E7B800"),
      # 在每一类上添加椭圆
      addEllipses = TRUE,
      # 修改标题
      title="PCA-聚类"
             )
```



13.2 多重因子分析 (MFA)

与对应分析、主成分分析类似，只要将上面 `CA()` 或 `PCA()` 函数替换为 `MFA()` 函数即可。在 R 包 `factoextra` 中其对应的可视化函数为 `fviz_mfa_quali_biplot()`,`fviz_mfa_ind()`,`fviz_mfa_var()`，其参数与上述方法几乎完全一致。