

R 机器学习

第 17 讲 关联规则

张敬信

2022 年 10 月 27 日

哈尔滨商业大学

“尿布与啤酒”的故事是营销界的神话，是数据挖掘的经典案例。



发掘关联规则就是找到频繁项集，是一种经典的数据挖掘算法，广泛应用于：

- **购物篮分析**：识别购物篮中的关联商品，即哪些商品总是一起被购买；零售商据此可以了解顾客的购物习惯，进行有针对性的营销或货物摆放。
- **客户流失分析和市场营销**：发现客户可能流入或流出的人口学特征和行为，识别可能接受服务或购买产品的客户群。
- **信用卡风险分析**：识别可能会信用卡或抵押违约的客户特征，帮助银行在做信用卡或抵押业务时降低风险。
- **股票市场分析**：识别个体股票之间或股票与经济因素之间的关系，帮助股票交易者选择感兴趣的股票或改进交易策略。
- **医疗诊断**：识别症状、检查结果和疾病之间的关系，协助医生做疾病诊断甚至是治疗。

一. 关联规则相关概念

项集是项的集合。包含 k 个项的项集称为 k 项集，如集合

$\{\text{牛奶, 麦片, 糖}\}$

是一个 3 项集。

项集的出现频率是所有包含项集的事务计数，也称为绝对支持度或支持度计数。如果项集 I 的相对支持度满足预定义的最小支持度阈值，则 I 是频繁项集。 k 项频繁集通常记作 L_k 。

频繁项集具有向下包含性，即所有非空子集也是频繁项集；非频繁项集 I 添加事务 A 的新项集 $I \cup A$ 一定是非频繁项集。

关联规则表现为项集之间的关联性或相关性规则，记为 $A \Rightarrow B$ ，其中 A 和 B 为互不相交的项集或属性值对，关联规则意味着这些数据元组若包含规则的左边项也很可能包含右边项。例如，

面包 \Rightarrow 黄油

计算机 \Rightarrow 软件

年龄在 $[20, 29]$ 收入在 $[6K, 100K] \Rightarrow$ 购买最时新手机

最常用的关联规则的评估指标有支持度、置信度、提升度（还有 20 多种其它指标）。

- **支持度**：项集 A 和 B 都出现的概率，即几个关联的数据在数据集中出现的次数占总数据集的占比，一般来说，支持度高的数据不一定构成频繁项集，但是支持度太低的数据肯定不构成频繁项集

$$\text{Support}(A \Rightarrow B) = P(A \cap B) = \frac{A, B \text{ 同时发生的事务个数}}{\text{所有事务个数}}$$

其中， $P(A)$ 为包含 A 的项集的占比或概率。

- **置信度**：项集 A 出现的条件下，项集 B 出现的概率。反映的是一个数据出现后，另一个数据出现的概率

$$\begin{aligned}\text{Confidence}(A \Rightarrow B) &= P(B|A) = \frac{\text{Support}(A \Rightarrow B)}{P(A)} \\ &= \frac{P(A \cup B)}{P(A)} = \frac{A, B \text{ 同时发生的事务个数}}{A \text{ 发生的事务个数}}\end{aligned}$$

- **提升度**: 反映 A 和 B 之间的关联关系, 提升度 > 1 则 $B \Rightarrow A$ 是有效的强关联规则; 提升度 ≤ 1 则 $B \Rightarrow A$ 是无效的强关联规则

$$\begin{aligned}\text{Lift}(A \Rightarrow B) &= \frac{\text{Confidence}(A \Rightarrow B)}{P(B)} = \frac{P(A \cup B)}{P(A)P(B)} \\ &= \frac{P(A \cup B)}{P(A)} = \frac{n \cdot A, B \text{ 同时发生的事务个数}}{A \text{ 发生的事务个数} \cdot B \text{ 发生的事务个数}}\end{aligned}$$

特别地, 若 A 与 B 相互独立, 则 $\text{Lift}(B \Rightarrow A) = 1$, 因为此时 $P(X|Y) = P(X)$.

例现有 100 名学生，其中 10 名知道数据挖掘，8 名知道 R 语言，6 名二者都知道。考察关联规则：R 语言 \Rightarrow 数据挖掘

$$\text{支持度} = P(\text{R 语言} \& \text{数据挖掘}) = 6/100 = 0.06$$

$$\text{置信度} = \frac{\text{支持度}}{P(\text{R 语言})} = \frac{0.06}{0.08} = 0.75$$

$$\text{提升度} = \frac{\text{置信度}}{P(\text{数据挖掘})} = \frac{0.75}{0.10} = 7.5$$

说明：提升度 $= 7.5 > 1$ 表明对知道 R 语言的学生“推荐”数据挖掘，其“购买”数据挖掘的概率是给随机学生“推荐”的 7.5 倍，因此该规则是有价值的。

挖掘关联规则主要分为两步：

- (1) 找到所有支持度大于等于最小支持度的频繁项集；
- (2) 利用频繁项集，生成置信度大于等于最小置信度的关联规则。

第二步很简单，关键是第一步生成频繁项集需要大量计算。所有可能的项集数为 $2^n - 1$ ，其中 n 是唯一项数目。

最小支持度是用户或专家定义的衡量支持度的一个阈值，表示项目集在统计意义上的最低重要性；最小置信度是用户或专家定义的衡量置信度的一个阈值，表示关联规则的最低可靠性。同时满足最小支持度阈值和最小置信度阈值的规则称作强规则。

常用算法：Apriori, ECLAT, FP-Growth.

二. Apriori 算法

1. 算法原理

Apriori 算法 (Agrawal & Srikant, 1994) 是一种计算交易记录 (transaction) 来发现频繁项集, 然后从这些频繁项集中导出关联规则的逐层 (level-wise)、广度优先 (breadth-first) 算法, 是最经典的挖掘关联规则频繁项集的算法, 第一次实现了在大数据集上可行的关联规则提取, 其核心是基于两阶段频繁项集思想的递推算法: 通过连接产生候选项及其支持度, 然后通过剪枝生成频繁项集。

使用支持度来作为判断频繁项集的标准。Apriori 算法的目标是找到最大的频繁 k 项集。这里有两层意思, 首先, 要找到符合支持度标准的频繁集。但这样的频繁集可能有很多。其次, 要找到最大个数的频繁集。比如找到符合支持度的频繁集 $\{A, B\}$ 和 $\{A, B, E\}$, 则需要抛弃 $\{A, B\}$, 只保留 $\{A, B, E\}$, 因为前者是 2 项频繁集, 而后者是 3 项频繁集。

具体来说, Apriori 算法是采用迭代方法:

第 1 步. 找出所有频繁项集

先搜索出候选 1 项集及对应的支持度, 剪枝去掉低于支持度阈值的 1 项集, 得到频繁 1 项集。然后对剩下的频繁 1 项集进行连接, 得到候选的频繁 2 项集, 筛选去掉低于支持度阈值的候选频繁 2 项集, 得到真正的频繁 2 项集, 以此类推, 迭代下去, 直到无法找到频繁 $k + 1$ 项集为止, 对应的频繁 k 项集的集合即为算法的输出结果。

其中, 第 i 次的迭代过程包括三步: 扫描计算候选频繁 i 项集的支持度, 剪枝得到真正频繁 i 项集, 连接生成候选频繁 $i + 1$ 项集。

(1) **连接步**: 找到 k 项集

- 对给定的最小支持度阈值, 分别对 1 项候选集 C_1 , 剔除小于该阈值的项集得到 1 项频繁集 L_1 ;
- 由自身连接产生 2 项候选集 C_2 , 保留其中满足约束条件的项集得到 2 项频繁集, 记为 L_2 ;
- 由 L_2 与 L_1 连接产生 3 项候选集 C_3 , 保留其中满足约束条件的项集得到 3 项频繁集, 记为 L_3 ;

这样循环下去, 得到最大频繁项集 L_k .

(2) **剪枝步**：紧接着连接步，在产生候选项 C_k 的过程中起到减小搜索空间的目的。由于 C_k 是 L_{k-1} 与 L_1 连接产生的，根据 Apriori 的性质，频繁项集的所有非空子集也必须是频繁项集，所以不满足该性质的项集将不会存在于 C_k ，该过程就是剪枝。

第 2 步. 由频繁项集产生强关联规则

由上一步知未超过预定的最小支持度阈值的项集已被剔除，若剩下这些规则又满足了预定的最小置信度阈值，那么就挖掘出了强关联规则。

2. 简单算例

例 1 现有餐饮店 10 个订单的点餐数据：

```
library(tidyverse)
df = read_csv("datas/menu_orders.csv")
head(df)
```

#> # A tibble: 6 x 4

#>	订单号	时间	菜品 id	菜品名称
#>	<dbl>	<date>	<dbl>	<chr>
#> 1	101	2014-08-21	18491	健康麦烧包
#> 2	101	2014-08-21	8693	香煎葱油饼
#> 3	101	2014-08-21	8705	翡翠蒸香茜饺
#> 4	102	2014-08-21	8842	菜心粒咸骨粥
#> 5	102	2014-08-21	7794	养颜红枣糕
#> 6	103	2014-08-21	8842	菜心粒咸骨粥

首先，将上表中的事务数据（一种特殊类型的记录数据）整理成关联规则模型所需的数据结构。为了方便，将菜品 {18491, 8842, 8693, 7794, 8705} 分别记为 $\{a, b, c, d, e\}$:

```
df = df %>%  
  select(-时间, -菜品名称) %>% # 移除无用的时间和菜品名称列  
  mutate(菜品id = factor(菜品 id,  
                           levels = c(18491, 8842, 8693, 7794,  
                           labels = letters[1:5])))  
  
head(df, 4)  
#> # A tibble: 4 x 2  
#>   订单号 菜品 id  
#>   <dbl> <fct>  
#> 1    101 a  
#> 2    101 c  
#> 3    101 e  
#> 4    102 b
```



```
df %>%                                # 转化为宽表格式看一看
  group_by(订单号) %>%
  summarise(菜品id = str_c(菜品 id, collapse = ",")) # 宽表

#> # A tibble: 10 x 2
#>   订单号 菜品 id
#>   <dbl> <chr>
#> 1    101 a,c,e
#> 2    102 b,d
#> 3    103 b,c
#> 4    104 a,b,c,d
#> 5    105 a,b
#> 6    106 b,c
#> 7    107 a,b
#> 8    108 a,b,c,e
#> 9    109 a,b,c
#> 10   110 a,c,e
```

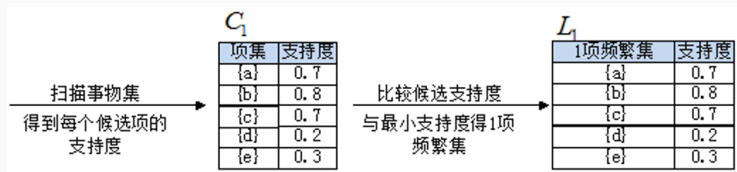
设支持度为 0.2, 算法过程如下:

第 1 步. 找出最大 k 项频繁集

- (1) 扫描所有事物, 事物中的每一项都是候选 1 项集 C_1 的成员, 计算每一项的支持度。比如

$$P(\{a\}) = \frac{\text{项}\{a\}\text{的支持度计数}}{\text{所有事务个数}} = \frac{7}{10} = 0.7$$

- (2) 将 C_1 中各项集的支持度与预先设定的最小支持度阈值做比较, 保留大于等于该阈值的项, 得到 1 项频繁集 L_1

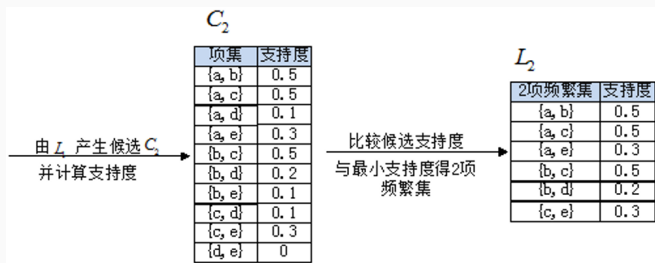


(3) 扫描所有事物, L_1 与 L_1 连接得到候选 2 项集 C_2 , 并计算每一项的支持度。比如,

$$P(\{a, b\}) = \frac{\text{项 } \{a, b\} \text{ 的支持度计数}}{\text{所有事务个数}} = \frac{5}{10} = 0.5$$

执行剪枝步, 由于 C_2 的每个子集 (即 L_1) 都是频繁集, 故没有项集从 C_2 中剔除;

(4) 将 C_2 中各项集的支持度与预先设定的最小支持度阈值作比较，保留大于等于该阈值的项，得到 2 项频繁集 L_2 ;

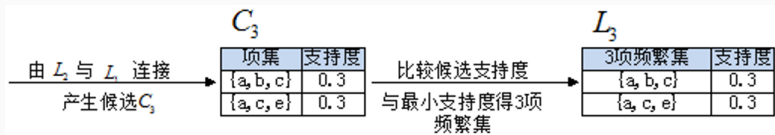


(5) 扫描所有事务, L_2 与 L_1 连接得到候选 3 项集 C_3 , 并计算每一项的支持度。比如,

$$P(\{a, b, c\}) = \frac{\text{项 } \{a, b, c\} \text{ 的支持度计数}}{\text{所有事务个数}} = \frac{3}{10} = 0.3$$

执行剪枝步, L_2 与 L_1 连接的所有非空子集也必须是频繁集, 因为 $\{b, d\}$, $\{b, e\}$, $\{c, d\}$ 不包含在频繁集 L_2 中, 即不是频繁集, 应剔除, 最后 C_3 中的项集只有 $\{a, b, c\}$ 和 $\{a, c, e\}$;

(6) 将 C_3 中各项集的支持度与预先设定的最小支持度阈值作比较, 保留大于等于该阈值的项, 得到 3 项频繁集 L_3 ;



(7) L_3 与 L_1 连接得到候选 4 项集，易知剪枝后为空集。最后得到最大 3 项频繁集 $\{a, b, c\}$ 和 $\{a, c, e\}$;

综上， L_1, L_2, L_3 都是频繁项集， L_3 是最大频繁项集。

三. Apriori 算法的实现

用 arules 包中的 apriori() 函数实现, 基本格式为:

```
apriori(data, parameter, appearance, control)
```

其中,

- data: 为交易类 (transactions) 数据框或二元关联矩阵或其稀疏矩阵形式;
- parameter: 用列表对象设置参数 supp (最小支持度, 默认 0.1)、conf (最小置信度, 默认 0.8)、maxlen (关联规则的最大项数 k , 默认 10);
- appearance: 用列表对象对规则的左端、右端等设置约束;
- control: 用列表对象设置对项排序、输出过程等。

```
library(arules)
```

transactions 数据

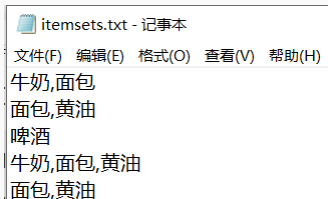
`apriori()` 接受的是 `transactions` 数据:

- 用 `read.transactions()` 直接从存放项集的 `txt` 读取
- 用 `transactions()` 函数从二值邻接矩阵、项集列表等转化

以如下的 `dat` 数据集为例,

交易 ID	项集
1	牛奶, 面包
2	面包, 黄油
3	啤酒
4	牛奶, 面包, 黄油
5	面包, 黄油

(1) 项集 txt



```
itemsets.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
牛奶,面包
面包,黄油
啤酒
牛奶,面包,黄油
面包,黄油
```

```
read.transactions("datas/itemsets.txt", format = "basket",
                  sep = ",", encoding = "UTF-8") %>%
```

```
inspect()
```

```
#>      items
```

```
#> [1] {面包, 牛奶}
```

```
#> [2] {黄油, 面包}
```

```
#> [3] {啤酒}
```

```
#> [4] {黄油, 面包, 牛奶}
```

```
#> [5] {黄油, 面包}
```

(2) 二值关联矩阵

```
dat1 = dat %>%  
  separate_rows(项集, sep = ",")  
dat1 %>%  
  mutate(n = 1) %>%  
  pivot_wider(names_from = 项集, values_from = n,  
              values_fill = 0)
```

#> # A tibble: 5 x 5

#>	交易 ID	牛奶	面包	黄油	啤酒
#>	<int>	<dbl>	<dbl>	<dbl>	<dbl>
#> 1	1	1	1	0	0
#> 2	2	0	1	1	0
#> 3	3	0	0	0	1
#> 4	4	1	1	1	0
#> 5	5	0	1	1	0

(3) 项集列表

```
split(dat1$项集, dat1$交易 ID)
```

```
#> $`1`
```

```
#> [1] " 牛奶" " 面包"
```

```
#>
```

```
#> $`2`
```

```
#> [1] " 面包" " 黄油"
```

```
#>
```

```
#> $`3`
```

```
#> [1] " 啤酒"
```

```
#>
```

```
#> $`4`
```

```
#> [1] " 牛奶" " 面包" " 黄油"
```

```
#>
```

```
#> $`5`
```

```
#> [1] " 面包" " 黄油"
```

例 1 (续) 调用 apriori() 函数计算

```
tr = transactions(split(df$菜品 id, df$订单号))  
inspect(tr)          # 展示交易数据或关联规则
```

```
#>      items      transactionID  
#> [1] {a, c, e}      101  
#> [2] {b, d}         102  
#> [3] {b, c}         103  
#> [4] {a, b, c, d}  104  
#> [5] {a, b}         105  
#> [6] {b, c}         106  
#> [7] {a, b}         107  
#> [8] {a, b, c, e}  108  
#> [9] {a, b, c}      109  
#> [10] {a, c, e}     110
```

```

rules = apriori(tr, control = list(verbose = FALSE),
  parameter = list(supp = 0.2, conf = 0.5, minlen = 2))
rules = rules[!is.redundant(rules)] %>%      # 删除冗余规则
  sort(by = "lift") %>%                      # 按提升度降序排列
  inspect()
#>      lhs      rhs support confidence coverage lift  count
#> [1] {a, c} => {e} 0.3      0.600      0.5      2.000 3
#> [2] {e}    => {c} 0.3      1.000      0.3      1.429 3
#> [3] {e}    => {a} 0.3      1.000      0.3      1.429 3
#> [4] {d}    => {b} 0.2      1.000      0.2      1.250 2
#> [5] {c}    => {a} 0.5      0.714      0.7      1.020 5
#> [6] {a}    => {c} 0.5      0.714      0.7      1.020 5
#> [7] {c}    => {b} 0.5      0.714      0.7      0.893 5
#> [8] {b}    => {c} 0.5      0.625      0.8      0.893 5
#> [9] {b}    => {a} 0.5      0.625      0.8      0.893 5
#> [10] {a}   => {b} 0.5      0.714      0.7      0.893 5

```

注：参数 `minlen = 2` 可避免规则表达式左端为空。

结果解释：以第 10 条关联规则为例，客户同时点菜品 a 和 b 的概率是 50%，点了菜品 a 再点菜品 b 的概率是 71.43%。知道了这些，就可以对顾客进行智能推荐，增加销量同时满足客户需求。

例 2 (购物篮分析) 来自 Kaggle 的 Random Shopping cart 数据集。

- 读入数据，剔除重复数据：

```
df = read_csv("datas/dataset_group.csv", col_names = FALSE) %>%  
  set_names("Date", "ID", "Item") %>%  
  distinct()  
head(df)  
#> # A tibble: 6 x 3  
#>   Date          ID Item  
#>   <date>      <dbl> <chr>  
#> 1 2000-01-01      1 yogurt  
#> 2 2000-01-01      1 pork  
#> 3 2000-01-01      1 sandwich bags  
#> 4 2000-01-01      1 lunch meat  
#> 5 2000-01-01      1 all- purpose  
#> 6 2000-01-01      1 flour
```

- 查看项频数

```
df %>% count(Item, sort = TRUE)
```

```
#> # A tibble: 38 x 2
```

```
#>   Item          n
```

```
#>   <chr>      <int>
```

```
#> 1 vegetables    842
```

```
#> 2 poultry       480
```

```
#> 3 ice cream     454
```

```
#> 4 cereals       451
```

```
#> 5 lunch meat    450
```

```
#> 6 waffles       449
```

```
#> 7 cheeses       445
```

```
#> 8 soda          445
```

```
#> 9 eggs          444
```

```
#> 10 dinner rolls 443
```

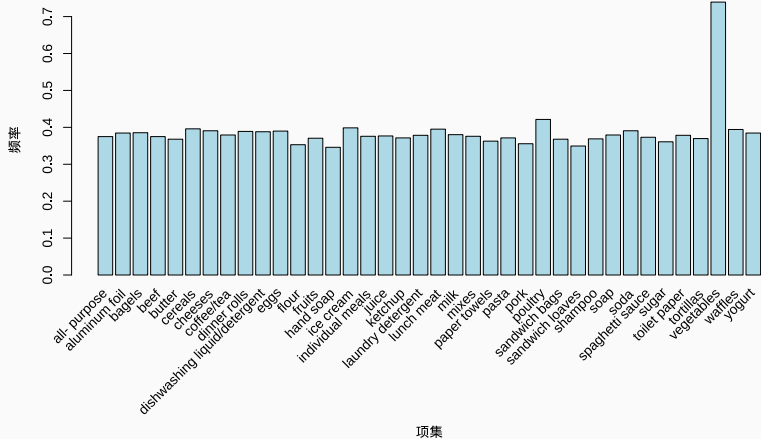
```
#> # ... with 28 more rows
```


- 转化 transactions 数据

```
trans = transactions(split(df$Item, df$ID))
```

- 可视化项频率

```
par(cex = 0.7)
itemFrequencyPlot(trans, support = 0.25, col = "lightblue",
                  xlab = " 项集", ylab = " 频率")
```



- 执行 Apriori 算法挖掘关联规则

```
rules = apriori(trans, control = list(verbose= FALSE),  
  parameter = list(supp = 0.25, conf = 0.6, minlen = 2))  
# 设置保留三位小数  
quality(rules) = round(quality(rules), digits = 3)  
rules = rules[!is.redundant(rules)] %>% # 剔除冗余关联规则  
  sort(by = "lift") # 按提升度从大到小排
```

- 转化为数据框

```
as(rules, "data.frame") %>% as_tibble()
```

```
#> # A tibble: 37 x 6
```

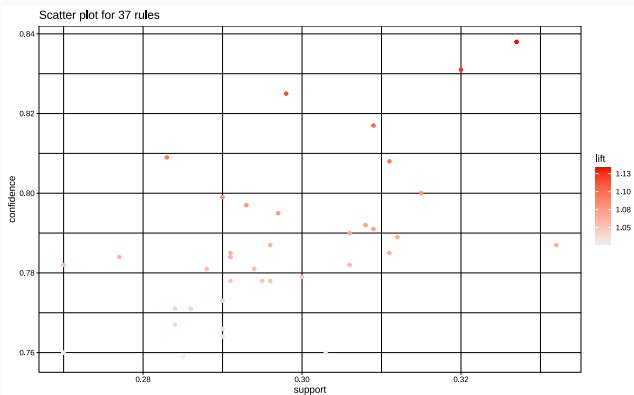
#>	rules	support	confidence
#>	<chr>	<dbl>	<dbl>
#>	1 {eggs} => {vegetables}	0.327	0.838
#>	2 {yogurt} => {vegetables}	0.32	0.831
#>	3 {sugar} => {vegetables}	0.298	0.825
#>	4 {laundry detergent} => {vegetables}	0.309	0.817
#>	5 {sandwich loaves} => {vegetables}	0.283	0.809
#>	6 {aluminum foil} => {vegetables}	0.311	0.808
#>	7 {waffles} => {vegetables}	0.315	0.8
#>	8 {paper towels} => {vegetables}	0.29	0.799
#>	9 {sandwich bags} => {vegetables}	0.293	0.797
#>	10 {spaghetti sauce} => {vegetables}	0.297	0.795
#>	# ... with 27 more rows		

- 可视化结果

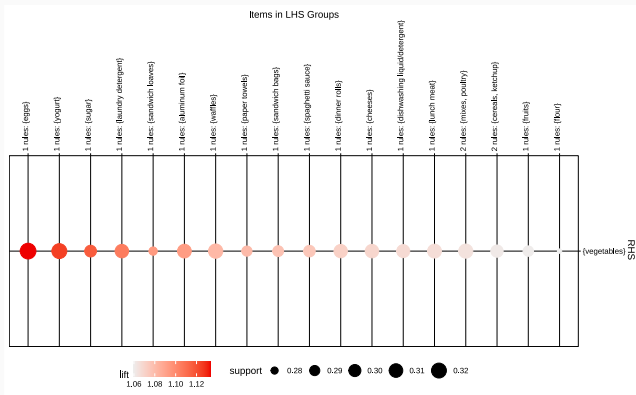
```
library(arulesViz)
```

```
plot(rules)
```

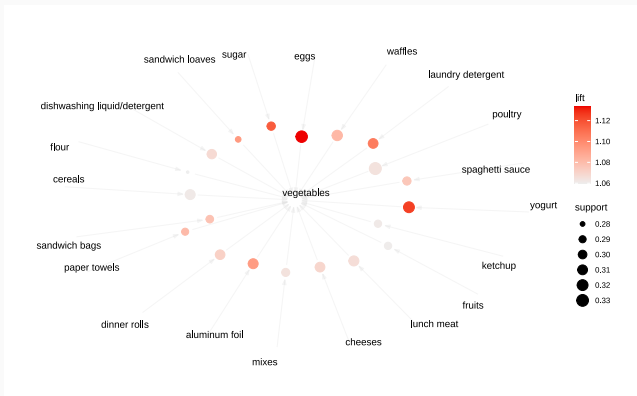
散点图



```
plot(rules[1:20], method = "grouped") # 气泡图
```



```
plot(rules[1:20], method = "graph") # 网络图
```



- [1] 张良均, 云伟标等. R 语言数据分析与挖掘实战. 机械工业出版社, 2016.
- [2] 薛震, 孙玉林. R 语言统计分析与机器学习. 中国水利水电出版社, 2020.
- [3] M Hahsler, B Grün, K Hornik, Introduction to arules - A computational environment for mining association rules and frequent item sets. Journal of Statistical Software 14 (15), 1-25.
- [4] Yanchang Zhao, R and Data Mining: Examples and Case Studies, <http://www.rdatamining.com/docs/RDataMining.pdf>
- [5] 方匡南. 数据科学. 电子工业出版社, 2018.