中山大學
SUN YAT-SEN UNIVERSITY

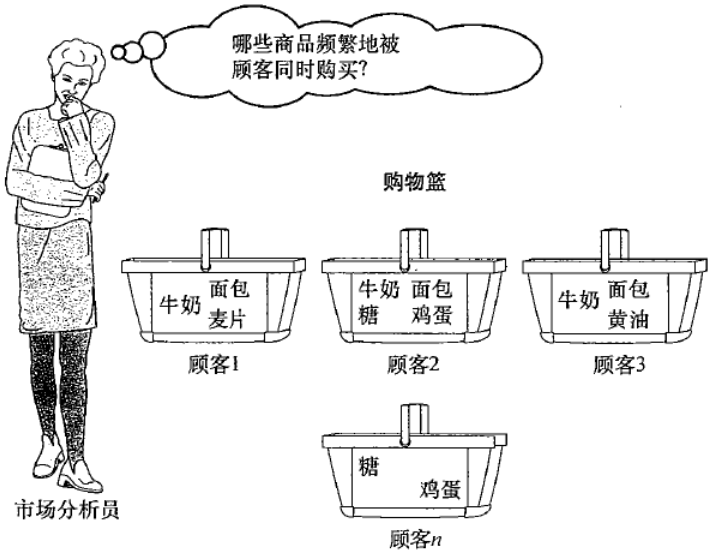
2012级《多元统计分析与数据挖掘》第4周

2015.3.24

中山大学数学与计算科学学院 黄志洪

数据挖掘：关联规则挖掘

- 例子：购物篮分析



购物篮分析的应用

- 超市里的货架摆设设计
- 电子商务网站的套餐推荐

英国史5

当当价 ¥60.70 （6.9折） 钻石VIP专享折上9.5折

定　价 ¥88.00

评　论 ★★★★★ 97.4%推荐 156条

配送至 广东广州市海珠区 有货 运费说明 本商品提供礼品包装服务

今天(3月16日)可送达，请在9小时24分钟内下单并选择 "普通快递送货上门"

作　者 [英]大卫·休谟 著，刘仲敬 译

出版社 吉林出版集团有限责任公司

出版时间 2013-7-1

ISBN 9787553405445

所属分类 图书 > 历史 > 世界史 > 欧洲史

我要买 1 件

加入购物车　一键购买　收藏商品

2015.3.24

购物篮分析的应用

- 推荐系统：网站或节目的阅读/收听推荐

2015.3.24

名词

- 挖掘数据集：购物篮数据
- 频繁模式：频繁地出现在数据集中的模式，例如项集，子结构，子序列等
- 挖掘目标：频繁模式，频繁项集，关联规则等
- 关联规则：牛奶=>鸡蛋【支持度=2%，置信度=60%】
- 支持度：分析中的全部事务的2%同时购买了牛奶和鸡蛋
- 置信度：购买了牛奶的筒子有60%也购买了鸡蛋
- 最小支持度阈值和最小置信度阈值：由挖掘者或领域专家设定

名词

- 项集：项（商品）的集合
- k-项集：k个项组成的项集
- 频繁项集：满足最小支持度的项集，频繁k-项集一般记为L_k
- 强关联规则：满足最小支持度阈值和最小置信度阈值的规则

关联规则挖掘： Apriori算法

- 两步过程：找出所有频繁项集；由频繁项集产生强关联规则
- 算法：Apriori
- 例子

表 6.1　AllElectronics 某分店的事务数据

TID	商品 *ID* 的列表	*TID*	商品 *ID* 的列表
T100	I1，I2，I5	T600	I2，I3
T200	I2，I4	T700	I1，I3
T300	I2，I3	T800	I1，I2，I3，I5
T400	I1，I2，I4	T900	I1，I2，I3
T500	I1，I3		

Apriori算法的工作过程

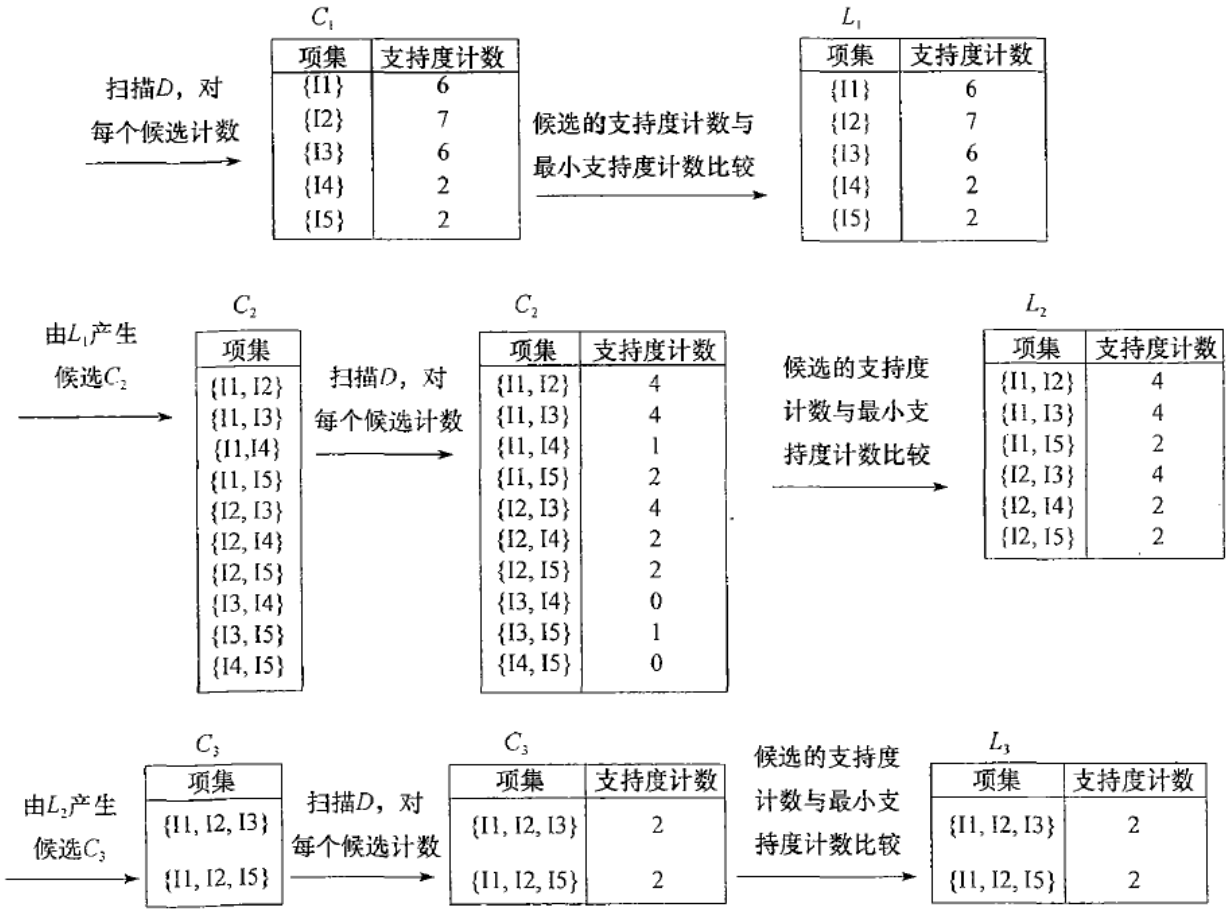


图 6.2　候选项集和频繁项集的产生，最小支持计数为 2

2015.3.24

步骤说明

- 扫描D，对每个候选项计数，生成候选1-项集C1
- 定义最小支持度阈值为2，从C1生成频繁1-项集L1
- 通过L1xL1生成候选2-项集C2
- 扫描D，对C2里每个项计数，生成频繁2-项集L2
- 计算L3xL3，利用apriori性质：频繁项集的子集必然是频繁的，我们可以删去一部分项，从而得到C3，由C3再经过支持度计数生成L3
- 可见Apriori算法可以分成 **连接，剪枝** 两个步骤不断循环重复

步骤说明

（a）连接：$C_3=L_2 \bowtie L_2=\{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$

$\bowtie\{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$

$=\{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}$

（b）使用先验性质剪枝：频繁项集的所有非空子集必须是频繁的。存在候选项集，其子集不是频繁的吗?

- {I1，I2，I3}的2项子集是{I1，I2}、{I1，I3}和{I2，I3}。{I1，I2，I3}的所有2项子集都是L_2的元素。因此，{I1，I2，I3}保留在C_3中。
- {I1，I2，I5}的2项子集是{I1，I2}、{I1，I5}和{I2，I5}。{I1，I2，I5}的所有2项子集都是L_2的元素。因此，{I1，I2，I5}保留在C_3中。
- {I1，I3，I5}的2项子集是{I1，I3}、{I1，I5}和{I3，I5}。{I3，I5}不是L_2的元素，因而不是频繁的。因此，从C_3中删除{I1，I3，I5}。
- {I2，I3，I4}的2项子集是{I2，I3}、{I2，I4}和{I3，I4}。{I3，I4}不是L_2的元素，因而不是频繁的。因此，从C_3中删除{I2，I3，I4}。
- {I2，I3，I5}的2项子集是{I2，I3}、{I2，I5}和{I3，I5}。{I3，I5}不是L_2的元素，因而不是频繁的。因此，从C_3中删除{I2，I3，I5}。
- {I2，I4，I5}的2项子集是{I2，I4}、{I2，I5}和{I4，I5}。{I4，I5}不是L_2的元素，因而不是频繁的。因此，从C_3中删除{I2，I3，I5}。

（c）因此，剪枝后$C_3=\{\{I1, I2, I3\}, \{I1, I2, I5\}\}$。

由频繁项集提取关联规则

- 例子：我们计算出频繁项集{I1,I2,I5}，能提取哪些规则？

I1^I2=>I5，由于{I1,I2,I5}出现了2次，{I1,I2}出现了4次，故置信度为2/4=50%

类似可以算出

$$\{I1, I2\} \Rightarrow I5, \quad confidence = 2/4 = 50\%$$
$$\{I1, I5\} \Rightarrow I2, \quad confidence = 2/2 = 100\%$$
$$\{I2, I5\} \Rightarrow I1, \quad confidence = 2/2 = 100\%$$
$$I1 \Rightarrow \{I2, I5\}, \quad confidence = 2/6 = 33\%$$
$$I2 \Rightarrow \{I1, I5\}, \quad confidence = 2/7 = 29\%$$
$$I5 \Rightarrow \{I1, I2\}, \quad confidence = 2/2 = 100\%$$

用 R 进行购物篮分析

- 安装arules包并加载
- 内置Groceries数据集

library(arules) #加载arules程序包

data(Groceries) #调用数据文件

Inspect(Groceries) #观看数据集里的数据

```
      specialty bar}
9823 {yogurt,
      long life bakery product}
9824 {pork,
      frozen vegetables,
      pastry}
9825 {ice cream,
      long life bakery product,
      specialty chocolate,
      specialty bar}
9826 {chicken,
      hamburger meat,
      citrus fruit,
```

用 R 进行购物篮分析

- 求频繁项集

frequentsets=eclat(Groceries,parameter=list(support=0.05,maxlen=10))

```
parameter specification:
 tidLists support minlen maxlen            target   ext
    FALSE    0.05      1     10 frequent itemsets FALSE

algorithmic control:
 sparse sort verbose
      7   -2    TRUE

eclat - find frequent item sets with the eclat algorithm
version 2.6 (2004.08.16)          (c) 2002-2004   Christian Borgelt
create itemset ...
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [28 item(s)] done [0.00s].
creating sparse bit matrix ... [28 row(s), 9835 column(s)] done [0.00s].
writing  ... [31 set(s)] done [0.02s].
Creating S4 object  ... done [0.00s].
```

用 R 进行购物篮分析

- 观看频繁项集

```
inspect(frequentsets[1:10])
inspect(sort(frequentsets,by="support")[1:10])    #根据支持度对求得的频繁项集排序并察看
```

```
   items                  support
1  {whole milk}           0.25551601
2  {other vegetables}     0.19349263
3  {rolls/buns}           0.18393493
4  {soda}                 0.17437722
5  {yogurt}               0.13950178
6  {bottled water}        0.11052364
7  {root vegetables}      0.10899847
8  {tropical fruit}       0.10493137
9  {shopping bags}        0.09852567
10 {sausage}              0.09395018
```

用 R 进行购物篮分析

- 利用apriori函数提取关联规则

rules=apriori(Groceries,parameter=list(support=0.01,confidence=0.5))

```
> rules=apriori(Groceries,parameter=list(support=0.01,confidence=0.5))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target   ext
        0.5    0.1    1 none FALSE            TRUE    0.01      1     10  rules FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [88 item(s)] done [0.00s].
creating transaction tree ... done [0.02s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [15 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

用 R 进行购物篮分析

- 列出关联规则

summary(rules)　#察看求得的关联规则之摘要

inspect(rules)

```
> inspect(rules)
    lhs                          rhs                    support confidence     lift
1   {curd,
     yogurt}                  => {whole milk}       0.01006609  0.5823529 2.279125
2   {other vegetables,
     butter}                  => {whole milk}       0.01148958  0.5736041 2.244885
3   {other vegetables,
     domestic eggs}           => {whole milk}       0.01230300  0.5525114 2.162336
4   {yogurt,
     whipped/sour cream}      => {whole milk}       0.01087951  0.5245098 2.052747
5   {other vegetables,
     whipped/sour cream}      => {whole milk}       0.01464159  0.5070423 1.984385
6   {pip fruit,
     other vegetables}        => {whole milk}       0.01352313  0.5175097 2.025351
7   {citrus fruit,
     root vegetables}         => {other vegetables} 0.01037112  0.5862069 3.029608
8   {tropical fruit,
     root vegetables}         => {other vegetables} 0.01230300  0.5845411 3.020999
9   {tropical fruit,
```

用 R 进行购物篮分析

- 按需要筛选关联规则

```
x=subset(rules,subset=rhs%in%"whole milk"&lift>=1.2)    #求所需要的关联规则子集
inspect(sort(x,by="support")[1:5])    #根据支持度对求得的关联规则子集排序并察看
```

其中 lift=P(L,R)/(P(L)P(R)) 是一个类似相关系数的指标。lift=1时表示L和R独立。这个数越大，越表明L和R存在在一个购物篮中不是偶然现象。

提高Apriori的效率

- 基于散列的算法
- 基于FP tree的算法

FP树

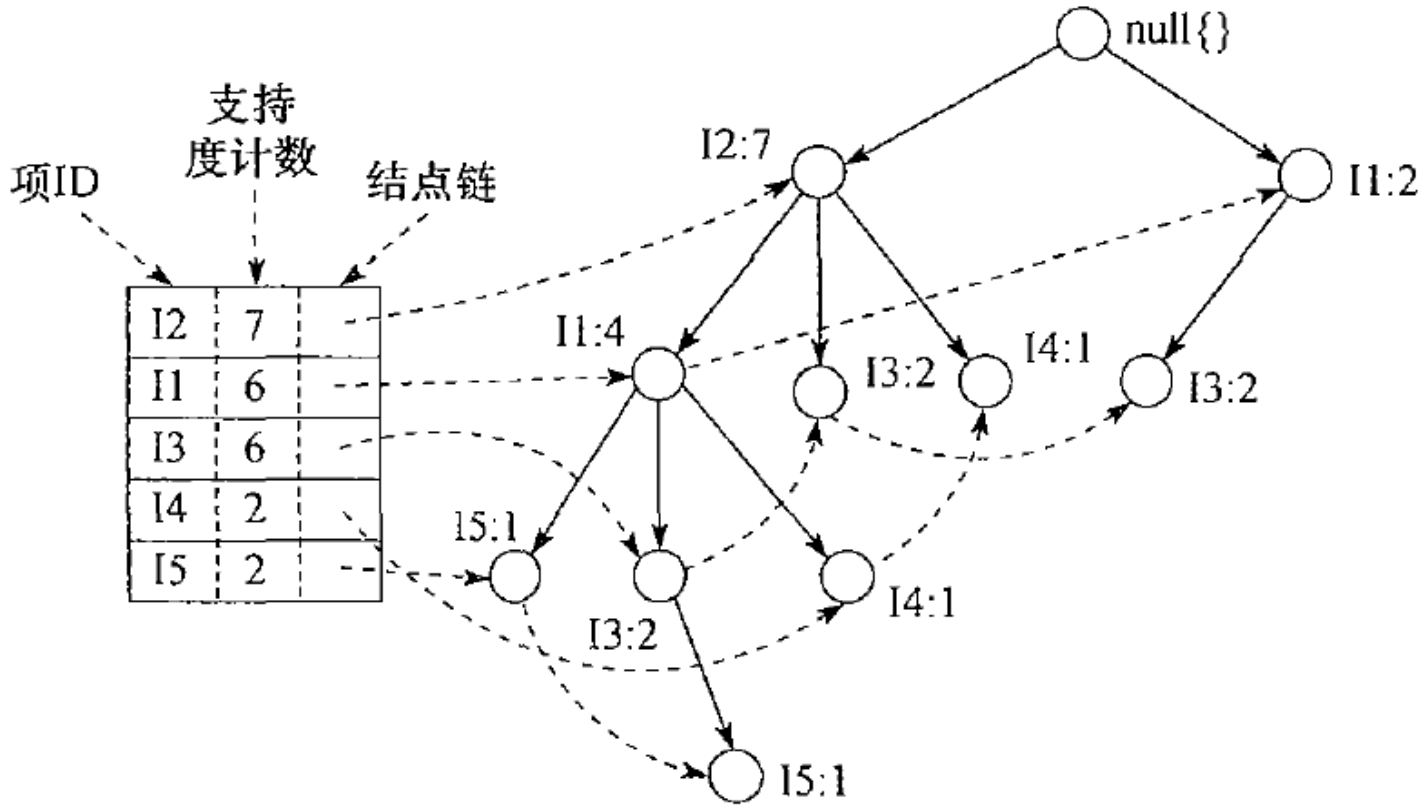


图 6.7　存放压缩的频繁模式信息的 FP 树

挖掘过程图示

表 6.2　通过创建条件（子）模式基挖掘 FP 树

项	条件模式基	条件 FP 树	产生的频繁模式
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}、{I1, I5: 2}、{I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}、{I1, I3: 4}、{I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}

到達I5的路徑

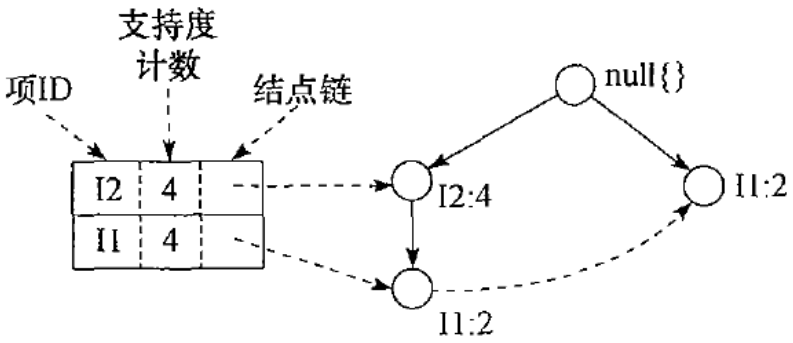


图 6.8　与条件结点 I3 相关联的条件 FP 树

FP-Growth算法

中山大學 SUN YAT-SEN UNIVERSITY

算法：**FP-Growth**。使用 FP 树，通过模式增长挖掘频繁模式。

输入：

- D：事务数据库。
- min_sup：最小支持度阈值。

输出：频繁模式的完全集。

方法：

1. 按以下步骤构造 FP 树：
   (a) 扫描事务数据库 D 一次。收集频繁项的集合 F 和它们的支持度计数。对 F 按支持度计数降序排序，结果为频繁项列表 L。
   (b) 创建 FP 树的根结点，以“null”标记它。对于 D 中每个事务 $Trans$，执行：
   选择 $Trans$ 中的频繁项，并按 L 中的次序排序。设 $Trans$ 排序后的频繁项列表为 $[p|P]$，其中 p 是第一个元素，而 P 是剩余元素的列表。调用 `insert_tree`$([p|P],T)$。该过程执行情况如下。如果 T 有子女 N 使得 $N.item\text{-}name = p.item\text{-}name$，则 N 的计数增加 1；否则，创建一个新结点 N，将其计数设置为 1，链接到它的父结点 T，并且通过结点链结构将其链接到具有相同 $item\text{-}name$ 的结点。如果 P 非空，则递归地调用 `insert_tree`(P,N)。
2. FP 树的挖掘通过调用 `FP_growth`$(FP\_tree, null)$ 实现。该过程实现如下。

```
procedure FP_growth(Tree, α)
(1) if Tree 包含单个路径 P then
(2) for 路径 P 中结点的每个组合(记作 β)
(3) 产生模式 β∪α,其支持度计数 support_count 等于 β 中结点的最小支持度计数;
(4) else for Tree 的头表中的每个 a_i {
(5) 产生一个模式 β = a_i ∪ α,其支持度计数 support_count = a_i.support_count;
(6) 构造 β 的条件模式基,然后构造 β 的条件 FP 树 Tree_β;
(7) if Tree_β ≠ ∅ then
(8) 调用 FP_growth(Tree_β, β); }
```

PFP算法

- mahout提供了内存中的FPG和分布式的PFP两种算频繁项集的方法
- Parallel Frequent Pattern Mining ?
- Parallel FPGrowth ?
- https://cwiki.apache.org/confluence/display/MAHOUT/Parallel+Frequent+Pattern+Mining
- http://infolab.stanford.edu/~echang/recsys08-69.pdf

分布式FP-Growth

Map inputs (transactions) key="": value	Sorted transactions (with infrequent items eliminated)	Map outputs (conditional transactions) key: value
f a c d g i m p	f c a m p	p: f c a m m: f c a a: fc c: f
a b c f l m o	f c a b m	m: f c a b b: f c a a: fc c: f
b f h j o	f b	b: f
b c k s p	c b p	p: c b b: c
a f c e l p m n	f c a m p	p: f c a m m: f c a a: fc c: f

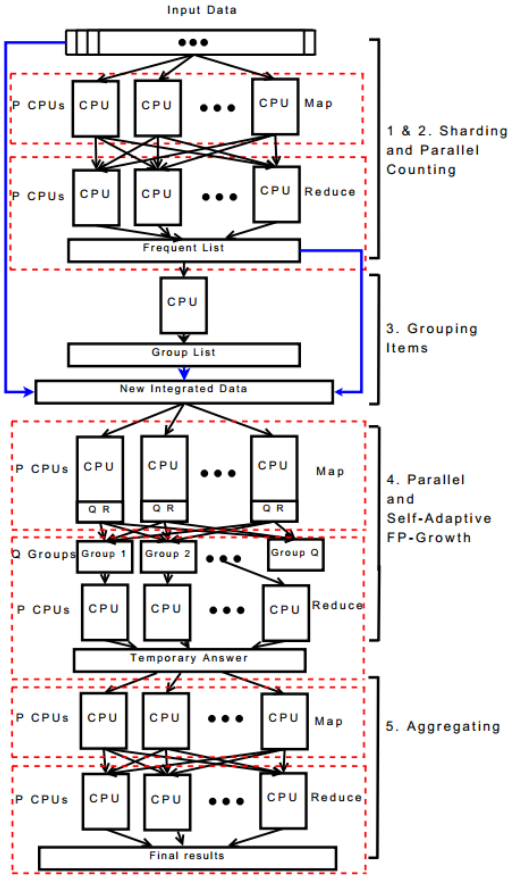
Reduce inputs (conditional databases) key: value		Conditional FP−trees
p:	{ f c a m / f c a m / c b }	{(c:3)} | p
m:	{ f c a / f c a / f c a b }	{ (f:3, c:3, a:3) } | m
b:	{ f c a / f / c }	{} | b
a:	{ f c / f c / f c }	{ (f:3, c:3) } | a
c:	{ f / f / f }	{ (f:3) } | c

Figure 1: A simple example of distributed FP-Growth.

主要步骤

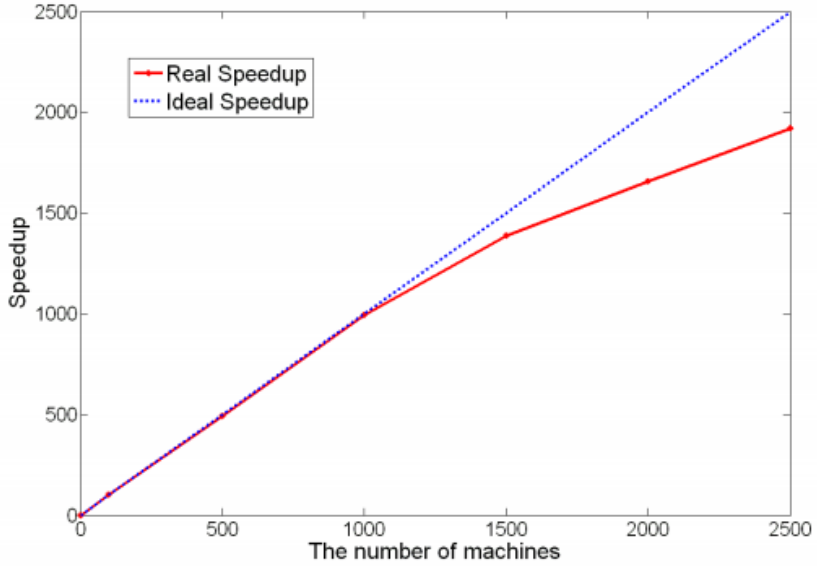
- 将数据集分片
- 计数，产生排序的F-List
- 将物品分组，产生G-List
- （PFP算法关键步骤）并行FP-Growth过程
- 聚合结果

PFP算法的五个阶段示意图



2015.3.24

PFP计算能力评估



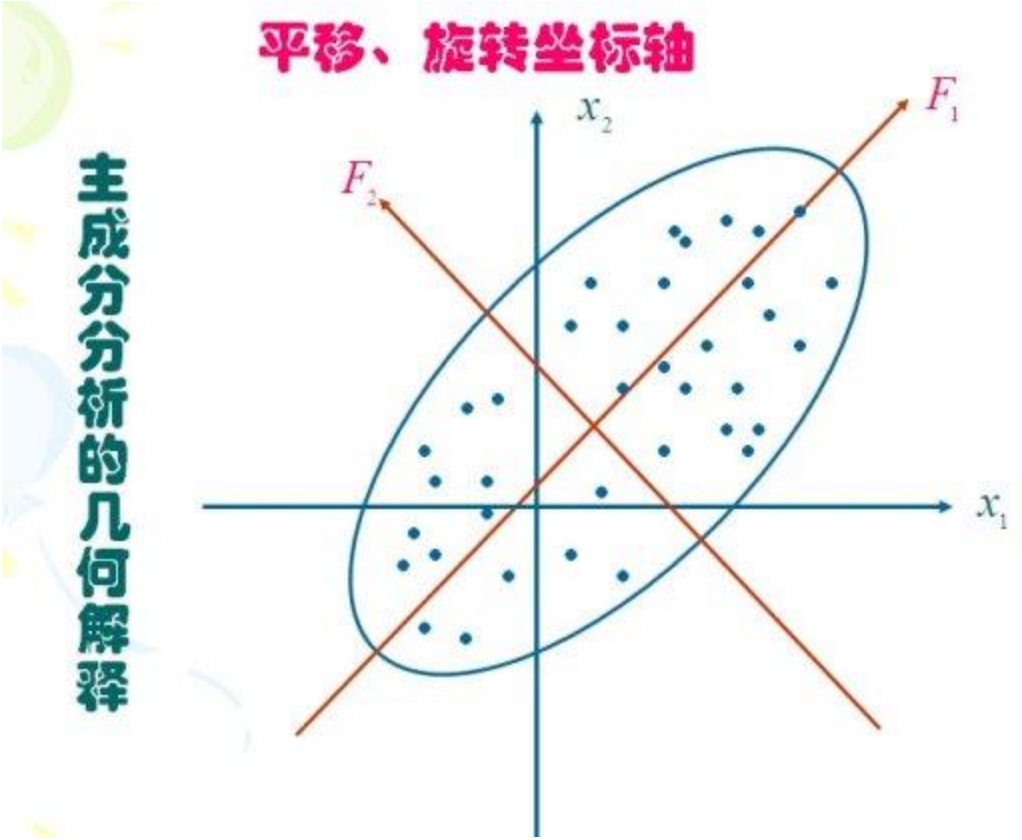
#. machines	#. groups	Time (sec)	Speedup
100	50000	27624	100.0
500	50000	5608	492.6
1000	50000	2785	991.9
1500	50000	1991	1387.4
2000	50000	1667	1657.1
2500	50000	1439	1919.7

Figure 4: The speedup of the PFP algorithm.

主成分分析

- 通过对原始变量进行线性组合，得到优化的指标
- 把原先多个指标的计算降维为少量几个经过优化指标的计算（占去绝大部分份额）
- 基本思想：**设法将原先众多具有一定相关性的指标，重新组合为一组新的互相独立的综合指标，并代替原先的指标**

主成分分析的直观几何意义



主成分分析的数学模型

- 薛毅书电子版p499
- 主成分分析思想最终可以通过矩阵写法转变为求解线性代数问题

设 X 是 p 维随机变量，并假设 $\mu = E(X)$, $\Sigma = \mathrm{Var}(X)$. 考虑如下线性变换

$$\begin{cases} Z_1 = a_1^T X \\ Z_2 = a_2^T X \\ \quad \vdots \\ Z_p = a_p^T X \end{cases}, \tag{9.1}$$

易见

$$\mathrm{Var}(Z_i) = a_i^T \Sigma a_i, \quad i = 1, 2, \cdots, p, \tag{9.2}$$

$$\mathrm{Cov}(Z_i, Z_j) = a_i^T \Sigma a_j, \quad i, j = 1, 2, \cdots, p, \quad i \neq j. \tag{9.3}$$

我们希望 Z_1 的方差达到最大，即 a_1 是约束优化问题

$$\begin{aligned} \max \quad & a^T \Sigma a \\ \text{s.t.} \quad & a^T a = 1 \end{aligned}$$

数学模型的求解

- 转化为将协方差矩阵对角化的问题（求解特征值）

一般情况. 对于协方差阵 Σ, 存在正交阵 Q, 将它化为对角阵, 即

$$Q^T \Sigma Q = \Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_p \end{bmatrix}, \tag{9.4}$$

且 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$. 则矩阵 Q 的第 i 列就对应于 a_i, 相应的 Z_i 为第 i 主成分.

一些性质和名词

- 性质（薛毅书电子版第501页）
- 主成分的贡献率
- 主成分的累计贡献率
- 主成分在原始变量上的载荷

基于样本的求解

- 为了抵消量纲的影响，可以从相关系数矩阵出发求解
- 样本相关系数矩阵

$$S = \frac{1}{n-1}\sum_{k=1}^{n}\left(X_{(k)}-\overline{X}\right)\left(X_{(k)}-\overline{X}\right)^T = (s_{ij})_{p\times p},$$

其中

$$\overline{X} = \frac{1}{n}\sum_{k=1}^{n}X_{(k)} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_p)^T,$$

$$s_{ij} = \frac{1}{n-1}\sum_{k=1}^{n}(x_{ki}-\bar{x}_i)(x_{kj}-\bar{x}_j), \quad i,j = 1,2,\cdots,p.$$

及样本的相关矩阵 R 为

$$R = \frac{1}{n-1}\sum_{k=1}^{n}X^*_{(k)}{X^*_{(k)}}^T = (r_{ij})_{p\times p},$$

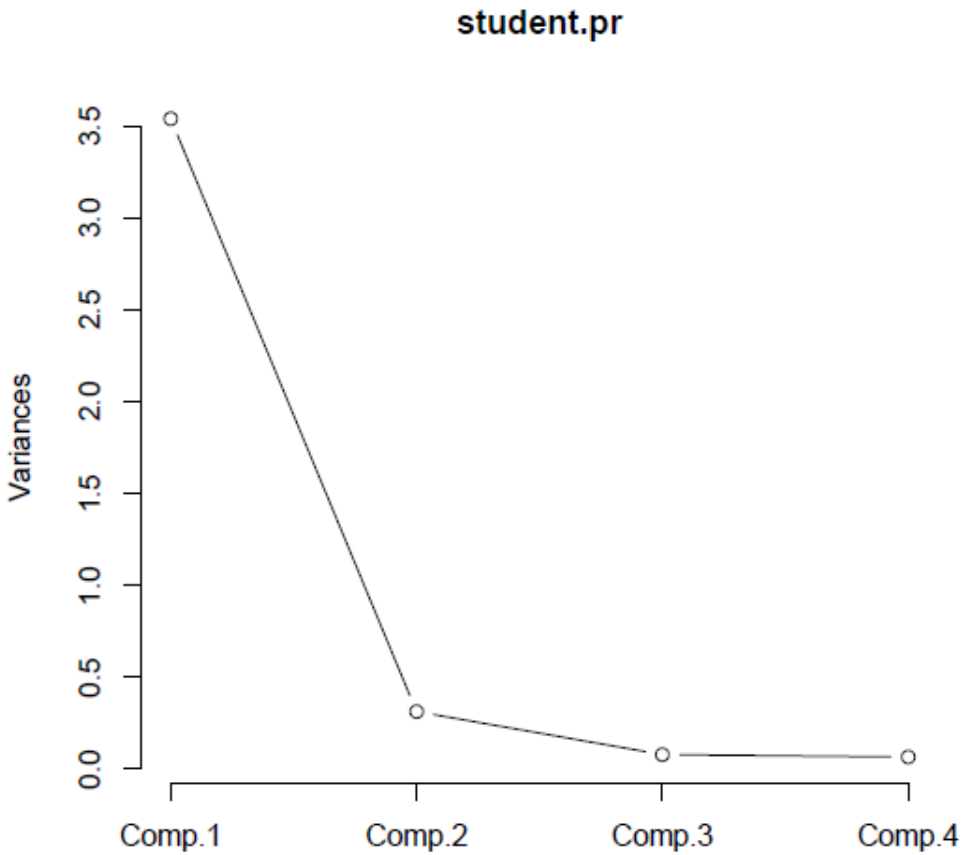
其中

$$X^*_{(k)} = \left[\frac{x_{k1}-\bar{x}_1}{\sqrt{s_{11}}}, \frac{x_{k2}-\bar{x}_2}{\sqrt{s_{22}}}, \cdots, \frac{x_{kp}-\bar{x}_p}{\sqrt{s_{pp}}}\right],$$

$$r_{ij} = \frac{s_{ij}}{\sqrt{s_{ii}s_{jj}}}, \quad i,j = 1,2,\cdots,p.$$

R中进行主成分分析

- 薛毅书P506
- princomp函数
- summary函数
- loadings函数
- predict函数
- 碎石图与screeplot函数
- 主成分方向，biplot函数
- 例子：薛毅书P508



主成分方向图

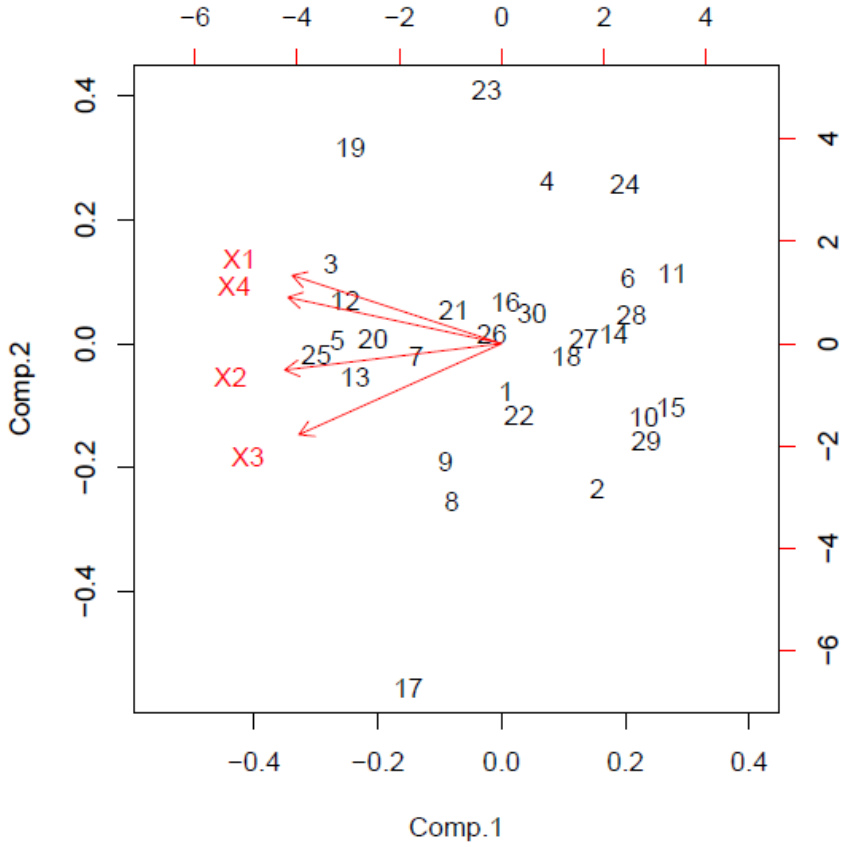


图 9.2: 30 名中学生身体指标数据关于第 1 主成分和第 2 主成分的散点图

例子：求相关矩阵特征值

- 薛毅书p487

```
> PCA=princomp(X,cor=T)
> PCA
Call:
princomp(x = X, cor = T)

Standard deviations:
   Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6    Comp.7
2.2556395 1.1632889 0.7567221 0.6376603 0.5278638 0.3502837 0.3063912
   Comp.8
0.2905094

 8  variables and  31 observations.
> PCA$loadings
```

例子：求主成分载荷

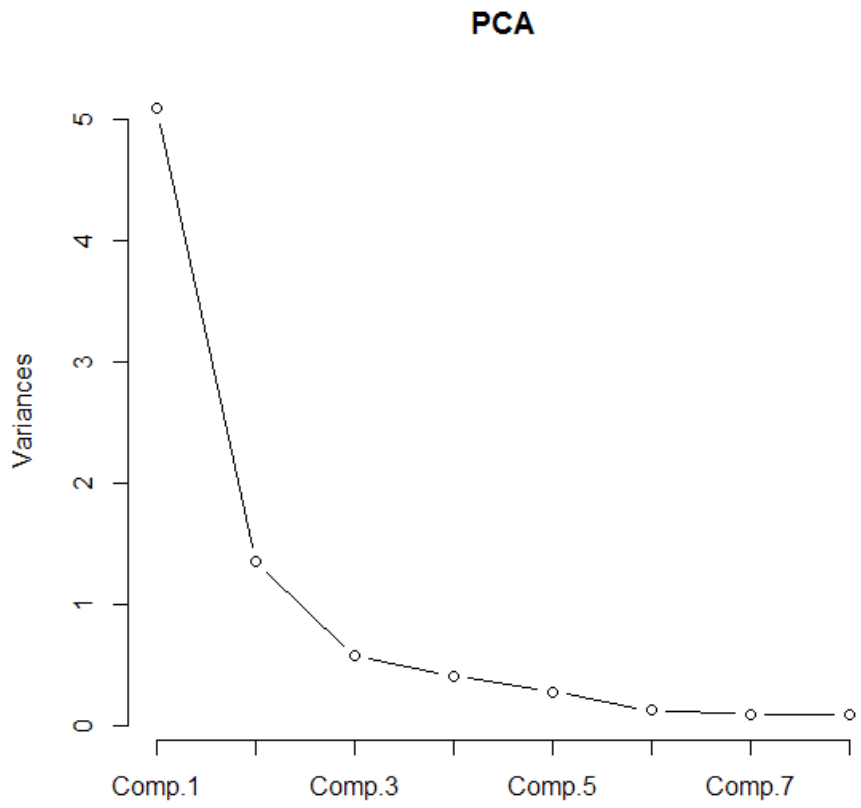
```
> PCA$loadings

Loadings:
   Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
x1 -0.399          0.416  0.214 -0.217         -0.280  0.693
x2 -0.132  0.749  0.339  0.157  0.523
x3 -0.375        -0.444  0.544         -0.562 -0.161 -0.121
x4 -0.320  0.346 -0.475 -0.657                        0.335
x5 -0.388 -0.231  0.282 -0.364  0.210 -0.109 -0.566 -0.456
x6 -0.406        -0.308  0.234         0.795         -0.229
x7 -0.327 -0.495                0.582          0.514  0.182
x8 -0.396          0.338 -0.116 -0.538 -0.127  0.551 -0.312

               Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
Proportion Var  0.125  0.125  0.125  0.125  0.125  0.125  0.125  0.125
Cumulative Var  0.125  0.250  0.375  0.500  0.625  0.750  0.875  1.000
> 
```

例子：画碎石图确定主成分

```
> screeplot(PCA,type="lines")
```



例子：主成分得分-相当于predict( )

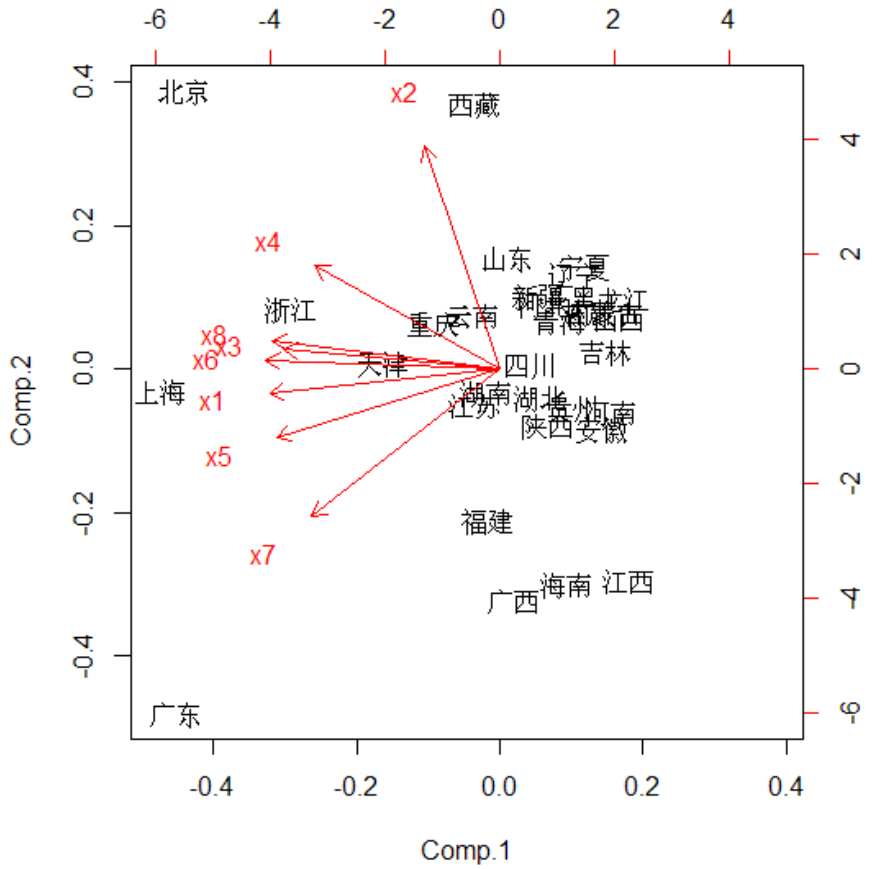
```
> PCA$score
          Comp.1      Comp.2      Comp.3      Comp.4      Comp.5      Comp.6      Comp
北京    -5.5068881  2.51368747 -0.77052784 -0.34499076 -0.48456544  0.73526042  0.142820
天津    -2.0391525  0.04696816 -0.83866069  0.84294280 -0.23905123 -0.36965072  0.438523
河北     0.7647412  0.58939950 -0.63809135 -0.40004970  0.32727289  0.02069393 -0.108875
山西     2.1042564  0.45779593 -0.29703426 -0.21190291 -0.16277216 -0.21169100  0.366478
内蒙古   1.8368141  0.51548336  0.14950198 -0.09308007  0.19160016  0.13617218 -0.010774
辽宁     1.3232250  0.85489639 -0.05242441 -0.56123733  0.43320901  0.10274050 -0.199007
吉林     1.8750798  0.14967842 -0.02016675 -0.28215689  0.45133137  0.36714488 -0.038957
黑龙江   1.9411347  0.64393452 -0.25831381 -0.84845435  0.37526772 -0.08315897 -0.086928
上海    -5.9397413 -0.19531943  0.09487298  1.07297060 -0.60041434 -0.09156896  0.065314
江苏    -0.4173225 -0.31874237 -0.21558331  0.85952388 -0.39145266 -0.42795347 -0.199799
浙江    -3.6407775  0.54489693 -0.77999195 -0.68115276  0.19016696 -0.41219749 -0.509992
安徽     1.8169295 -0.53363884  0.33919645  0.64984975 -0.04126297  0.49854622 -0.528359
福建    -0.1976522 -1.36531052  1.29563886  0.23492502  0.12124119 -0.19422385 -0.489680
江西     2.2557443 -1.90231267  0.08063848  0.33710287  0.09292676  0.00724231  0.403240
山东     0.1360728  0.99920233 -0.34711211  0.92327895  0.53080961 -0.29793692 -0.123394
河南     1.9613045 -0.39761168 -0.20088982 -0.23566368  0.30206294 -0.49375497  0.224554
湖北     0.7167909 -0.25396283 -0.03587219  0.29134913  0.81888494  0.66366667  0.443813
湖南    -0.2318682 -0.20807224 -0.01570997  0.47810304  0.47020168  0.52874605  0.065600
广东    -5.6676807 -3.11520051  0.51838684 -1.53211943  0.90023275 -0.21946848  0.129630
广西     0.2480444 -2.09427753 -0.03594804  0.29165788 -0.04979176  0.44518529  0.146873
海南     1.1715466 -1.94839070  0.44408295 -0.60362333 -1.85888240  0.34575391 -0.284233
重庆    -1.1363085  0.41532157  0.13949690  0.63934241  0.56936685  0.28511495 -0.703780
四川     0.5349560  0.03922716  0.17181794  0.42545284  0.12711946  0.30779276  0.254154
```

例子：结果解释

- Z1：日常必需消费开支
- Z2：衣着和居住
- 解读是非常重要的环节，甚至决定主成分分析的成败

例子：成分图

```
> biplot(PCA,choices=1:2,scale=1)
```



2015.3.24

例子：聚类

```
> kmeans(PCA$score[,1:2],5)
K-means clustering with 5 clusters of sizes 7, 4, 10, 6, 4

Cluster means:
      Comp.1      Comp.2
1  0.6787254  0.27889640
2 -5.1887719 -0.06298388
3  1.7232375  0.27928061
4 -0.7843413  0.46952434
5  0.8694208 -1.82757285

Clustering vector:
  北京   天津   河北   山西 内蒙古   辽宁   吉林 黑龙江   上海   江苏
     2      4      1      3      3      3      3      3      2      4
  浙江   安徽   福建   江西   山东   河南   湖北   湖南   广东   广西
     2      3      5      5      1      3      1      4      2      5
  海南   重庆   四川   贵州   云南   西藏   陕西   甘肃   青海   宁夏
     5      4      1      3      4      4      1      3      1      3
  新疆
     1
```

主成分回归

- 薛毅书P516

Thanks

FAQ时间