# 回归分析

setwd("E:/teaching\_plan\_notes/msa11091083/rmd")

getwd()

data(women)#读取women数据框

?women#查看数据的介绍

par(ask=TRUE)

opar <- par(no.readonly=TRUE)

## 查看数据

women#查看数据框

view(women)#显示数据框

head(women)#前六行

tail(women)#后六行

class(women)#显示变量类型

dim(women)#读取行数列数

women$height

women$weight

women[1,2]#第一行第二列

women[1,]#第一行

women[100,]

women[,2]#第二列

women[12:14,]#第12到14行

women[12:14,2]#第12到14行的第二列

## 求值

summary(women)#拟合前求变量的最小值，Q1，平均值，Q3，最大值

summary(women$height)

var(women$height)#求方差

sd(women$height)#求标准差

min(women$height)#求最小值

max(women$height)#求最大值

quantile(women$weight)#求分位数

data\_outline<-c(mean(women$height),sd(women$height),mean(women$weight),sd(women$weight))#计算women$height, women$weight 的均值和标准差

cov(women)#求协方差

### 建立模型后：

summary(fit)#查看模型建立的结果，各系数的值，R2，p值检验等

coef(fit)#求系数的最小二乘回归

fit$coefficients

confint(fit,level=0.95)#计算置信区间

fitted(fit)#拟合值y\_hat

fit$fitted.values

resid(fit)#残差值e\_hat

fit$residuals

anova(fit)# 方差分析表,方差显著性检验

shapiro.test(fit$residuals)#对残差做正态性检验

cov(fit)# 模型参数的协方差矩阵

AIC(fit, k=2)# AIC的值

AIC(fit)

cor(fit)#求fit的相关系数

cor.text(fit$x, fit$y)#求y关于x的相关系数及做相关性检验

SRE<-rstandard(fit) #学生化残差

rstudent(fit)

car::outlierTest()#异常值检测，看最后一个P 值

### 公式计算系数估计：

LXX = sum((w$rm-mean(w$rm))^2)#(w$rm-w$rm均值的平方和)

Lxy = sum((w$rm-mean(w$rm))\*(w$cmedv-mean(w$cmedv)))

beta1 = Lxy/Lxx#β1的估计

beta0 = mean(y) - beta1\*mean(x)# β0的估计

## 画图

hist(w[,"cmedv"])#建立w数据的cmedv变量的直方图

boxplot(w[,"cmedv"])#建立w数据的cmedv变量的箱线图

plot(x,y)/plot(x1,y,1x2,y2,……)#画散点图

abline(w)#绘制拟合曲线

lines(x,y)

plot(w)#绘制w数据的回归诊断图

qqnorm(a1$resid)#画qq正态残差的图

qqline(a1$resid)#关于残差做正态性检验，画线

library(GGally)#加载GGally

library(corrplot)#加载corrplot

ggpairs(w)#绘制变量相关图

ggcorr(w)#绘制线性相关系数图

library(car) #加载car包

scatterplot(weight~height,data=women,spread=FALSE,smoother.args=list(lty=2),pch=19)#散点图，线性拟合曲线。平滑拟合曲线

scatterplotMaticx(states,spread=FALSE,smoother.args=list(lty=2),main="Scatter Plot Matrix")#散点图矩阵

### 可视化误差：

qqplot(fit,labels=row.names(states),id.method="identify",simulate=TRUE, main="Q-Q Plot")#学生化残差的正态分布图

residplot <- function(fit, nbreaks=10) {

z <- rstudent(fit)

hist(z, breaks=nbreaks, freq=FALSE, xlab="Studentized Residual", main="Distribution of Errors")

rug(jitter(z), col="brown")

curve(dnorm(x, mean=mean(z), sd=sd(z)), add=TRUE, col="blue", lwd=2)

lines(density(z)$x, density(z)$y, col="red", lwd=2, lty=2)

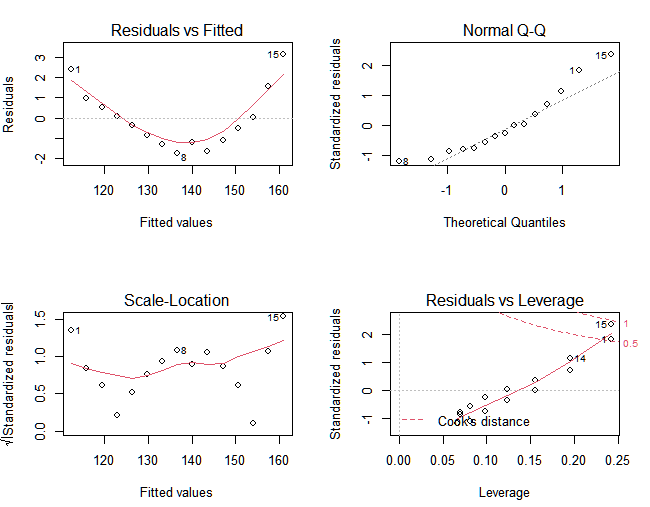
legend("topright",

legend = c( "Normal Curve", "Kernel Density Curve"), lty=1:2, col=c("blue","red"), cex=.7)

}#生成学生化残差柱状图（即直方图），并添加正态曲线、核密度曲线和轴须图。

residplot(fit)

## 诊断图：（https://zhuanlan.zhihu.com/p/341318994）



独立性：你无法从这些图中分辨出因变量值是否相互独立，只能从收集的数据中来验证。上面的例子中，没有任何先验的理由去相信一位女性的体重会影响另外一位女性的体重。假若你发现数据是从一个家庭抽样得来的，那么可能必须要调整模型独立性的假设。

线性：若因变量与自变量线性相关，那么残差值与预测（拟合）值就没有任何系统关联。换句话说，除了白噪声，模型应该包含数据中所有的系统方差。在“残差图与拟合图”（Residuals vs Fitted，左上）中可以清楚地看到一个曲线关系，这暗示着你可能需要对回归模型加上一个二次项。

正态性：当预测变量值固定时，因变量成正态分布，则残差值也应该是一个均值为0的正态分布。“正态Q-Q图”（Normal Q-Q，右上）是在正态分布对应的值下，标准化残差的概率图。若满足正态假设，那么图上的点应该落在呈45度角的直线上；若不是如此，那么就违反了正态性的假设。

同方差性：若满足不变方差假设，那么在“位置尺度图”（Scale-Location Graph，左下）中，水平线周围的点应该随机分布。该图似乎满足此假设。

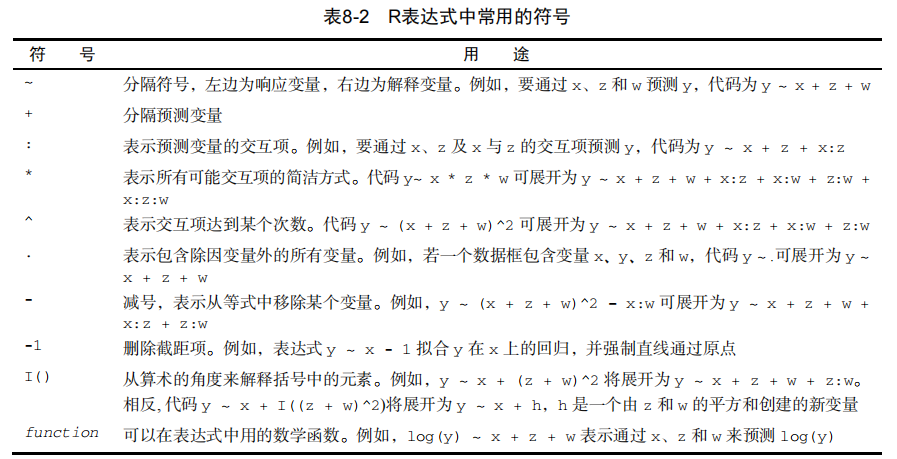
最后一幅“残差与杠杆图”（Residuals vs Leverage，右下）提供了你可能关注的单个观测点的信息。从图形可以鉴别出离群点、高杠杆值点和强影响点。下面来详细介绍。

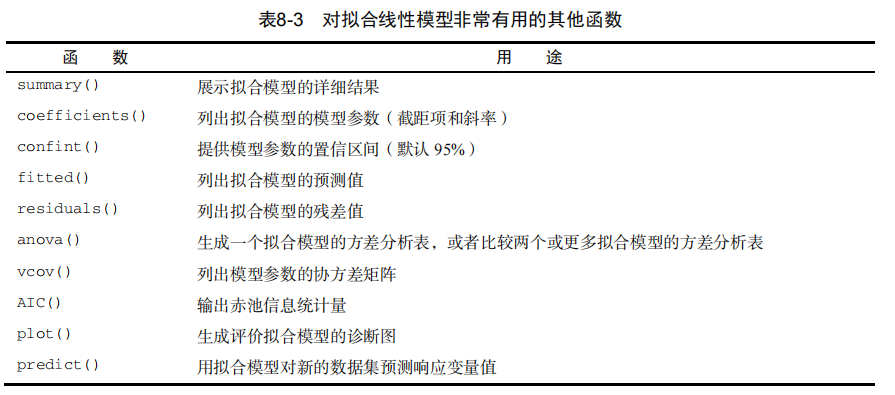
（1）一个观测点是离群点，表明拟合回归模型对其预测效果不佳（产生了巨大的或正或负的残差）。

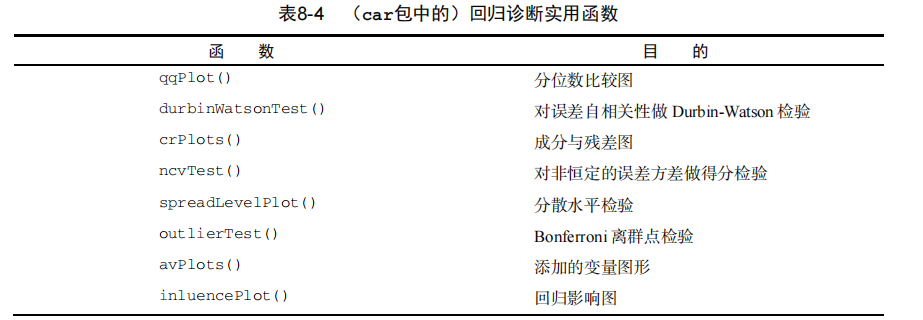
（2）一个观测点有很高的杠杆值，表明它是一个异常的预测变量值的组合。也就是说，在预测变量空间中，它是一个离群点。因变量值不参与计算一个观测点的杠杆值。

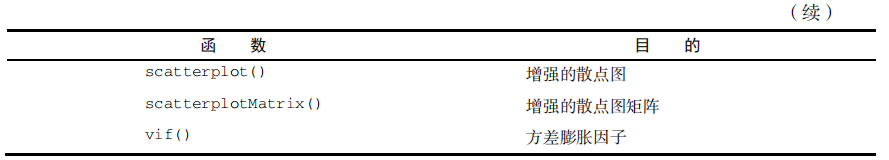
（3）一个观测点是强影响点（influential observation），表明它对模型参数的估计产生的影响过大，非常不成比例。强影响点可以通过Cook距离即Cook’s D统计量来鉴别。

## 建立回归模型









## 1.1简单线性回归

> **fit <- lm(weight ~ height, data=women)#weight hat=β0+height\*β1**

> **fit#系数估计**

# Call:

# lm(formula = weight ~ height, data = women)

#

# Coefficients:

# (Intercept)**β0**  height**β1**

# -87.52 3.45

**#拟合线性回归weight hat=-87.52+3.45\*height**

**> summary(fit)**

Call:

lm(formula=weight ~ height, data=women)

Residuals:

Min 1Q Median 3Q Max

-1.733 -1.133 -0.383 0.742 3.117

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) **-87.5167** 5.9369 -14.7 **1.7e-09** \*\*\*

height **3.4500** 0.0911 37.9 **1.1e-14** \*\*\*

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 '' 1

Residual standard error: **1.53** on 13 degrees of freedom

Multiple R-squared: **0.991**, Adjusted R-squared: 0.99

F-statistic: 1.43e+03 on 1 and 13 DF, p-value: **1.09e-14**

**> women$weight**

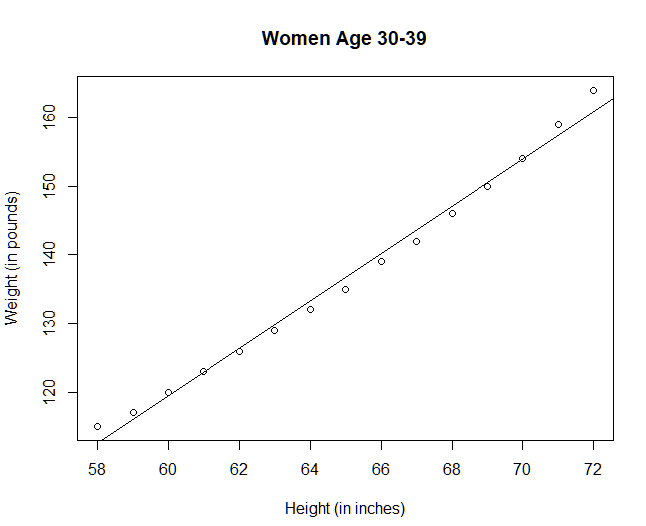
**> fitted(fit)**

**> residuals(fit)**

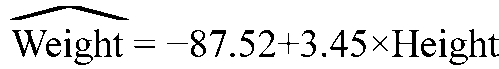
**>plot(women$height,women$weight,xlab="Height(in inches)",ylab="Weight (in pounds)") #绘制散点图**

**> abline(fit)#绘制拟合曲线**

**>lines(women$height,women$weight)#绘制拟合曲线，点点之间连线**



### 相关结论：

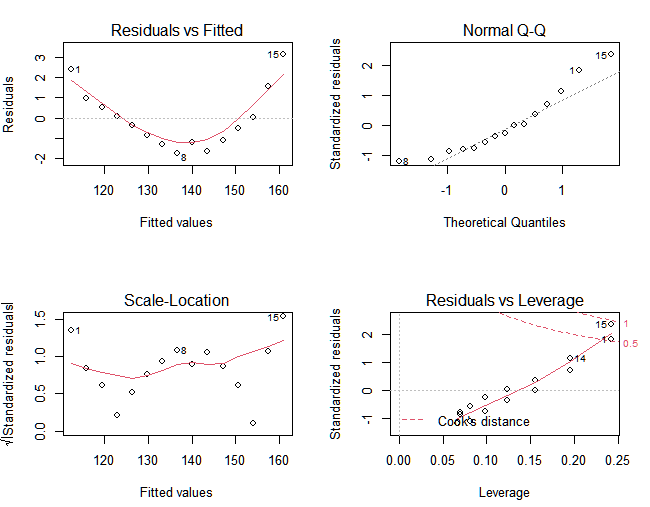
得到预测等式：

在Pr(>|t|)栏，可以看到回归系数（3.45）显著不为0（p<0.001），表明身高每增高1英寸，体重将预期增加3.45磅①。R平方项（0.991）表明模型可以解释体重99.1%的方差，它也是实际和预测值之间相关系数的平方（R2=r2ŶY）。残差标准误（1.53 lbs）则可认为是模型用身高预测体重的平均误差。

## 回归诊断：

**> par(mfrow=c(2,2))**

**> plot(fit)**



相关结论：

线性：在“残差图与拟合图”（Residuals vs Fitted，左上）中可以清楚地看到一个曲线关系，这暗示着可能需要对回归模型加上一个二次项。

正态性：“正态Q-Q图”（Normal Q-Q，右上）是在正态分布对应的值下，标准化残差的概率图。图上的点没有落在呈45度角的直线上，违反了正态性的假设。

同方差性：该图似乎满足水平线周围的点应该随机分布。

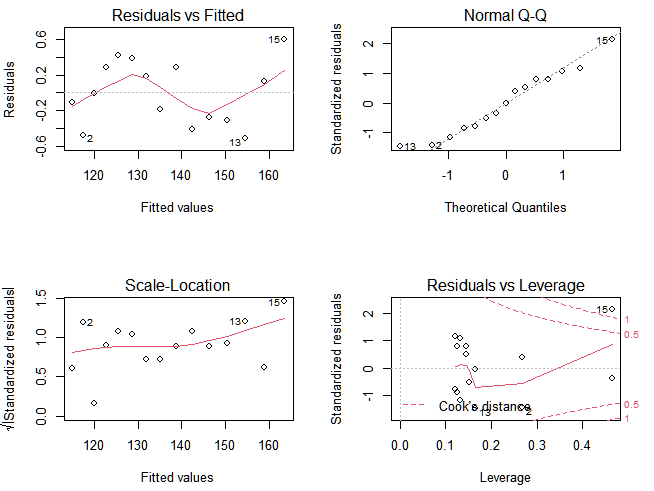
最后一幅“残差与杠杆图”（Residuals vs Leverage，右下）中，1和14为强影响点，删除拟合效果会更好

### 进行二次拟合：

**> fit2 <- lm(weight ~ height + I(height^2), data=women)**

**> par(mfrow=c(2,2))**

**> plot(fit2)**



### 相关结论：

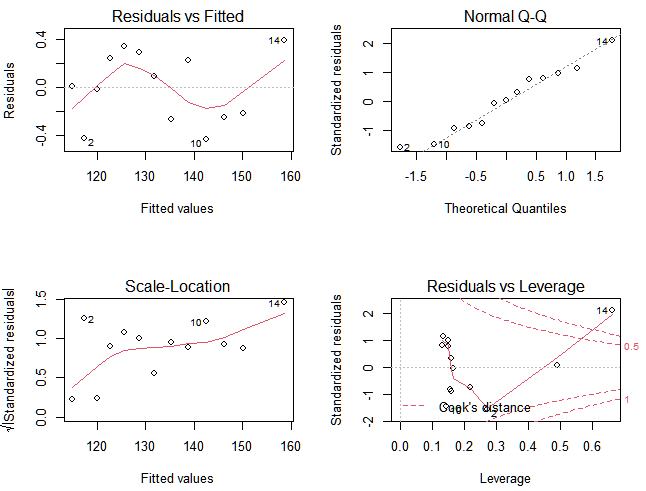
这第二组图表明多项式回归拟合效果比较理想，基本符合了线性假设、残差正态性（除了观测点13）和同方差性（残差方差不变）。观测点15看起来像是强影响点（根据是它有较大的Cook距离值），删除它将会影响参数的估计。事实上，删除观测点13和15，模型会拟合得会更好。

**删除13和15，再拟合：**

**> fit3<-lm(weight ~ height + I(height^2), data=women[-c(13,15),])#去除点13 15**

**> par(mfrow=c(2,2))**

**> plot(fit3)**



## 1.2多项式回归：

**> fit2 <- lm(weight ~ height + I(height^2), data=women)**

**> summary(fit2)**

Call:

lm(formula=weight ~ height + I(height^2), data=women)

Residuals:

Min 1Q Median 3Q Max

-0.5094 -0.2961 -0.0094 0.2862 0.5971

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) **261.87818** 25.19677 10.39 **2.4e-07** \*\*\*

Height **-7.34832** 0.77769 -9.45 **6.6e-07** \*\*\*

I(height^2) **0.08306**  0.00598 13.89 **9.3e-09** \*\*\*

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

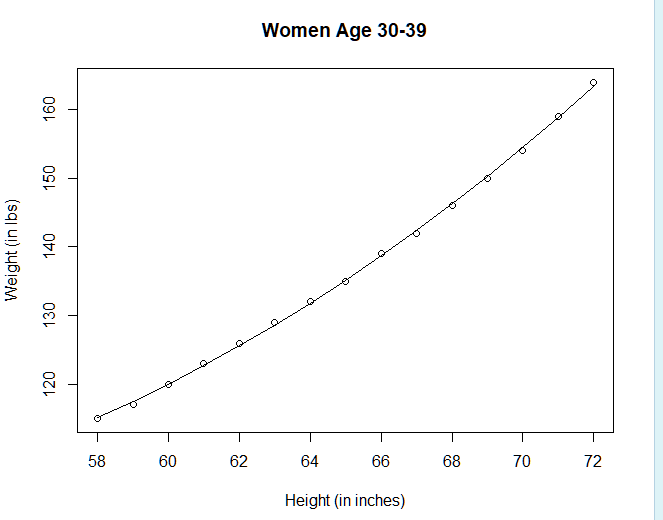
Residual standard error: 0.384 on 12 degrees of freedom

Multiple R-squared: **0.999**, Adjusted R-squared: 0.999

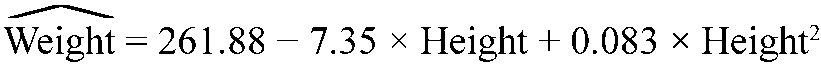
F-statistic: 1.14e+04 on 2 and 12 DF, p-value: **<2e-16**

**> plot(women$height,women$weight,xlab="Height(ininches)",ylab="Weight (in lbs)")**

**> lines(women$height,fitted(fit2))**



### 相关结论：

新的预测等式为：

在p<0.001水平下，回归系数都非常显著。模型的方差解释率已经增加到了99.9%。二次项的显著性（t=13.89，p<0.001）表明包含二次项提高了模型的拟合度。

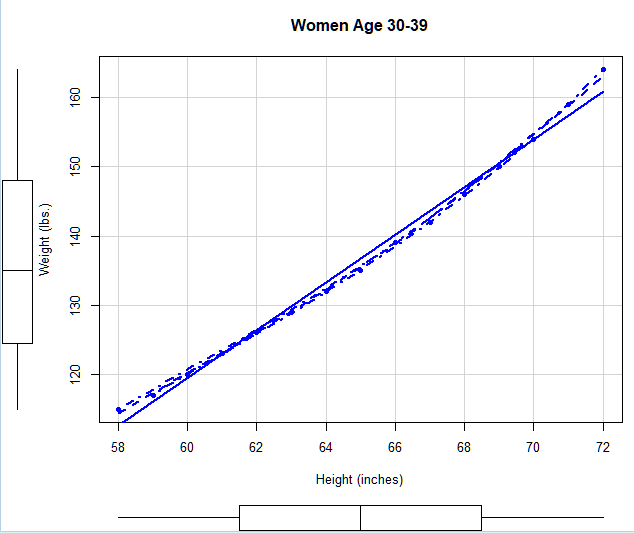
注意：虽然更高次的多项式也可用，但我发现使用比三次更高的项几乎没有必要。

#### 附加：

car包中的scatterplot()函数，它可以很容易、方便地绘制二元关系图。提供了身高与体重的散点图、线性拟合曲线和平滑拟合（loess）曲线，还在相应边界展示了每个变量的箱线图。

**library(car)**

**scatterplot(weight~height,data=women,spread=FALSE,smoother.args=list(lty=2), pch=19, main="Women Age 30-39", xlab="Height (inches)", ylab="Weight (lbs.)")**



spread=FALSE选项删除了残差正负均方根在平滑曲线上的展开和非对称信息。smoother.args=list(lty=2)选项设置loess拟合曲线为虚线。pch=19选项设置点为实心圆（默认为空心圆）。

## 1.3多元线性回归

以基础包中的state.x77数据集为例，我们想探究一个州的犯罪率和其他因素的关系，包括人口、文盲率、平均收入和结霜天数（温度在冰点以下的平均天数）。

多元回归分析中，第一步最好检查一下变量间的相关性。

**>states<-as.data.frame(state.x77[,c("Murder","Population","Illiteracy", "Income", "Frost")])**

**> cor(states)**

Murder Population Illiteracy Income 　Frost

Murder 　　1.00 　0.34 　　0.70 　　　-0.23 -0.54

Population 0.34 　1.00　　　0.11 　　　 0.21 　　-0.33

Illiteracy 0.70 　0.11　　　1.00 　　　 -0.44　　-0.67

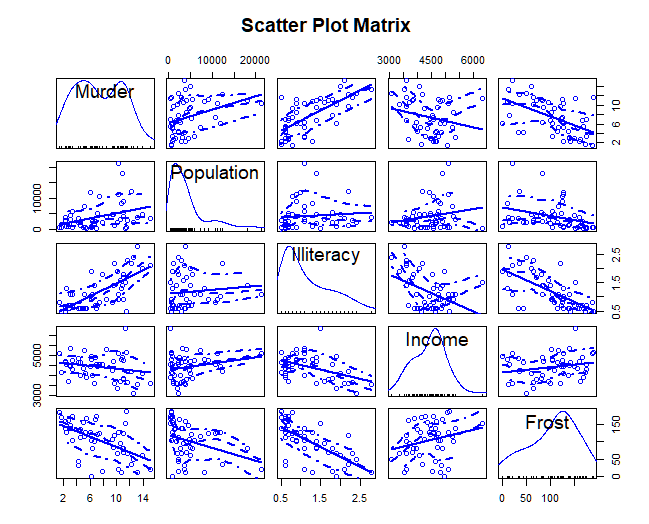
Income 　　-0.23　0.21　　　-0.44 　 1.00　　 0.23

Frost　　　-0.54　-0.33 　　-0.67 　　 0.23 　　1.00

**> library(car)**

**> scatterplotMatrix(states, spread=FALSE, smoother.args=list(lty=2),**

**main="Scatter Plot Matrix")**



从图中可以看到，谋杀率是双峰的曲线，每个预测变量都一定程度上出现了偏斜。谋杀率随着人口和文盲率的增加而增加，随着收入水平和结霜天数增加而下降。同时，越冷的州府文盲率越低，收入水平越高。

**建立多元回归：**

**> states <- as.data.frame(state.x77[,c("Murder", "Population",**

**"Illiteracy", "Income", "Frost")])**

**> fit <- lm(Murder ~ Population + Illiteracy + Income + Frost,**

**data=states)**

**> summary(fit)**

Call:

lm(formula=Murder ~ Population + Illiteracy + Income + Frost,

data=states)

Residuals:

Min 1Q Median 3Q Max

-4.7960 -1.6495 -0.0811 1.4815 7.6210

Coefficients:

Estimate Std.Error t value Pr(>|t|)

(Intercept) **1.23e+00** 3.87e+00 0.32 0.751

Population  **2.24e-04** 9.05e-05 2.47 0.017 \*

Illiteracy **4.14e+00** 8.74e-01 4.74 2.2e-05 \*\*\*

Income **6.44e-05** 6.84e-04 0.09 0.925

Frost **5.81e-04** 1.01e-02 0.06 0.954

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.v 0.1 'v' 1

Residual standard error: 2.5 on 45 degrees of freedom

Multiple R-squared: **0.567**, Adjusted R-squared: 0.528

1. statistic: 14.7 on 4 and 45 DF, p-value: **9.13e-08**

### 相关结论：

当预测变量不止一个时，回归系数的含义为：一个预测变量增加一个单位，其他预测变量保持不变时，因变量将要增加的数量。

例如本例中，人口的回归系数为2.24，表示控制文盲率、收入和温度不变时，人口上升1%，谋杀率将会上升4.14%，它的系数在p<0.001的水平下显著不为0；文盲率的回归系数为4.14，表示控制人口、收入和温度不变时，文盲率上升1%，谋杀率将会上升4.14%，它的系数在p<0.001的水平下显著不为0；相反，Income（p=0.925）、Frost（p=0.954）的系数没有显著为0，表明当控制其他变量不变时，Income与Murder不呈线性相关，Frost与Murder不呈线性相关。总体来看，所有的预测变量解释了各州谋杀率57%的方差。

注意：由于预测变量不止一个，所以无法绘制回归曲线和散点图

## 进行回归诊断：

**> states <- as.data.frame(state.x77[,c("Murder", "Population",**

**"Illiteracy", "Income", "Frost")])**

**> fit <- lm(Murder ~ Population + Illiteracy + Income + Frost, data=states)**

**> confint(fit)**

2.5 % 97.5 %

(Intercept) **-6.55e+00** 9.021318

Population **4.14e-05** 0.000406

Illiteracy **2.38e+00** 5.903874

Income **-1.31e-03** 0.001441

Frost **-1.97e-02** 0.020830

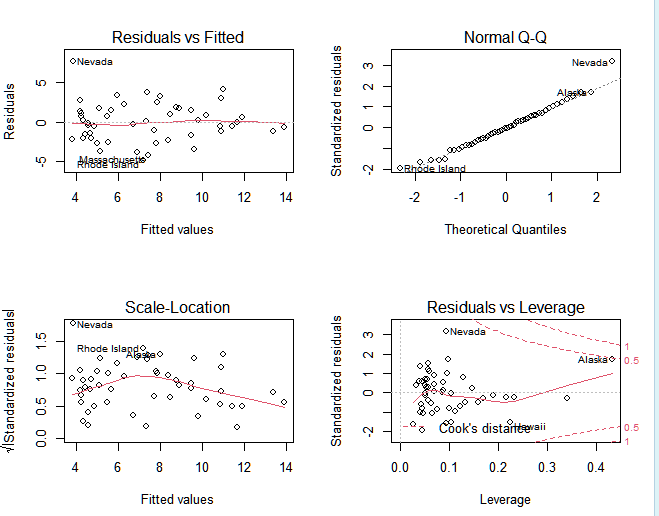
**> par(mfrow=c(2,2))**

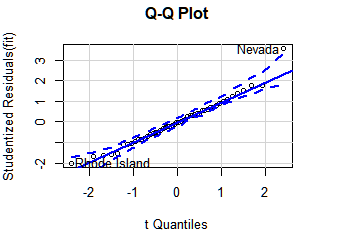
**> plot(fit)**

**> library(car)**

**> qqPlot(fit, labels=row.names(states), id.method="identify",**

**simulate=TRUE, main="Q-Q Plot")**





在n–p–1个自由度的t分布下的学生化残差（studentized residual，也称学生化删除残差或折叠化残差）图形，其中n是样本大小，p是回归参数的数目（包括截距项）。

### 相关结论：

结果表明，文盲率改变1%，谋杀率就在95%的置信区间[2.38, 5.90]中变化；另外，因为Income、Frost的置信区间包含0，所以可以得出结论：当其他变量不变时，收入、温度的改变与谋杀率无关。不过，这些结果的信念，都只建立在你的数据满足统计假设的前提之上。

除去Nevada一个离群点，模型假设得到了很好的满足。Alaska是强影响点。

id.method = "identify"选项能够交互式绘图——待图形绘制后，用鼠标单击图形内的点，将会标注函数中labels选项的设定值。敲击Esc键，从图形下拉菜单中选择Stop，或者在图形上右击，都将关闭这种交互模式。此处，我已经鉴定出了Nevada异常。当simulate=TRUE时，95%的置信区间将会用参数自助法生成。

鉴定出了Nevada异常，超出置信区间。除了Nevada，所有的点都离直线很近，并都落在置信区间内，这表明正态性假设符合得很好。但是也必须关Nevada，它有一个很大的正残差值（真实值－预测值），表明模型低估了该州的谋杀率。

**> states["Nevada",]**

Murder Population Illiteracy Income Frost

Nevada 11.5 590 0.5 5149 188

**> fitted(fit)["Nevada"]**

Nevada

3.878958

**> residuals(fit)["Nevada"]**

Nevada

7.621042

**> rstudent(fit)["Nevada"]**

Nevada

3.542929

Nevada的谋杀率是11.5%，而模型预测的谋杀率为3.9%。

## 有交互项的多元线性回归:

**> fit <- lm(mpg ~ hp + wt + hp:wt, data=mtcars)**

**> summary(fit)**

Call:

lm(formula=mpg ~ hp + wt + hp:wt, data=mtcars)

Residuals:

Min 1Q Median 3Q Max

-3.063 -1.649 -0.736 1.421 4.551

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) **49.80842** 3.60516 13.82 5.0e-14 \*\*\*

hp  **-0.12010**  0.02470 -4.86 4.0e-05 \*\*\*

wt **-8.21662** 1.26971 -6.47 5.2e-07 \*\*\*

hp:wt  **0.02785** 0.00742 3.75 0.00081 \*\*\*

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

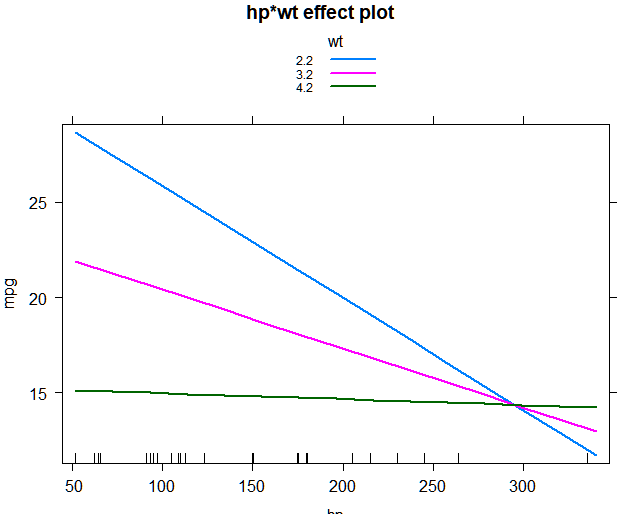
Residual standard error: 2.1 on 28 degrees of freedom

Multiple R-squared: **0.885**, Adjusted R-squared: 0.872

F-statistic: 71.7 on 3 and 28 DF, p-value: **2.98e-13**

**> library(effects)**

**> plot(effect("hp:wt", fit,, list(wt=c(2.2,3.2,4.2))), multiline=TRUE)**



term即模型要画的项，mod为通过lm()拟合的模型，xlevels是一个列表，指定变量要设定的常量值，multiline=TRUE选项表示添加相应直线。

### 相关结论：

你可以看到Pr(>|t|)栏中，马力与车重的交互项是显著的。若两个预测变量的交互项显著，说明响应变量与其中一个预测变量的关系依赖于另外一个预测变量的水平。因此此例说明，每加仑汽油行驶英里数与汽车马力的关系依车重不同而不同。

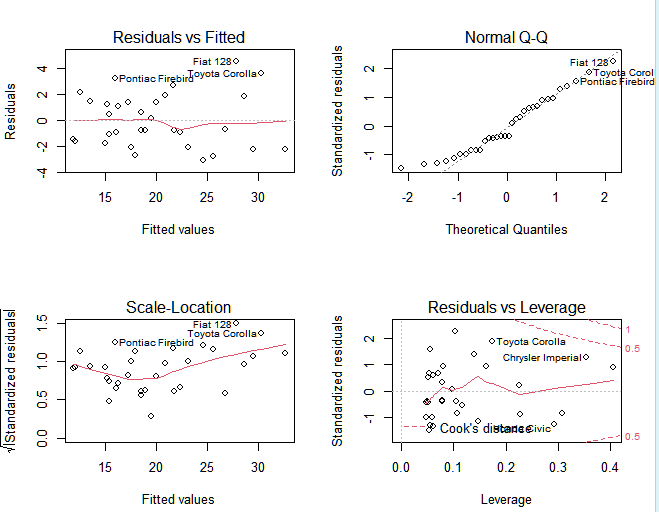
预测mpg的模型为=49.81–0.12×hp–8.22×wt+0.03×hp×wt。为更好地理解交互项，你可以赋给wt不同的值，并简化等式。例如，可以试试wt的均值（3.2），少于均值一个标准差和多于均值一个标准差的值（分别是 2.2 和 4.2 ）。 若 wt=2.2 ，则等式可以化简为=49.81–0.12×hp–8.22×(2.2) + 0.03×hp×(2.2) =31.41–0.06×hp；若wt=3.2，则变成了=23.37–0.03×hp；若wt=4.2，则等式为=15.33–0.003×hp。你将发现，随着车重增加（2.2、3.2、4.2），hp每增加一个单位引起的mpg预期改变却在减少（0.06、0.03、0.003）。

从图中可以很清晰地看出，随着车重的增加，马力与每加仑汽油行驶英里数的关系减弱了。当wt=4.2时，直线几乎是水平的，表明随着hp的增加，mpg不会发生改变。然而，拟合模型只不过是分析的第一步，一旦拟合了回归模型，在信心十足地进行推断之前，必须对方法中暗含的统计假设进行检验。

## 进行回归诊断：

**> par(mfrow=c(2,2))**

**> plot(fit)**



# 判别分析

## 步骤：

1. 建立一次判别函数
2. 预测训练样本： （1）读取预测变量

（2）输出直方图，密度概率图（一维，二维）

1. 查看预测后的训练样本分组
2. 混淆矩阵，查看错误率（因变量和类别）
3. 预测新样本
4. 建立二次判别函数
5. 预测训练样本
6. 查看预测后的训练样本分组
7. 混淆矩阵，查看错误率
8. 预测新样本

## 求值：

library(mass)

(ld1<-lda(g~x1+x2+x3+x4,data=w, prior=c(0.3,0.7)))#贝叶斯一次判别分析

MASS::lda(g~x1+x2+x3+x4,data=w, prior=c(0.3,0.7))#无需加载包，进行判别分析

(ld2<-lda(g~., data=w))#Fisher一次判别

(qd1<-qda(g~., data=w, prior=c(0.2,0.8)))#贝叶斯二次判别分析

(qd2<-qda(g~.,data=w))#Fisher二次判别

p.ld1<-predict(ld1)#预测训练样本

names(p.ld1)#查看预测样本的变量

p.ld1$class#查看样本分组

table(w$g, p.ld1$class)#混淆矩阵，查看错误率

p.ld2<-predict(ld1,newdata=data.frame(x1=-0.23, x2=0.02,x3=1.56,x4=0.60))#预测新样本

p.qd2$posterior#后验概率

ld2$scaling#求特征向量

(er=sum(w$g!=p.ld2$class)/dim(v)[1])#计算误判率

## 画图：

plot(ld1)

plot(“ld1”,type=“density”,dimen=1)#一维概率密度图

plot(“ld1”,dimen=2)#二维概率密度图

points(y1$x, col="red")#图上画标记点

## 贝叶斯判别

### 贝叶斯一次判别

**>library(MASS)**

**>(ld1<-lda(g~x1+x2+x3+x4,data=w, prior=c(0.3,0.7)))#判别分析**

**>MASS::lda(g~x1+x2+x3+x4,data=w, prior=c(0.3,0.7))#无需加载包，进行判别分析**

# Call:

# lda(g ~ x1 + x2 + x3 + x4, data = w, prior = c(0.3, 0.7))

#

# Prior probabilities of groups:（先验概率）

# 1 2

# 0.3 0.7

#

# Group means:

# x1 x2 x3 x4

# 1 -0.06904762 -0.08142857 1.366667 0.437619

# 2 0.23520000 0.05560000 2.593600 0.426800

#

# Coefficients of linear discriminants:（线性判别系数）

# LD1

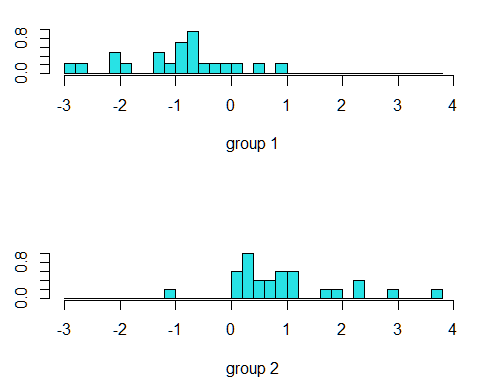
# x1 0.6612498

# x2 4.3935204

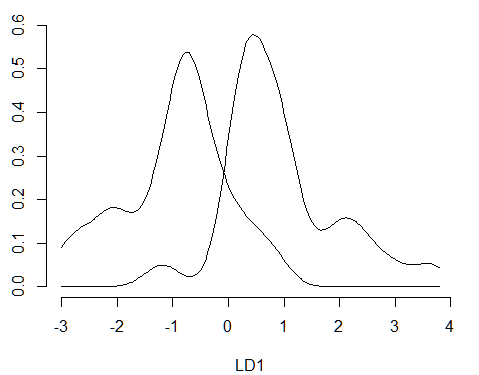
# x3 0.8872152

# x4 -1.1785089

**>plot(ld1)#两类样本经过线性变换后的直方图**



**>plot(ld1, type="density", dimen=1)#两类样本经过线性变换后的密度图**



**>p.ld1<-predict(ld1)#预测训练样本**

**>names(p.ld1)**

# "class" "posterior" "x"

**>p.ld1$class#查看预测分组**

# [1] 1 1 1 1 1 1 1 1 2 1 1 2 2 1 2 2 1 1 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2

# Levels: 1 2

**>table(w$g, p.ld1$class)#混淆矩阵**

# 1 2

# 1 15 6

# 2 1 24

**>predict(ld1, newdata=data.frame(x1=-0.23, x2=0.02,x3=1.56,x4=0.60))#预测新样本**

# $class

# [1] 1

# Levels: 1 2

#

# $posterior（后验概率，属于第k类的概率）

# 1（属于第一类） 2（属于第二类）

# 1 0.737115 0.262885

#

# $x（预测之后新样本x的x'a->ykba）

# LD1

# 1 -0.7157087

### 相关结论：

混淆矩阵中应属于样本1实属于样本1有15个，应属于样本1实属于样本2有6个，应属于样本2实属于样本1有1个,应属于样本2实属于样本2有24个。新样本属于样本1。

### 贝叶斯二次判别

**>library(MASS)**

**>(qd1<-qda(g~., data=w, prior=c(0.2,0.8)))#二次判别**

**>p.qd1<-predict(qd1)**

**>head(p.qd1$class)**

# [1] 1 1 2 1 1 1

# Levels: 1 2

**>table(w$g, p.qd1$class)#混淆矩阵**

# 1 2

# 1 16 5

# 2 0 25

**>newdata=data.frame(x1=-0.23, x2=0.02,x3=1.56,x4=0.60)**

**>predict(qd1, newdata = newdata)#对新的值进行预测**

# $class

# [1] 2

# Levels: 1 2

#

# $posterior

# 1 2

# 1 0.00236558 0.9976344

### 相关结论：

混淆矩阵中应属于样本1实属于样本1有16个，应属于样本1实属于样本2有5个，应属于样本2实属于样本1有0个,应属于样本2实属于样本2有25个。新样本属于样本2。

## Fisher判别

### Fisher一次判别

**>(ld2<-lda(g~., data=w))**

# Call:

# lda(g ~ ., data = w)

#

# Prior probabilities of groups:

# 1 2

# 0.4565217 0.5434783

#

# Group means:

# x1 x2 x3 x4

# 1 -0.06904762 -0.08142857 1.366667 0.437619

# 2 0.23520000 0.05560000 2.593600 0.426800

#

# Coefficients of linear discriminants:

# LD1

# x1 0.6612498

# x2 4.3935204

# x3 0.8872152

# x4 -1.1785089

**>names(ld2)**

# [1] "prior" "counts" "means" "scaling" "lev" "svd" "N" "call" "terms" "xlevels"

**>ld2$scaling#(特征向量a1)**

# LD1

# x1 0.6612498

# x2 4.3935204

# x3 0.8872152

# x4 -1.1785089

**>p.ld2<-predict(ld2)**

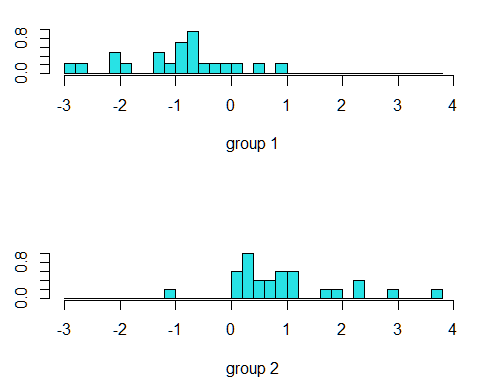
**>table(w$g,p.ld2$class)**

# 1 2

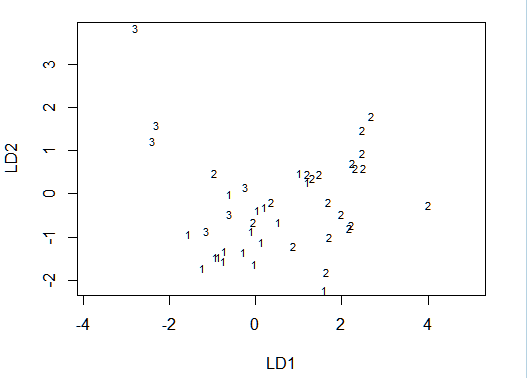
# 1 18 3

# 2 1 24

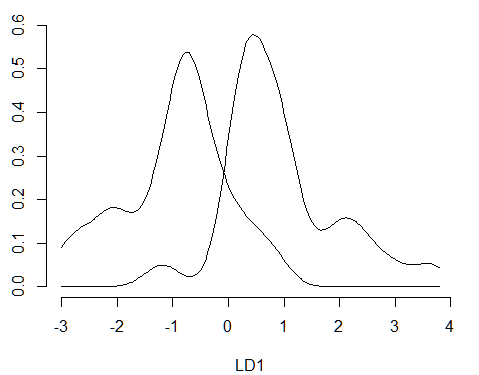
**>plot(ld2,dimen =1)#第一判别函数变换后的直方图**



**>plot(ld2,dimen =2)#第一判别函数变换后的散点图（二维以上）**



**>plot(ld2, type="density",dimen = 1)#第一判别函数变换后的密度函数**



**>plot(ld2,dimen=2)#二维图->两个特征向量**

**>newdata=data.frame(x1=-0.23, x2=0.02,x3=1.56,x4=0.60)**

**>(y1=predict(ld2, newdata=newdata))#新样本预测**

# $class

# [1] 1

# Levels: 1 2

#

# $posterior

# 1 2

# 1 0.737115 0.262885

#

# $x

# LD1

# 1 -0.7157087

**>plot(ld2,dimen=2)#二维图->两个特征向量**

**>points(y1$x, col="red")#二维图**

### 相关结论：

混淆矩阵中应属于样本1实属于样本1有18个，应属于样本1实属于样本2有3个，应属于样本2实属于样本1有1个,应属于样本2实属于样本2有24个。新样本属于样本1。

### Fisher二次判别

**>(qd2<-qda(g~.,data=w))#进行二次判别**

# Call:

# qda(g ~ ., data = w)

#

# Prior probabilities of groups:

# 1 2

# 0.4565217 0.5434783

#

# Group means:

# x1 x2 x3 x4

# 1 -0.06904762 -0.08142857 1.366667 0.437619

# 2 0.23520000 0.05560000 2.593600 0.426800

**>p.qd2<-predict(qd2)**

**>table(w$g, p.qd2$class)**

# 1 2

# 1 19 2

# 2 1 24

**>names(p.qd2)**

# [1] "class" "posterior"

**vhead(p.qd2$class)**

**>head(p.qd2$posterior)#新样本在各个类别内的后验概率**

**vp.qd2$posterior#**

**>newdata=data.frame(x1=-0.23, x2=0.02,x3=1.56,x4=0.60)**

**>(y1=predict(qd2, newdata = newdata))#利用二次判别函数进行预测**

# $class

# [1] 2

# Levels: 1 2

#

# $posterior

# 1 2

# 1 0.007904223 0.9920958

**>y1$class#新样本的判别类别**

# [1] 2

# Levels: 1 2

**>y1$posterior#新样本在各个类别内的后验概率**

# 1 2

# 1 0.007904223 0.9920958

## Logistic回归

**>q=read.csv("C:/Users/86133/Desktop/《多元统计分析——R与Python的实现》数据文件/MVAPureData/Trans.csv")**

**>a=glm(Donate~.,q,family=binomial)#logistic拟合**

**>summary(a)**

# Call:

# glm(formula = Donate ~ ., family = binomial, data = q)

#

# Deviance Residuals:

# Min 1Q Median 3Q Max

# -2.4875 -0.7933 -0.4997 -0.1701 2.6450

#

# Coefficients:

# Estimate Std.Error z value Pr(>|z|)

# (Intercept) -0.449540 0.180349 -2.493 0.012681 \*

# Recency -0.098584 0.017317 -5.693 1.25e-08 \*\*\*

# Frequency 0.135390 0.025672 5.274 1.34e-07 \*\*\*

# Time -0.023092 0.005964 -3.872 0.000108 \*\*\*

# ---

# Signif. codes: 0 ‘\*\*\*’0.001‘\*\*’0.01‘\*’0.05‘.’0.1‘ ’1

#

# (Dispersion parameter for binomial family taken to be 1)

#

# Null deviance: 820.89 on 747 degrees of freedom

# Residual deviance: 707.87 on 744 degrees of freedom

# AIC: 715.87

#

# Number of Fisher Scoring iterations: 5

**>pred<-predict(a)**

**>table(q$Donate,(pred>0.5)\*1)**

**>sum(q$donate!=(pred>0.5\*1)/nrow(q))**

若阈值不合适，启用以下程序自动选择阈值，使得训练集误判率最小

**>BI=function(D,q,ff,fm="binomial"){**

**a=glm(ff,q,family=fm)**

**z=predict(a,q,type="response")**

**L=max(levels(q[,D]))**

**ee=NULL**

**for(p in seq(.01,.99, .01)){**

**u=rep(L,nrow(q));u[!(z>p)]=min(levels(q[,D]))**

**e=sum(u!=q[,D])/nrow(q);ee=rbind(ee,c(p,e))}**

**I=which(ee[,2]==min(ee[,2]))**

**return(ee[min(I),])**

**}#该程序自动从0.01到0.99以间隔0.01的步长搜索使得误判率最小的Pt**

**>ff=Donate~. #公式**

**>q$Donate=factor(q$Donate)#该函数要求把因变量因子化**

**>BI(D=4,q,ff)#调用程序**

# [1] 0.4400000 0.2072193

### 相关结论：

=-0.449540，=-0.098584，=0.135390，=-0.023092

阈值为0.5时，误差率为很大；阈值为0.44时，误判率为0.207

#ROC等描述性曲线

**>library(caTools)**

**vset.seed(1010)**

**>D=4**

**>split<-sample.split(q[,D],SplitRatio=0.75)#训练集占75%，得到一个训练集和一个测试集**

**>q\_train<-subset(q,split == TRUE)**

**>q\_test<-subset(q,split == FALSE)**

#Logistic回归

**>a<-glm(Donate~.,data=q\_train,family = binomial)**

**>pred<-predict(a,q\_test,type='response')**

#四个点图

**>library(ROCR)**

**>par(mfrow-c(1,4),mar=c(4,4,4,2))**

**>ROCRpred<-prediction(pred,q\_test$Donate)**

**>ROCRperf<-performance(ROCRpred, 'tpr','fpr')**

**>par(mfrow=c(2,2))**

**>plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))**

**>title("ROC curve")**

**>precperf<-performance(ROCRpred,'prec','rec')**

**>plot (Precperf, colorize=TRUE, text.adj = c(-0.2,1.7))**

**>title("Precision/recall graph")**

**>Sensperf<-performance(ROCRpred,'sens','spec')**

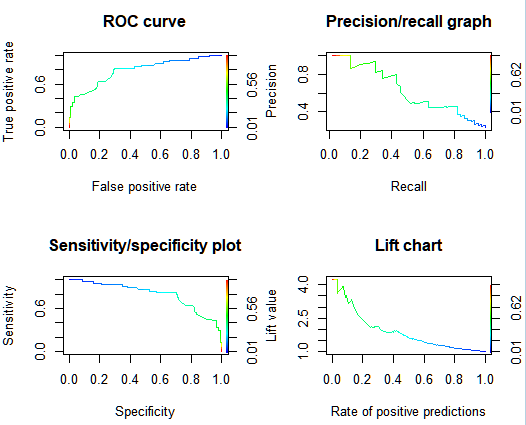
**>plot (Sensperf, colorize = TRUE, text.adj = c(-0.2,1.7))**

**>title("Sensitivity/specificity plot")**

**>Liftperf <- performance (ROCRpred, 'lift' ,'rpp')**

**>plot (Liftperf, colorize = TRUE, text.adj = c(-0.2,1.7))**

**>title("Lift chart")**



#可以得到ROC曲线下面的面积:

**>pr<-prediction(pred, q\_test$Donate)**

**>auc<-performance(ROCRpred,measure="auc")**

**>auc<-auc@y.values [[1]]**

**>auc**

# [1] 0.793774

### 相关结论：

AUC=0.793774

# 主成分分析

## 步骤：

1. 作主成分分析
2. 计算均值方差等
3. 特征向量

2.求相关系数阵

3.求相关系数的特征值和特征向量

4.计算贡献率和累计贡献率

5.绘制贡献率图（崖底碎石图）和累计贡献率图

6.计算载荷并绘制载荷图

7.计算主成分得分并绘制主成分得分图

8.挑选出第一主成分，第二主成分等

9.针对第一主成分，第二主成分等进行排序

10.显示排在前几的数据

11.绘制第一主成分与第二主成分的散点图

## 相关函数代码：

1.#相关系数图

corrplot::corrplot(cor(u[,-1]),type="lower",diag=FALSE,t1.col="black",t1.srt=45)2.rgb.palette<-colorRampPalette(c("white","black"),space="rgb")

levelplot(cor(u[,-1]),last=1,main="Correlationmatrix",xlab="",ylab="",col.regions=rgb.palette(120),cuts=100,at=seq(0,1,0.01))

GGally::ggpairs(x1)

3.eigen(a)#求特征值和特征向量

summary(x1.pr,loadings=TRUE)

4.(cca=(a1$va)/sum(a1$va))#贡献率

5.(ca=cumsum(a1$va)/sum(a1$va))#累积贡献率

6.#绘制崖底碎石图

plot(1:(ncol(u)-1),a1$va,type="o",pch=17,col=4,

main="Scree Plot",xlab="Component Number",ylab="Eigen Value")

fviz\_eig(x1.pr,addlabels =TRUE)

screeplot(x1.pr,type="barplot",main="崖底碎石图")

screeplot(x1.pr,type="lines",main="崖底碎石图")

7.#累计贡献率图

plot(1:(ncol(u)-1),ca,type="o",pch=17,col=4)

8.(a2=sweep(a1$vec[,a1$va>0],2,sqrt(a1$value[a1$va>0]),"\*"))#求相关系数(公式—样本方差出发)

9.order(prc1, decreasing = T)#第一主成分得分按照降序排列的学生学号

1. #作主成分分析

x1.pr<-princomp(x1,cor=TRUE)

(race.pr<-prcomp(w[,-c(1,2)], center = T, scale. = T))#对数据作标准化，相当于用相关矩阵做主成分分析

12.#计算主成分得分

x1.p<-predict(x1.pr)

sc=scale(u[,-1])%\*%a1$ve[,1:2]

(prc1<-predict(x1.pr)[,1])#第一主成分得分

13.#绘制主成分得分图

plot(sc[,1],sc[,2],type="n",ylim=c(-6,10),xlim=c(-11,7),

main="Sample Principal Components",xlab="Component 1",ylab="Component 2")

text(sc[,1],sc[,2],u[,1],cex=.4);

abline(v=0,col=2);abline(h=0,co1=2)

14.#绘制载荷图

pct=paste("(",round(a1$va/sum(a1$va)\*100),"%)",sep="")

tt=NULL;for(i in 1:4)tt=c(tt,paste("Component", i,pct[i]))

plot(a2[,1:2], type="n", xlab=tt[1], ylab=tt[2],main="Loadings",xlim=c(-1,0.05))

text(a2[,1],a2[,2],names(u[,-1]));abline(h=0);abline(v=0)#标注点

a1<-race.pr$rotation[,1]\*race.pr$sdev[1]

a2<-race.pr$rotation[,2]\*race.pr$sdev[2]

plot(a1, a2, type="n", main="以相关系数作载荷图")

abline(h=0)#画横线

text(a1, a2, labels=rownames(race.pr$rotation))

plot(z1$ve[,1],z2$ve[,2],type="n",main="载荷图")

abline(h=0,col=2)

text(z1$ve[,1],z2$ve[,2],labels=row.names(z1))

15.biplot(student.pr)#样本的第一主成分得分和第二主成分得分的散点图

### 教师数据分析

**>a=read.csv("C:/Users/86133/Desktop/《多元统计分析——R与Python的实现》数据文件/MVAPureData/full.aaup.csv" )**

**>dim(a)**

**>names(a)**

**>summary(a)**

# FICE College State Type ASF ASA1

# Min. : 1002 Length:1161 Length:1161 Length:1161 Min. : 270.0 Min. :234.0

# 1st Qu.: 1903 Class :character Class :character Class :character 1st Qu.: 433.0 1st Qu.:364.0

# Median : 2668 Mode :character Mode :character Mode :character Median : 499.0 Median :410.0

# Mean : 3052 Mean : 516.5 Mean :414.3

# 3rd Qu.: 3420 3rd Qu.: 594.0 3rd Qu.:460.0

# Max. :29269 Max. :1009.0 Max. :733.0

# ASA2 ASALL ACF ACA1 ACA2 ACALL NF

# Min. :199 Min. :232.0 Min. : 319.0 Min. :292 Min. :246.0 Min. : 265.0 Min. : 0.0

# 1st Qu.:312 1st Qu.:352.0 1st Qu.: 534.0 1st Qu.:452 1st Qu.:386.0 1st Qu.: 436.0 1st Qu.: 18.0

# Median :348 Median :407.0 Median : 627.0 Median :517 Median :435.0 Median : 510.0 Median : 40.0

# Mean :351 Mean :420.4 Mean : 643.5 Mean :521 Mean :440.8 Mean : 526.7 Mean : 95.1

# 3rd Qu.:387 3rd Qu.:475.0 3rd Qu.: 741.0 3rd Qu.:580 3rd Qu.:491.0 3rd Qu.: 600.0 3rd Qu.:105.0

# Max. :576 Max. :866.0 Max. :1236.0 Max. :909 Max. :717.0 Max. :1075.0 Max. :997.0

# NA1 NA2 NIN NALL

# Min. : 0.00 Min. : 0.00 Min. : 0.00 Min. : 7.0

# 1st Qu.: 19.00 1st Qu.: 21.00 1st Qu.: 2.00 1st Qu.: 68.0

# Median : 38.00 Median : 40.00 Median : 6.00 Median : 132.0

# Mean : 72.39 Mean : 68.63 Mean : 12.74 Mean : 257.4

# 3rd Qu.: 89.00 3rd Qu.: 92.00 3rd Qu.: 14.00 3rd Qu.: 323.0

# Max. :721.00 Max. :510.00 Max. :178.00 Max. :2261.0

**>u=a[,-c(1,3,4)] #去掉无关定性变量(仅保存学校名称)**

**>round(cor(u[,-1]),digits=2)#求相关系数阵，保留小数点后两位（总体方差）**

# ASF ASA1 ASA2 ASALL ACF ACA1 ACA2 ACALL NF NA1 NA2 NIN NALL

# ASF 1.00 0.95 0.92 0.97 0.99 0.93 0.90 0.96 0.58 0.56 0.55 0.15 0.58

# ASA1 0.95 1.00 0.94 0.94 0.94 0.98 0.93 0.94 0.49 0.50 0.48 0.14 0.50

# ASA2 0.92 0.94 1.00 0.93 0.91 0.93 0.97 0.92 0.53 0.53 0.51 0.15 0.54

# ASALL 0.97 0.94 0.93 1.00 0.96 0.93 0.92 0.99 0.61 0.54 0.51 0.08 0.57

# ACF 0.99 0.94 0.91 0.96 1.00 0.95 0.92 0.97 0.55 0.54 0.52 0.13 0.55

# ACA1 0.93 0.98 0.93 0.93 0.95 1.00 0.95 0.95 0.47 0.47 0.46 0.11 0.48

# ACA2 0.90 0.93 0.97 0.92 0.92 0.95 1.00 0.94 0.52 0.52 0.50 0.14 0.52

# ACALL 0.96 0.94 0.92 0.99 0.97 0.95 0.94 1.00 0.58 0.52 0.49 0.07 0.55

# NF 0.58 0.49 0.53 0.61 0.55 0.47 0.52 0.58 1.00 0.89 0.86 0.36 0.96

# NA1 0.56 0.50 0.53 0.54 0.54 0.47 0.52 0.52 0.89 1.00 0.92 0.45 0.96

# NA2 0.55 0.48 0.51 0.51 0.52 0.46 0.50 0.49 0.86 0.92 1.00 0.54 0.95

# NIN 0.15 0.14 0.15 0.08 0.13 0.11 0.14 0.07 0.36 0.45 0.54 1.00 0.48

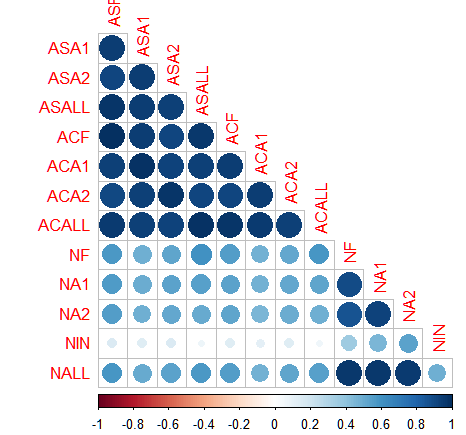
# NALL 0.58 0.50 0.54 0.57 0.55 0.48 0.52 0.55 0.96 0.96 0.95 0.48 1.00

### 相关结论：

上面输出表明,以字母“A”开头的和收入有关的8个变量之间的相关系数比较大,都在0.9以上;以“N”开头的和人数有关的后面5个变量之间的相关系数(除了变量 NIN有些“另类”之外)也不小,但不如前8个变量那么相关;而前8个变量与后5个变量之间的相关系数最大只有0.606,其余则小于0.6.

**>library(corrplot)**

**>corrplot::corrplot(cor(u[,-1]),type="lower",diag=FALSE,t1.col="black",t1.srt=45)**

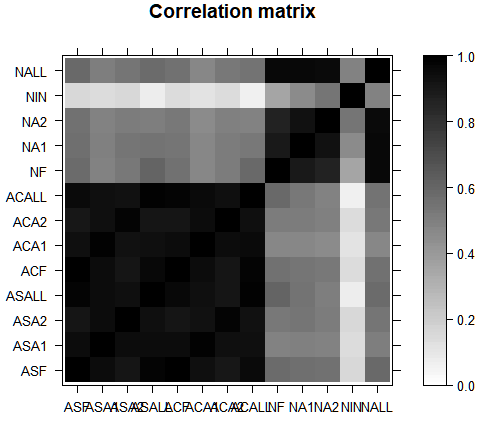


**>library(lattice)**

**>rgb.palette<-colorRampPalette(c("white","black"),space="rgb")**

**>levelplot(cor(u[,-1]),last=1,main="Correlation matrix",xlab="",ylab="",**

**col.regions=rgb.palette(120),cuts=100,at=seq(0,1,0.01))**



**>a1=eigen(cor(u[,-1]))#求特征值和特征向量**

**>data.frame(a1$ce,row.names=names(u)[-1])[,1:4]#带有变量名的四个特征向量**

**a1**

# eigen() decomposition

# $values

# [1] 9.3052882802 2.4399106790 0.7060923174 0.1801957775 0.1366777593 0.0762450750 0.0667250540 0.0521300766

# [9] 0.0270207415 0.0049802461 0.0025604214 0.0019002617 0.0002733104

#

# $vectors

# [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]

# [1,] -0.31273824 -0.1352035 0.02804996 -0.30274173 0.32643955 0.28192136 0.17484613 -0.159873273 0.33476995

# [2,] -0.30502296 -0.1891519 0.11642372 0.11576999 0.18700923 -0.35740015 -0.32418154 -0.491220053 0.04624502

# [3,] -0.30568127 -0.1559534 0.08826369 0.41070451 -0.30616883 0.37246374 0.13247771 -0.422042021 0.07974184

# [4,] -0.31187011 -0.1573095 -0.07045888 -0.30599752 -0.10684707 0.09330105 0.07321106 -0.251561784 -0.54754026

# [5,] -0.31098143 -0.1579576 0.03176836 -0.27621024 0.28963441 0.14171432 0.17656911 0.314945577 0.35147911

# [6,] -0.30214696 -0.2076846 0.11074421 0.13769700 0.11603561 -0.52250450 -0.31369581 0.233562800 0.08124207

# [7,] -0.30382362 -0.1683954 0.08791390 0.42882607 -0.37563696 0.11901324 0.06832069 0.455775334 0.11333917

# [8,] -0.31028804 -0.1751978 -0.05822265 -0.25605919 -0.14015905 -0.06470867 0.06353627 0.314917356 -0.48301519

# [9,] -0.24334906 0.3569358 -0.33116826 -0.33087167 -0.49060183 -0.08159424 -0.27164234 -0.059909963 0.26598977

# [10,] -0.24106277 0.3936994 -0.17830881 0.26108295 0.19783336 -0.38558589 0.65940894 -0.062524647 -0.10773403

# [11,] -0.23529148 0.4123763 -0.04912807 0.26515144 0.43521666 0.41989608 -0.43309280 0.152933841 -0.29736676

# [12,] -0.08890082 0.3897086 0.87898802 -0.19914972 -0.14464824 -0.03828567 0.05486419 -0.002737863 -0.03729913

# [13,] -0.24752096 0.4067060 -0.17891781 -0.01775209 -0.07102274 -0.02816521 -0.06284513 -0.014121317 0.17654660

# [,10] [,11] [,12] [,13]

# [1,] 0.397052607 -0.011721132 -0.453639003 0.271394494

# [2,] 0.252128315 -0.016162227 0.498477780 0.143800738

# [3,] -0.495790485 -0.009285338 -0.102229544 0.122173471

# [4,] 0.127550574 0.041559336 -0.094667107 -0.603215307

# [5,] -0.381051498 -0.079858378 0.462551377 -0.278623854

# [6,] -0.276702677 0.000540541 -0.535702739 -0.157401914

# [7,] 0.525297637 -0.025158869 0.111726324 -0.134715621

# [8,] -0.134880188 0.107107153 0.108070970 0.634335634

# [9,] -0.013937711 -0.444300576 -0.010753221 0.032242523

# [10,] 0.018083131 -0.219693402 -0.009647609 0.009764184

# [11,] -0.013023806 -0.190800929 -0.006202769 0.011849555

# [12,] -0.003306703 -0.033590645 -0.002114228 0.001361278

# [13,] -0.010319094 0.834328424 0.031206692 -0.051551129

**>(cca=(a1$va)/sum(a1$va))#贡献率**

# [1] 7.157914e-01 1.876854e-01 5.431479e-02 1.386121e-02 1.051367e-02 5.865006e-03 5.132696e-03 4.010006e-03

# [9] 2.078519e-03 3.830959e-04 1.969555e-04 1.461740e-04 2.102388e-05

**>(ca=cumsum(a1$va)/sum(a1$va))#累积贡献率**

# [1] 0.7157914 0.9034768 0.9577916 0.9716529 0.9821665 0.9880315 0.9931642 0.9971742 0.9992528 0.9996358 0.9998328

# [12] 0.9999790 1.0000000

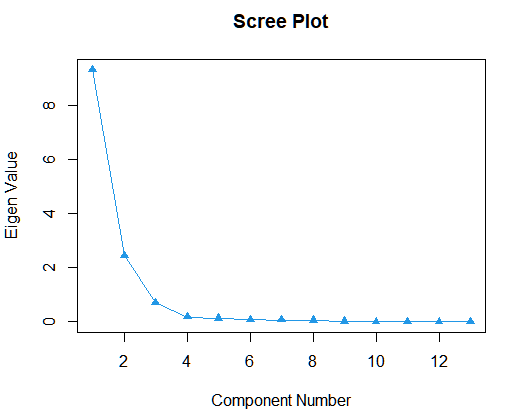
### 相关结论：

输出表明,贡献率最大的前两个主成分的贡献率之和(即累积贡献率)已经超过90%了

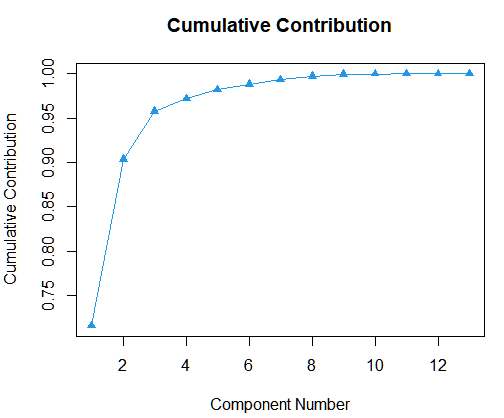
**#崖底碎石图及累计贡献率图**

>**plot(1:(ncol(u)-1),a1$va,type="o",pch=17,col=4,**

**main="Scree Plot",xlab="Component Number",ylab="Eigen Value")#崖底碎石图**



**>plot(1:(ncol(u)-1),ca,type="o",pch=17,col=4,main="CumulativeContribution",xlab="ComponentNumber",ylab="Cumulative Contribution")#累计贡献率图**



**>a1$values**

**>a1$vectors**

**>(a2=sweep(a1$vec[,a1$va>0],2,sqrt(a1$value[a1$va>0]),"\*"))#求相关系数(样本方差)**

# [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]

# [1,] -0.9539946 -0.2111907 0.02357019 -0.128512271 0.12068455 0.077845566 0.04516484 -0.0365022847

# [2,] -0.9304595 -0.2954593 0.09783004 0.049143751 0.06913723 -0.098687154 -0.08373996 -0.1121554208

# [3,] -0.9324676 -0.2436025 0.07416736 0.174341903 -0.11319048 0.102846590 0.03422057 -0.0963606843

# [4,] -0.9513464 -0.2457208 -0.05920610 -0.129894337 -0.03950131 0.025762764 0.01891129 -0.0574366164

# [5,] -0.9486355 -0.2467332 0.02669473 -0.117249791 0.10707771 0.039130883 0.04560991 0.0719084114

# [6,] -0.9216863 -0.3244078 0.09305758 0.058451651 0.04289831 -0.144276612 -0.08103137 0.0533270863

# [7,] -0.9268009 -0.2630371 0.07387343 0.182034406 -0.13887282 0.032862543 0.01764805 0.1040626784

# [8,] -0.9465203 -0.2736627 -0.04892409 -0.108695779 -0.05181674 -0.017867689 0.01641218 0.0719019682

# [9,] -0.7423258 0.5575414 -0.27827837 -0.140453280 -0.18137528 -0.022530219 -0.07016846 -0.0136786500

# [10,] -0.7353516 0.6149668 -0.14983165 0.110828336 0.07313890 -0.106469945 0.17033320 -0.0142756349

# [11,] -0.7177465 0.6441405 -0.04128197 0.112555391 0.16089940 0.115943851 -0.11187304 0.0349178727

# [12,] -0.2711881 0.6087331 0.73860748 -0.084538009 -0.05347639 -0.010571634 0.01417207 -0.0006251091

# [13,] -0.7550521 0.6352834 -0.15034338 -0.007535667 -0.02625708 -0.007777121 -0.01623365 -0.0032241809

# [,9] [,10] [,11] [,12] [,13]

# [1,] 0.055029441 0.0280203435 -0.0005930964 -1.977503e-02 4.486722e-03

# [2,] 0.007601750 0.0177929117 -0.0008178185 2.172964e-02 2.377329e-03

# [3,] 0.013107954 -0.0349883603 -0.0004698437 -4.456389e-03 2.019784e-03

# [4,] -0.090004597 0.0090013536 0.0021029276 -4.126728e-03 -9.972418e-03

# [5,] 0.057776090 -0.0268911315 -0.0040408823 2.016354e-02 -4.606238e-03

# [6,] 0.013354561 -0.0195271456 0.0000273517 -2.335235e-02 -2.602185e-03

# [7,] 0.018630678 0.0370707054 -0.0012730540 4.870373e-03 -2.227132e-03

# [8,] -0.079397974 -0.0095186107 0.0054196869 4.711029e-03 1.048690e-02

# [9,] 0.043723363 -0.0009835963 -0.0224818786 -4.687543e-04 5.330367e-04

# [10,] -0.017709306 0.0012761421 -0.0111166194 -4.205585e-04 1.614225e-04

# [11,] -0.048881108 -0.0009191012 -0.0096546427 -2.703910e-04 1.958981e-04

# [12,] -0.006131226 -0.0002333569 -0.0016997070 -9.216339e-05 2.250479e-05

# [13,] 0.029020707 -0.0007282274 0.0422175243 1.360362e-03 -8.522486e-04

par(mfrow=c(1,1))

**#载荷图**

**>(b=sweep(a$vec[,a$va>0],2,sqrt(a$value[a$va>0]),"\*"))#计算相关系数**

# [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]

# [1,] -0.9539946 -0.2111907 0.02357019 -0.128512271 0.12068455 0.077845566 0.04516484 -0.0365022847

# [2,] -0.9304595 -0.2954593 0.09783004 0.049143751 0.06913723 -0.098687154 -0.08373996 -0.1121554208

# [3,] -0.9324676 -0.2436025 0.07416736 0.174341903 -0.11319048 0.102846590 0.03422057 -0.0963606843

# [4,] -0.9513464 -0.2457208 -0.05920610 -0.129894337 -0.03950131 0.025762764 0.01891129 -0.0574366164

# [5,] -0.9486355 -0.2467332 0.02669473 -0.117249791 0.10707771 0.039130883 0.04560991 0.0719084114

# [6,] -0.9216863 -0.3244078 0.09305758 0.058451651 0.04289831 -0.144276612 -0.08103137 0.0533270863

# [7,] -0.9268009 -0.2630371 0.07387343 0.182034406 -0.13887282 0.032862543 0.01764805 0.1040626784

# [8,] -0.9465203 -0.2736627 -0.04892409 -0.108695779 -0.05181674 -0.017867689 0.01641218 0.0719019682

# [9,] -0.7423258 0.5575414 -0.27827837 -0.140453280 -0.18137528 -0.022530219 -0.07016846 -0.0136786500

# [10,] -0.7353516 0.6149668 -0.14983165 0.110828336 0.07313890 -0.106469945 0.17033320 -0.0142756349

# [11,] -0.7177465 0.6441405 -0.04128197 0.112555391 0.16089940 0.115943851 -0.11187304 0.0349178727

# [12,] -0.2711881 0.6087331 0.73860748 -0.084538009 -0.05347639 -0.010571634 0.01417207 -0.0006251091

# [13,] -0.7550521 0.6352834 -0.15034338 -0.007535667 -0.02625708 -0.007777121 -0.01623365 -0.0032241809

# [,9] [,10] [,11] [,12] [,13]

# [1,] 0.055029441 0.0280203435 -0.0005930964 -1.977503e-02 4.486722e-03

# [2,] 0.007601750 0.0177929117 -0.0008178185 2.172964e-02 2.377329e-03

# [3,] 0.013107954 -0.0349883603 -0.0004698437 -4.456389e-03 2.019784e-03

# [4,] -0.090004597 0.0090013536 0.0021029276 -4.126728e-03 -9.972418e-03

# [5,] 0.057776090 -0.0268911315 -0.0040408823 2.016354e-02 -4.606238e-03

# [6,] 0.013354561 -0.0195271456 0.0000273517 -2.335235e-02 -2.602185e-03

# [7,] 0.018630678 0.0370707054 -0.0012730540 4.870373e-03 -2.227132e-03

# [8,] -0.079397974 -0.0095186107 0.0054196869 4.711029e-03 1.048690e-02

# [9,] 0.043723363 -0.0009835963 -0.0224818786 -4.687543e-04 5.330367e-04

# [10,] -0.017709306 0.0012761421 -0.0111166194 -4.205585e-04 1.614225e-04

# [11,] -0.048881108 -0.0009191012 -0.0096546427 -2.703910e-04 1.958981e-04

# [12,] -0.006131226 -0.0002333569 -0.0016997070 -9.216339e-05 2.250479e-05

# [13,] 0.029020707 -0.0007282274 0.0422175243 1.360362e-03 -8.522486e-04

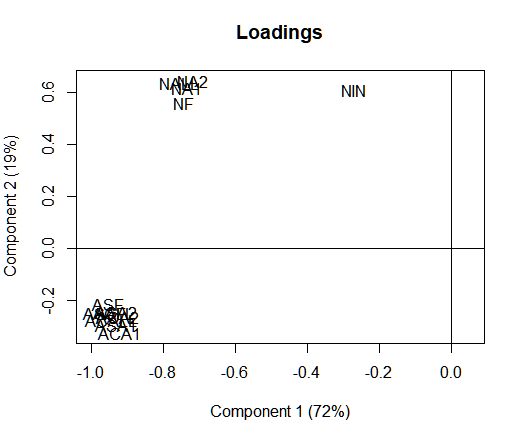
**>pct=paste("(",round(a1$va/sum(a1$va)\*100),"%)",sep="")**

**>tt=NULL;for(i in 1:4)tt=c(tt,paste("Component", i,pct[i]))**

**>plot(a2[,1:2], type="n", xlab=tt[1], ylab=tt[2],main="Loadings",**

**xlim=c(-1,0.05))**

**>text(a2[,1],a2[,2],names(u[,-1]));abline(h=0);abline(v=0)#标注点**



### 相关结论：



图中各个点的坐标对应的方差最大的2个主成分与原始变量的相关系数，即上表的两行数目相应于图中点的横、纵坐标。可以看出,第一主成分绝对值大(这里是负数)的点既对应和收入有关的变量(字母“A”开头的变量),也对应和人数有关的变量(字母“N”开头的变量),但变量NIN是例外;而第二主成分绝对值大(这里是正数)的点主要对应和人数有关的变量(字母“N”开头的变量)。

每一个选中的主成分所代表的特征可以用来给这些成分起名字，但并不一定都合理.给前两个主成分起名字就不那么容易. 第一主成分可以勉强称为“总体成分”(代表人数和收入二者),而第二主成分可称为“人力资源成分”(代表人数).

注：

1.载荷图每一点的横坐标为相应变量与第一主成分的线性相关系数，而其纵坐标为该变量与第二主成分的线性相关系数.

2.对于一个成分而言，我们仅仅关注各个变量与该成分相关程度的相对大小，比如，上图中，变量ASF 比 NIN 与第一主成分更加相关（无论正负)，但变量 NIN 比 ASF 与第二主成分更加相关，而且由于符号不同（无所谓哪个为正，哪个为负)，它们与第二主成分的相关方向相反（如果一个为正相关，则另一个为负相关,而且程度不同)。

**#计算主成分得分**

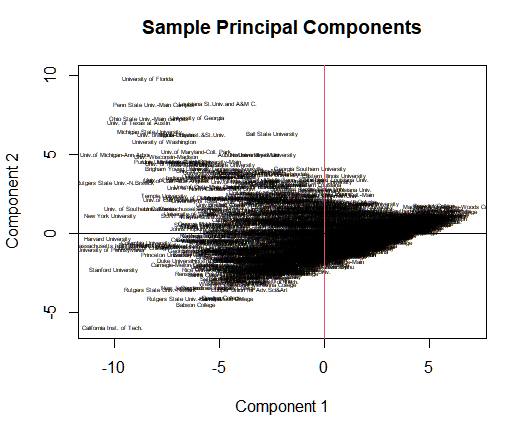
**>sc=scale(u[,-1])%\*%a1$ve[,1:2] #计算得分**

**>plot(sc[,1],sc[,2],type="n",ylim=c(-6,10),xlim=c(-11,7),**

**main="Sample Principal Components",xlab="Component 1",ylab="Component 2")**

**>text(sc[,1],sc[,2],u[,1],cex=.4);**

**>abline(v=0,col=2);abline(h=0,co1=2)**



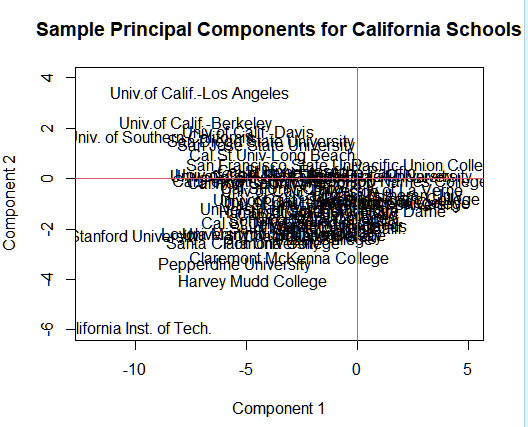
**>CA=(a$State=="CA")**

**>plot(sc[CA,1],sc[CA,2],type="n",ylim=c(-6,4),xlim=c(-12,5),**

**main="Sample Principal Components for California Schools",xlab="Component 1", >ylab="Component 2")**

**>text(sc[CA,1],sc[CA,2],u[CA,1],cex=1)**

**>abline(v=0,col=2);abline(h=0,col=2)**



### 相关结论：

上图为加利福尼亚州学校的得分图。从图中的横坐标(相应于第一主成分)可以看出加利福尼亚州哪些学校的平均收入和教师人数综合得分较高(负值越大)，比如加利福尼亚理工学院、斯坦福大学、南加利福尼亚大学、加利福尼亚大学伯克利分校和洛杉矶分校等最高.从图中的纵坐标(相应于第二主成分)可以看出加利福尼亚州哪些学校的教师人数得分较高，这里加利福尼亚大学洛杉矶分校、加利福尼亚大学伯克利分校、加利福尼亚大学戴维斯分校、南加利福尼亚大学、圣迭戈州立大学等得分最高,但由于第二主成分和变量的相关性远不如第一主成分强,因此对数据总体的解释性也弱.

注：主成分得分不能相加

**>par(mfrow=c(1,1))**

**# 得到上面两个成分的前5名学校名称的代码为**

**>u[CA,1][order(sc[CA, 1],decreasing =F)][1:5]**

**# [1] "California Inst. of Tech." "Stanford University" "Univ. of Southern California"**

**# [4] "Univ.of Calif.-Berkeley" "Univ.of Calif.-Los Angeles"**

**>u[CA,1][order(sc[CA, 2],decreasing =T)][1:5]**

### 江景图片压缩

**>library(jpeg)**

**>xmy=readJPEG("MVAPureData/XMY.JPG")**

**>xmypca=list()**

**>for(i in 1:3)xmypca[[i]]=prcomp(xmy[,,i],center = FALSE)**

**>imge=list()**

**>for (i in c(3,10,100)){**

**>imge[[i]]<-sapply(xmypca, function(y) {**

**comimge<-y$x[,1:i] %\*% t(y$rotation[,1:i])**

**},simplify='array')**

**writeJPEG(imge[[i]],paste('compXMY',round(i,0),'.jpg',sep=''))**

**}**

**>FS=file.info('XMY.jpg')$size**

**>z=NULL**

**>for(i in c(3,10,100))**

**>z=c(z,file.info(paste("compXMY ",i,".jpg",sep=""))$size/FS)**

**>z**

＃0.1165802　0.142081　0.2239737

### 洛杉矶街区数据

**>w=read.csv("C:/Users/86133/Desktop/《多元统计分析——R与Python的实现》数据文件/MVAPureData/LA.Neighborhoods.csv")**

**>w$density=w$Population/w$Area#增加密度**

**>u=w[,-c(12:15)]#去掉人口,面积,经纬度**

**>library("FactoMineR")**

**>library("ggplot2")**

**>library("factoextra")**

**>row.names(u)=u[,1]**

**>u.r=PCA(u[,-1],scale.unit=TRUE, graph=FALSE)#主程序**

**>print(u.r)**

# \*\*Results for the Principal Component Analysis (PCA)\*\*

# The analysis was performed on 110 individuals, described by 11 variables

# \*The results are available in the following objects:

#

# name description

# 1 "$eig" "eigenvalues"

# 2 "$var" "results for the variables"

# 3 "$var$coord" "coord. for the variables"

# 4 "$var$cor" "correlations variables - dimensions"

# 5 "$var$cos2" "cos2 for the variables"

# 6 "$var$contrib" "contributions of the variables"

# 7 "$ind" "results for the individuals"

# 8 "$ind$coord" "coord. for the individuals"

# 9 "$ind$cos2" "cos2 for the individuals"

# 10 "$ind$contrib" "contributions of the individuals"

# 11 "$call" "summary statistics"

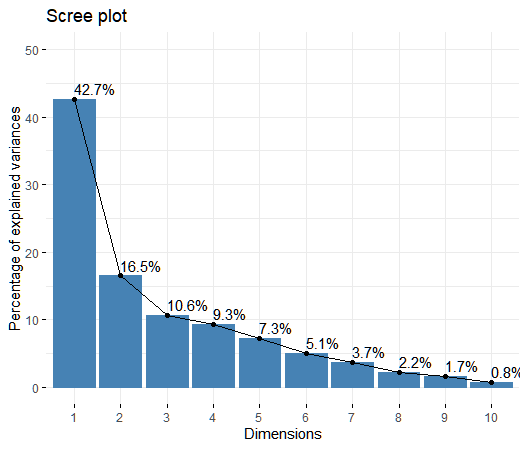
# 12 "$call$centre" "mean of the variables"

# 13 "$call$ecart.type" "standard error of the variables"

# 14 "$call$row.w" "weights for the individuals"

# 15 "$call$col.w" "weights for the variables"

**>fviz\_eig(u.r,addlabels =TRUE,ylim = c(0,50))#特征值，各个特征值所占总方差的比例及比例崖底碎石图**

****

**>(u.eigen<-get\_eigenvalue(u.r))#特征值，各个特征值所占总方差的比例及累积比例**

# eigenvalue variance.percent cumulative.variance.percent

# Dim.1 4.6930331513 42.66393774 42.66394

# Dim.2 1.8176421789 16.52401981 59.18796

# Dim.3 1.1687855390 10.62532308 69.81328

# Dim.4 1.0269743864 9.33613079 79.14941

# Dim.5 0.8062945744 7.32995068 86.47936

# Dim.6 0.5577210494 5.07019136 91.54955

# Dim.7 0.4099288177 3.72662562 95.27618

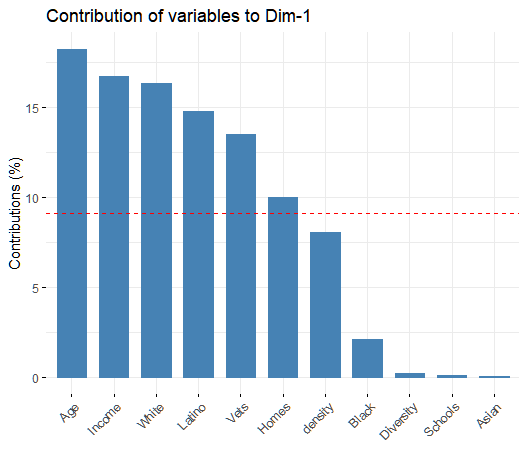
# Dim.8 0.2437073056 2.21552096 97.49170

# Dim.9 0.1870581845 1.70052895 99.19223

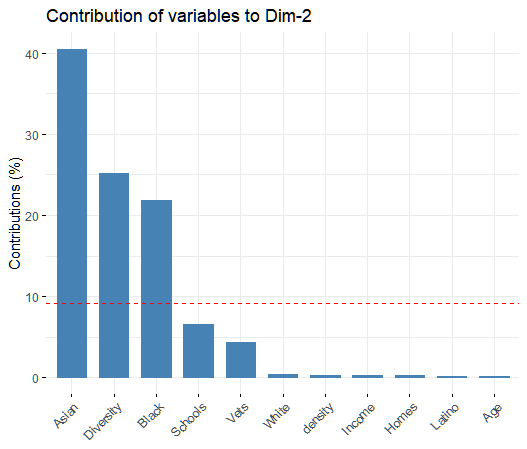
# Dim.10 0.0882870577 0.80260962 99.99484

# Dim.11 0.0005677551 0.00516141 100.00000

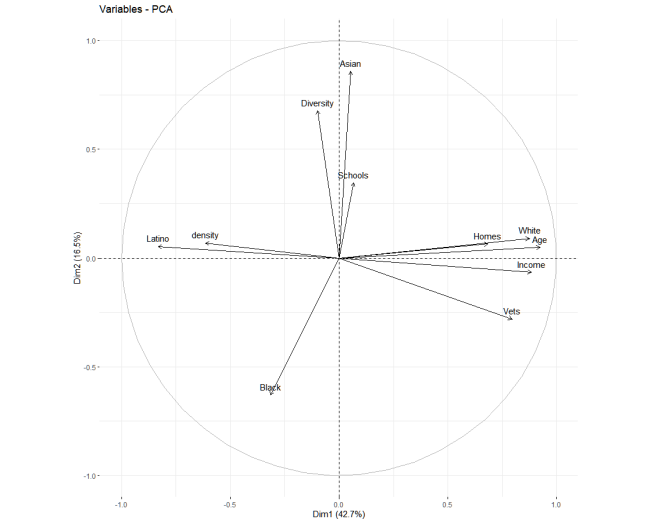
**>fviz\_contrib(u.r, choice="var",axes=1)#主成分各个变量对第一主成分的贡献率图**



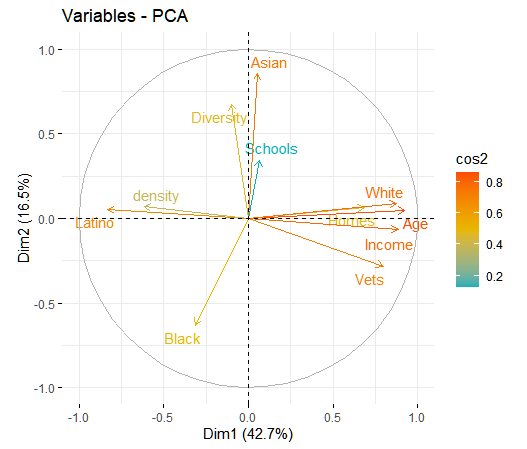
**>fviz\_contrib(u.r,choice="var",axes=2)#主成分各个变量对第二主成分的贡献率图**



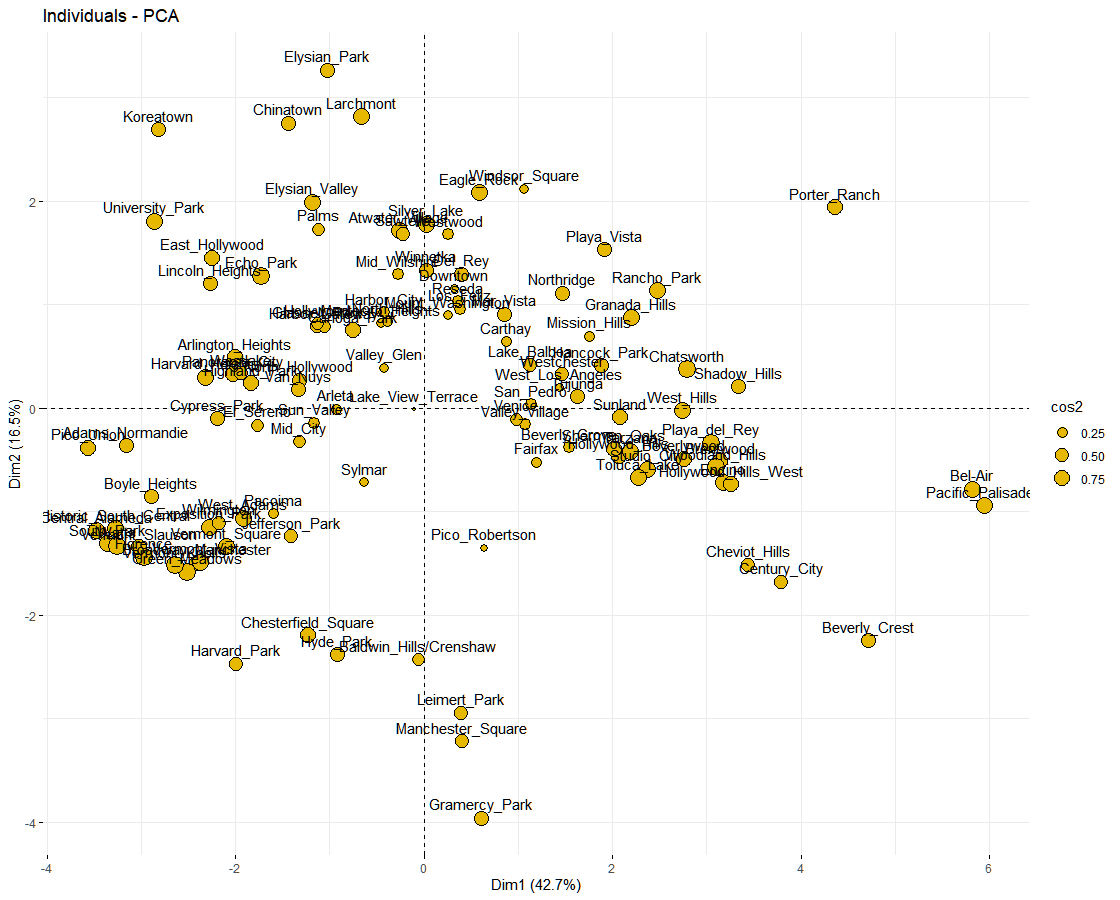
**>fviz\_pca\_var(u.r)#载荷图**



**>fviz\_pca\_var(u.r,col.var="cos2",gradient.cols=c("#00AFBB","#E7B800","#FC4E07"),repel = TRUE)#加上cos2的载荷图**



**>fviz\_pca\_ind(u.r,pointsize="cos2",pointshape=21,fill="#E7B800",repel=F)#记分点图**



# 因子分析

## 步骤：

1.中心化原始变量矩阵，求特征值和特征向量

2.初步选择因子，不做旋转

3.按照方差大于1的原则，选取因子做因子分析并旋转

4.计算因子得分，并做出因子得分图

5.因子得分排序

## 求值：

1.v1=scale(v[,-1])#中心化原始变量矩阵

2.(v2=factanal(v1,factors=7,rotation ="none"))#初步因子分析，不进行旋转Q,暂时选择7个因子

3.SS=apply(v2$loadings^2,2,sum)#计算载荷的每一列平方和"2"表示对列进行操作，"1"表示对行进行操作

4.PV=SS/ncol(v[,-1])#计算各个因子所占方差的比例

5.CV=cumsum(SS/ncol(v[,-1]))#计算累计因子比例

6.rbind(SS,PV,CV)#查看

7.(v31=factanal(v1,factors=6,rotation ="varimax",scores="regression"))

8.v5=princomp(v1)#求每个变量的标准偏差

9.(e<-eigen(r))#相关矩阵求特征值和特征向量

10.(cumsum(e$values/p))#因子解释方差的累计贡献率

11.or1<-order(a2$scores[,1],decreasing = T)#因子得分排序，decreasing:递减

12.colSums(e$vectors[,1:m]^2)#纵向加和，特征向量1到m列的平方

13.(Q=e$vectors[,1:m]%\*%diag(sqrt(e$values[1:m])))#1、%\*%做内积，sqrt：平方根

14.lambda<-colSums(Q^2)#每个因子对方差的贡献，函数式接口

15.(Qs=varimax(x=Q, normalize = TRUE, eps = 1e-5))#对因子载荷矩阵进行旋转 varimax：方差最大正交旋转，正交旋转法 normalize:标准化

16.(as.matrix(Qs$loadings))#转换为矩阵，维数相同

17.print(Qs$loadings, cutoff=0.00001)#打印旋转之后的因子载荷，cutoff最佳阀值

18.Qs$rotmat#旋转的正交阵

19.#旋转之后的共性方差

hs2<-c()#定义一个空变量

for(i in 1:p){

hs2[i]<-sum(Qs$loadings[i,]^2)

}

hs2

20.colSums(e$vectors[,1:m]^2)#特征向量纵向加和，特征向量1到m列的平方

21.#求载荷，%\*%做内积

(Q=e$vectors[,1:m]%\*%diag(sqrt(e$values[1:m])))

h2#输出h2,Q的平方求和

1-h2#特殊方差

22.(colSums(Q^2))#载荷的累计平方和

23.#对因子载荷矩阵进行旋转 varimax：方差最大正交旋转，正交旋转法 normalize:标准化

(Qs=varimax(x=Q, normalize = TRUE, eps = 1e-5))

24.(as.matrix(Qs$loadings))#转换为矩阵，维数相同

25.print(Qs$loadings, cutoff=0.00001)#打印旋转之后的因子载荷，cutoff最佳阀值

26.Qs$rotmat#旋转的正交阵

27.#旋转之后的共性方差

hs2<-c()#定义一个空变量

for(i in 1:p){

hs2[i]<-sum(Qs$loadings[i,]^2)

}

hs2

## 画图：

1.corrplot::corrplot(cor(v[,-c(1)]),type="lower",diag=FALSE,tl.col="black",tl.srt=45)

2.#

B=matrix(c(1:4,4,5),3,2,b=T)

TT=c("FA with rotation","FA without rotation","PCA")

A=list(v3$loadings,v4$loadings,v5$loadings)

for(k in 1:3)

for (i in 1:3){

plot(A[[k]][,B[i,]],xlim=c(-1,1),ylim=c(-1,1),type="n")

text(A[[k]][,B[i,]],names(v1),cex=.6)

abline(h=0,lty=2)

abline(v=0,ity=2)

title(TT[k])

}

3.#因子得分图

for(i in 1:2){

plot(v3$scores[,B[i,]],type="n",ylim=range(v3$scores[,B[i,2]]),

xlim=range(v3$scores[,B[i,1]]+c(-.3,.3)),main="Factor Scores")

abline(h=0)

abline(v=0)

text(v3$scores[,B[i,]],labels=v[,1],cex=.5)

}

### WHO数据:

**>v=read.csv("C:/Users/86133/Desktop/《多元统计分析——R与Python的实现》数据文件/MVAPureData/who1.csv")**

**>corrplot::corrplot(cor(v[,-c(1)]),type="lower",diag=FALSE,tl.col="black",tl.srt=45)**

**>v1=scale(v[,-1])#中心化原始变量矩阵**

**>(v2=factanal(v1,factors=7,rotation ="none"))#初步因子分析，不进行旋转Q,暂时选择7个因子**

# Call:

# factanal(x = v1, factors = 7, rotation = "none")

#

# Uniquenesses:

# PT PGR PUP P60P P15P TFR IMR I3R HER HGR HGGR NDP NNMP NPP NMP HPP HPHP NMPP HGDP DR1

# 0.554 0.382 0.490 0.179 0.008 0.138 0.120 0.081 0.520 0.005 0.653 0.219 0.005 0.402 0.766 0.882 0.005 0.667 0.718 0.910

# DR2 DR3 DR4 TBN NMN YCP YIP YNCP

# 0.102 0.347 0.005 0.366 0.174 0.005 0.005 0.005

#

# Loadings:

# Factor1 Factor2 Factor3 Factor4 Factor5 Factor6 Factor7

# PT 0.414 0.494 0.161

# PGR 0.633 -0.129 0.154 -0.302 -0.290

# PUP -0.675 0.134 -0.154

# P60P -0.770 0.176 -0.322 0.110 -0.131 0.240

# P15P 0.886 -0.242 0.145 -0.166 -0.308

# TFR 0.824 -0.257 0.113 0.108 -0.125 -0.268

# IMR -0.565 0.104 -0.109 0.144 0.715

# I3R -0.561 0.124 -0.117 0.135 0.743

# HER 0.527 -0.413 0.169

# HGR -0.639 -0.738 0.168 0.130

# HGGR -0.455 -0.296 0.192

# NDP -0.202 0.438 0.712 0.157 -0.109

# NNMP -0.232 0.482 0.824 0.172

# NPP -0.160 0.431 0.586 0.201

# NMP -0.426 -0.190

# HPP 0.117 -0.204 0.218

# HPHP 0.638 0.738 -0.168 -0.130

# NMPP 0.363 -0.366 0.213

# HGDP -0.394 0.235 -0.151 0.104 -0.147

# DR1 0.103 -0.152 0.215

# DR2 0.526 -0.101 0.396 0.666

# DR3 0.596 0.486 -0.154 0.189

# DR4 0.719 -0.130 0.384 0.558

# TBN 0.648 -0.252 0.202 0.309

# NMN 0.875 -0.150 0.116 0.100

# YCP 0.932 -0.236 0.154 -0.202

# YIP -0.549 0.332 -0.299 0.661 -0.240

# YNCP -0.912 0.186 -0.179 0.281

#

# Factor1 Factor2 Factor3 Factor4 Factor5 Factor6 Factor7

# SS loadings 9.796 2.798 2.290 1.416 1.228 1.184 0.583

# Proportion Var 0.350 0.100 0.082 0.051 0.044 0.042 0.021

# Cumulative Var 0.350 0.450 0.532 0.582 0.626 0.668 0.689

#

# Test of the hypothesis that 7 factors are sufficient.

# The chi square statistic is 3077.73 on 203 degrees of freedom.

# The p-value is 0

**>SS=apply(v2$loadings^2,2,sum)#计算载荷的每一列平方和"2"表示对列进行操作，"1"表示对行进行操作**

**>PV=SS/ncol(v[,-1])#计算各个因子所占方差的比例**

**>CV=cumsum(SS/ncol(v[,-1]))#计算累计因子比例**

**>rbind(SS,PV,CV)#查看**

# Factor1 Factor2 Factor3 Factor4 Factor5 Factor6 Factor7

# SS 9.796052 2.79791183 2.29027212 1.41608066 1.22791858 1.18353393 0.58334721

# PV 0.349859 0.09992542 0.08179543 0.05057431 0.04385424 0.04226907 0.02083383

# CV 0.349859 0.44978443 0.53157986 0.58215417 0.62600841 0.66827748 0.68911130

**>v2$uniquenesses#特殊因子**

# PT PGR PUP P60P P15P TFR IMR I3R HER HGR

# 0.554393814 0.382240719 0.490043417 0.178799916 0.008012719 0.138124773 0.120311661 0.081445414 0.520127612 0.002504068

# HGGR NDP NNMP NPP NMP HPP HPHP NMPP HGDP DR1

# 0.652513004 0.218738695 0.004992962 0.402195070 0.765843226 0.882282708 0.002504136 0.666899733 0.718325238 0.910053595

# DR2 DR3 DR4 TBN NMN YCP YIP YNCP

# 0.101938619 0.346873851 0.004995482 0.366457554 0.174120155 0.002398975 0.004882982 0.002863391

(1-apply(v2$loadings^2,1,sum))#独特性=1-sum（SS loadings），具有较低的共同度

# PT PGR PUP P60P P15P TFR IMR I3R HER HGR

# 0.554393814 0.382240719 0.490043417 0.178799916 0.008012719 0.138124773 0.120311661 0.081445414 0.520127612 0.002504068

# HGGR NDP NNMP NPP NMP HPP HPHP NMPP HGDP DR1

# 0.652513004 0.218738695 0.004992962 0.402195070 0.765843226 0.882282708 0.002504136 0.666899733 0.718325238 0.910053595

# DR2 DR3 DR4 TBN NMN YCP YIP YNCP

# 0.101938619 0.346873851 0.004995482 0.366457554 0.174120155 0.002398975 0.004882982 0.002863391

### 相关结论：

根据方差大于1的原则选择因子，由于第七个因子的方差小于1，舍弃，选择6个因子

**#选择因子数目m，旋转并作载荷图**

**#按照方差大于1的原则，选取因子做因子分析并旋转**

**>(v31=factanal(v1,factors=6,rotation ="varimax",scores="regression"))**

# Call:

# factanal(x = v1, factors = 6, scores = "regression", rotation = "varimax")

#

# Uniquenesses:

# PT PGR PUP P60P P15P TFR IMR I3R HER HGR HGGR NDP NNMP NPP NMP HPP HPHP NMPP HGDP DR1

# 0.556 0.503 0.490 0.288 0.158 0.243 0.130 0.067 0.524 0.005 0.654 0.232 0.005 0.404 0.779 0.887 0.005 0.667 0.735 0.965

# DR2 DR3 DR4 TBN NMN YCP YIP YNCP

# 0.113 0.373 0.005 0.424 0.177 0.005 0.005 0.005

#

# Loadings:

# Factor1 Factor2 Factor3 Factor4 Factor5 Factor6

# PT 0.650 -0.128

# PGR 0.630 -0.179 -0.234

# PUP -0.559 0.185 -0.287 0.106 -0.260

# P60P -0.748 0.141 0.201 -0.260 0.134

# P15P 0.839 -0.120 -0.200 0.160 -0.234

# TFR 0.771 -0.141 0.159 -0.319 0.100

# IMR -0.370 0.153 0.840

# I3R -0.367 0.127 0.882

# HER 0.609 -0.102 0.126 0.117 -0.126 0.222

# HGR -0.176 -0.125 0.967

# HGGR -0.204 0.493 -0.228

# NDP -0.105 0.861 -0.114

# NNMP -0.140 0.984

# NPP -0.105 0.762

# NMP -0.375 0.210 -0.138 0.118

# HPP -0.144 -0.137 0.259

# HPHP 0.176 0.125 -0.967

# NMPP 0.471 0.161 0.284

# HGDP -0.281 0.154 0.307 -0.109 0.134 0.195

# DR1 -0.133

# DR2 0.294 -0.148 0.882

# DR3 0.565 -0.207 0.332 -0.211 -0.324

# DR4 0.509 -0.153 0.835 -0.101

# TBN 0.719 0.218

# NMN 0.774 -0.239 0.290 -0.238 0.157

# YCP 0.929 -0.214 -0.229 0.166

# YIP -0.392 0.113 0.191 -0.889

# YNCP -0.945 0.232 0.210

#

# Factor1 Factor2 Factor3 Factor4 Factor5 Factor6

# SS loadings 7.496 2.884 2.785 2.091 2.041 1.309

# Proportion Var 0.268 0.103 0.099 0.075 0.073 0.047

# Cumulative Var 0.268 0.371 0.470 0.545 0.618 0.664

#

# Test of the hypothesis that 6 factors are sufficient.

# The chi square statistic is 3377.87 on 225 degrees of freedom.

# The p-value is 0

**>(v3=factanal(v1,factors=5,rotation ="varimax",scores="regression"))**

# Call:

# factanal(x = v1, factors = 5, scores = "regression", rotation = "varimax")

#

# Uniquenesses:(特殊因子)

# PT PGR PUP P60P P15P TFR IMR I3R HER HGR HGGR NDP NNMP NPP NMP HPP HPHP NMPP HGDP DR1

# 0.557 0.512 0.493 0.292 0.160 0.260 0.655 0.657 0.525 0.005 0.654 0.233 0.005 0.404 0.780 0.894 0.005 0.675 0.743 0.977

# DR2 DR3 DR4 TBN NMN YCP YIP YNCP

# 0.114 0.388 0.005 0.435 0.179 0.005 0.005 0.005

#

# Loadings:

# Factor1 Factor2 Factor3 Factor4 Factor5

# PT 0.651 -0.127

# PGR 0.655 -0.181 0.146

# PUP -0.584 0.182 -0.303 0.198

# P60P -0.722 0.146 0.200 -0.280 -0.217

# P15P 0.867 -0.124 -0.199 0.180

# TFR 0.825 -0.142 0.177

# IMR -0.557 0.165

# I3R -0.562 0.141

# HER 0.637 -0.107 0.129 0.135 -0.155

# HGR -0.190 -0.126 0.966 -0.102

# HGGR -0.199 0.494 -0.231

# NDP -0.105 0.862 -0.113

# NNMP -0.128 0.985

# NPP -0.108 0.763

# NMP -0.359 0.210 -0.147 -0.157

# HPP -0.143 -0.140 0.255

# HPHP 0.190 0.126 -0.966 0.102

# NMPP 0.492 0.165 -0.225

# HGDP -0.281 0.154 0.310 -0.112 -0.213

# DR1 0.118

# DR2 0.265 -0.149 0.890

# DR3 0.556 -0.211 0.341 0.371

# DR4 0.499 -0.154 0.848

# TBN 0.734 -0.134

# NMN 0.812 -0.239 0.310

# YCP 0.967 -0.211

# YIP -0.511 0.115 0.847

# YNCP -0.959 0.230 -0.118

#

# Factor1 Factor2 Factor3 Factor4 Factor5

# SS loadings 8.317 2.896 2.796 2.198 1.182

# Proportion Var 0.297 0.103 0.100 0.078 0.042

# Cumulative Var 0.297 0.400 0.500 0.579 0.621

#

# Test of the hypothesis that 5 factors are sufficient.

# The chi square statistic is 3648.44 on 248 degrees of freedom.

# The p-value is 0

**>v4=factanal(v1,factors=5,rotation ="none" )**

**>v5=princomp(v1)#求每个变量的标准偏差**

**#绘制前5个因子间的 图**

**>B=matrix(c(1:4,4,5),3,2,b=T)**

**>TT=c("FA with rotation","FA without rotation","PCA")**

**>A=list(v3$loadings,v4$loadings,v5$loadings)**

**>for(k in 1:3)**

**for (i in 1:3){**

**plot(A[[k]][,B[i,]],xlim=c(-1,1),ylim=c(-1,1),type="n")**

**text(A[[k]][,B[i,]],names(v1),cex=.6)**

**abline(h=0,lty=2)**

**abline(v=0,ity=2)**

**title(TT[k])**

**}**

### 相关结论：

由 图可知

**#计算因子得分，并做出因子得分图**

**>layout(t(1:2))**

**>for(i in 1:2){**

**plot(v3$scores[,B[i,]],type="n",ylim=range(v3$scores[,B[i,2]]),**

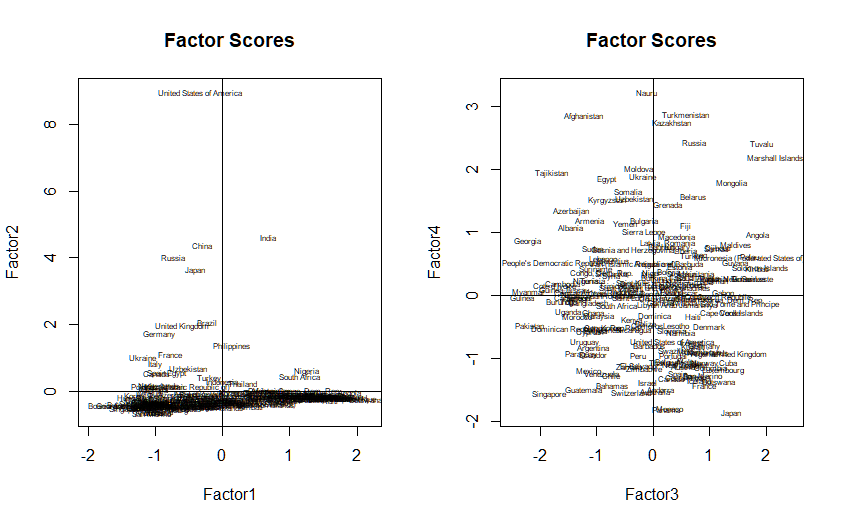
**xlim=range(v3$scores[,B[i,1]]+c(-.3,.3)),main="Factor Scores")**

**abline(h=0)**

**abline(v=0)**

**text(v3$scores[,B[i,]],labels=v[,1],cex=.5)**

**}**



### 相关结论：

### 因子分析数据：

**>w<-read.csv("C:/Users/86133/Desktop/《多元统计分析——R与Python的实现》数据文件/因子分析//因子分析练习题/因子分析练习题/examp8.4.4.csv")**

**>dim(w)**

**>head(w)**

**>(r<-cor(w[,-c(1:2)]))#相关系数**

**>w1=w[,-c(1:2)]**

**>scale(w1)**

**>(round(r,digits = 3))#保留相关系数的后三位**

**>(p=ncol(w))#列数**

**>(e<-eigen(r))#相关矩阵求特征值和特征向量**

# $values

# [1] 4.87935018 2.57351770 0.92872744 0.71337423 0.38954188 0.18086312 0.15852455

# [8] 0.08889171 0.07457816 0.01263102

#

# $vectors

# [,1] [,2] [,3] [,4] [,5] [,6]

# [1,] -0.2984794 0.294455398 -0.12560876 0.49329208 -0.50476261 -0.53105795

# [2,] -0.3778624 0.215980580 -0.10058822 -0.04881946 -0.42113542 0.65486564

# [3,] -0.4010015 -0.001666714 0.03826003 -0.52610607 -0.09039290 -0.12397627

# [4,] -0.4019224 -0.023336298 0.08475375 -0.50270049 0.01020917 -0.24436294

# [5,] -0.3014348 -0.431120895 -0.11352121 0.11853050 -0.03058685 -0.06884232

# [6,] -0.1772059 -0.228708567 -0.84512501 0.06854450 0.30491597 0.02825564

# [7,] -0.2387871 -0.417371808 0.33762295 0.34018548 -0.01659949 0.34724914

# [8,] -0.2631010 -0.438041806 0.26930235 0.15724831 0.09495669 -0.20373059

# [9,] -0.3383548 0.351581614 -0.01967564 0.19445164 0.30219597 0.20075105

# [10,] -0.2877430 0.371804696 0.22679050 0.16664859 0.60420240 -0.07821980

# [,7] [,8] [,9] [,10]

# [1,] -0.13162870 0.088406212 -0.05437446 -0.013610229

# [2,] 0.36582164 0.216432795 -0.01310705 0.105007560

# [3,] -0.16055098 0.016517681 -0.02472114 -0.714662182

# [4,] -0.19825819 0.003649635 -0.07102397 0.688361338

# [5,] 0.18185155 -0.243232938 0.77333832 0.019392611

# [6,] -0.09794906 0.203225343 -0.22949571 -0.004567762

# [7,] -0.61229153 0.184437169 -0.09582422 -0.010406841

# [8,] 0.58179645 -0.052535415 -0.49793140 -0.044054039

# [9,] -0.10646762 -0.747308861 -0.14835217 -0.016438313

# [10,] 0.12797694 0.499344229 0.24772832 -0.038582275

**(cumsum(e$values/p))#因子解释方差的累计贡献率**

# [1] 0.4066125 0.6210723 0.6984663 0.7579141 0.7903760 0.8054479 0.8186583 0.8260659

# [9] 0.8322807 0.8333333

**>m=3#因子数为3**

**>GGally::ggpairs(w1)**

**>corrplot::corrplot(cor(w1),type="lower",diag=FALSE)**

**>ws<-scale(w1) #极大似然估计,直接进行数据的中心化和标准化**

**>a=factanal(scale(w1),factor=6,rotation="none",scores="regression") #极大似然估计 scale:标准化 4个因子 没旋转**

# Call:

# factanal(x = scale(w1), factors = 6, scores = "regression", rotation = "none")

#

# Uniquenesses:

# x1 x2 x3 x4 x5 x6 x7 x8 x9 x10

# 0.005 0.168 0.005 0.009 0.030 0.251 0.202 0.111 0.005 0.187

#

# Loadings:

# Factor1 Factor2 Factor3 Factor4 Factor5 Factor6

# x1 0.785 -0.526 0.314

# x2 0.874 -0.127 0.214

# x3 0.861 0.492

# x4 0.851 0.506

# x5 0.411 0.480 0.605 0.444

# x6 0.257 0.182 0.400 0.228 -0.619 -0.234

# x7 0.275 0.371 0.583 0.354 0.324 0.117

# x8 0.314 0.492 0.573 0.359 0.296

# x9 0.873 -0.333 0.118 -0.330

# x10 0.742 -0.287 -0.355 0.228

#

# Factor1 Factor2 Factor3 Factor4 Factor5 Factor6

# SS loadings 4.568 1.612 1.228 0.838 0.654 0.126

# Proportion Var 0.457 0.161 0.123 0.084 0.065 0.013

# Cumulative Var 0.457 0.618 0.741 0.825 0.890 0.903

#

# The degrees of freedom for the model is 0 and the fit was 0.0187

### 相关结论

由于第4、5、6个因子的方差小于1，舍弃4、5、6，选择前3个因子，做下一步因子分析。

**a2=factanal(scale(w1),factors=3,rotation="varimax",scores="regression")**

# Call:

# factanal(x = scale(w1), factors = 3, scores = "regression", rotation = "varimax")

#

# Uniquenesses:

# x1 x2 x3 x4 x5 x6 x7 x8 x9 x10

# 0.427 0.273 0.005 0.028 0.159 0.830 0.228 0.120 0.011 0.251

#

# Loadings:

# Factor1 Factor2 Factor3

# x1 0.741 0.132

# x2 0.674 0.156 0.498

# x3 0.384 0.341 0.855

# x4 0.383 0.387 0.822

# x5 0.878 0.262

# x6 0.377 0.138

# x7 0.876

# x8 0.922 0.170

# x9 0.969 0.222

# x10 0.842 0.198

#

# Factor1 Factor2 Factor3

# SS loadings 2.956 2.830 1.881

# Proportion Var 0.296 0.283 0.188

# Cumulative Var 0.296 0.579 0.767

#

# Test of the hypothesis that 3 factors are sufficient.

# The chi square statistic is 786.89 on 18 degrees of freedom.

# The p-value is 1.96e-155

**>plot(a2$loadings[,1:3],type="n",xlim=c(-1.1,1.1))**

**>text(a2$loadings[,1:3],labels=names(w),cex=.6)**

**>abline(h=0)#横虚线**

**>abline(v=0)#竖直线**

**>a2$scores**

**>or1<-order(a2$scores[,1],decreasing = T)#decreasing:递减**

**>or1[1:5]**

# [1] 73 58 74 110 80

**>w[or1[1:2],]**

# code name x1 x2 x3 x4 x5 x6

# 73 600115 东方航 12839339289 2219037964 163016306 132919443 0.02731062 1.282399

# 58 600688 上海石 20197396000 1776146000 93509000 71604000 0.00994500 1.885069

# x7 x8 x9 x10

# 73 2.129651 0.4859015 27355226064 4866950000

# 58 0.527567 0.2890734 24770182000 7200000000

### 相关结论

第1因子得分较高的是东方航和上海石

**>or2<-order(a2$scores[,2],decreasing = T)#decreasing:递减**

**>or2[1:5]**

# [1] 599 604 603 601 591

w[or2[1:2],]

# code name x1 x2 x3 x4 x5 x6 x7

# 599 600345 长江通 389556202 59264065 197859128 199529581 1.2092702 5.404813 22.37395

# 604 600536 中软股 325209335 77179132 35175581 30106691 0.5557527 1.382341 40.20374

# x8 x9 x10

# 599 14.95774 1333955439 165000000

# 604 18.40036 163620137 54172824

### 相关结论

第2因子得分较高的是长江通和中软股

**>or3<-order(a2$scores[,3],decreasing = T)#decreasing:递减**

**>or3[1:5]**

# [1] 564 582 569 574 562

w[or3[1:2],]

# code name x1 x2 x3 x4 x5 x6

# 564 600188 兖州煤 6469352955 3557065177 1391875094 1000387449 0.3485671 3.039487

# 582 600098 广州控 3907556249 1637235778 1334451371 942831080 0.7525791 3.833131

# x7 x8 x9 x10

# 564 11.46796 8.813813 11350223177 2870000000

# 582 19.63353 10.330735 9126466814 1252800000

### 相关结论

第3因子得分较高的是兖州煤和广州控

**以下？？？**

**>colSums(e$vectors[,1:m]^2)#特征向量纵向加和，特征向量1到m列的平方**

**>(Q=e$vectors[,1:m]%\*%diag(sqrt(e$values[1:m])))#求载荷，%\*%做内积**

# [,1] [,2] [,3]

# [1,] -0.6593187 0.472370862 -0.12104979

# [2,] -0.8346698 0.346480089 -0.09693738

# [3,] -0.8857822 -0.002673774 0.03687139

# [4,] -0.8878164 -0.037436527 0.08167762

# [5,] -0.6658468 -0.691612210 -0.10940097

# [6,] -0.3914346 -0.366898564 -0.81445124

# [7,] -0.5274628 -0.669555667 0.32536895

# [8,] -0.5811704 -0.702714866 0.25952804

# [9,] -0.7474003 0.564013807 -0.01896151

# [10,] -0.6356027 0.596456053 0.21855916

**>round(Q,digits = 3)#round:四舍五入保留有效位数,digits:小数点后3位**

**>(h2<-c())#给h2附空值**

**>for(i in 1:p){**

**h2[i]<-sum(Q[i,]^2)**

**}#做循环Q的平方求和**

**>h2#输出h2,Q的平方求和**

# [1] 0.6724884 0.8261189 0.7859768 0.7962908 0.9336480 0.9511664 0.8323868 0.8989221

# [9] 0.8770783 0.8075188

**>1-h2#特殊方差**

# [1] 0.32751161 0.17388108 0.21402317 0.20370925 0.06635195 0.04883358 0.16761323

# [8] 0.10107792 0.12292166 0.19248122

**>(colSums(Q^2))#载荷的累计平方和**

# [1] 4.8793502 2.5735177 0.9287274

**>lambda<-colSums(Q^2)#每个因子对方差的贡献**

**>lambda#函数式接口**

# [1] 4.8793502 2.5735177 0.9287274

**>e$values[1:m]#1到m个特征值**

# [1] 4.8793502 2.5735177 0.9287274

#对因子载荷矩阵进行旋转 varimax：方差最大正交旋转，正交旋转法 normalize:标准化

**>(Qs=varimax(x=Q, normalize = TRUE, eps = 1e-5))**

# $loadings

#

# Loadings:

# [,1] [,2] [,3]

# [1,] -0.809 -0.129

# [2,] -0.874 -0.172 -0.182

# [3,] -0.706 -0.509 -0.167

# [4,] -0.688 -0.553 -0.135

# [5,] -0.114 -0.849 -0.447

# [6,] -0.199 -0.951

# [7,] -0.912

# [8,] -0.943

# [9,] -0.936

# [10,] -0.869 0.228

#

# [,1] [,2] [,3]

# SS loadings 4.044 3.077 1.261

# Proportion Var 0.404 0.308 0.126

# Cumulative Var 0.404 0.712 0.838

#

# $rotmat

# [,1] [,2] [,3]

# [1,] 0.79866895 0.5577036 0.2260412

# [2,] -0.60165119 0.7475203 0.2814769

# [3,] -0.01198976 -0.3608048 0.9325643

**>(as.matrix(Qs$loadings))#转换为矩阵，维数相同**

**>print(Qs$loadings, cutoff=0.00001)#打印旋转之后的因子载荷，cutoff最佳阀值**

**>Qs$rotmat#旋转的正交阵**

**>hs2<-c()#定义一个空变量**

**>for(i in 1:p){**

**hs2[i]<-sum(Qs$loadings[i,]^2)**

**}**

**>hs2#旋转之后的共性方差**

# [1] 0.6724884 0.8261189 0.7859768 0.7962908 0.9336480 0.9511664 0.8323868 0.8989221

# [9] 0.8770783 0.8075188

# 聚类分析

## 步骤：

1.计算距离，绘制距离图或者距离树状图

2.计算集群倾向，hopkins统计量值

3.分层聚类，画聚类图

4.确定聚类个数

5.判断聚类正确率

## 求值：

1.#计算hopkins统计量值

a=get\_clust\_tendency(scale(cw[,-8]),nrow(cw)-1,graph=TRUE)

clustertend::hopkins()

a$hopkins\_stat

2.#计算欧氏距离，利用完全连接距离，进行层次聚类

hh=hclust(dist(scale(cw[,-8])),"complete")

d<-dist(x=cw[,-8],method = "euclidean")

ih<-hclust(d=d,method = "complete")

3.#输出每一类编号和求每一类的中位数

(ihc<-cutree(tree = ih,k=3))#得到每一类的编号

ihc1<-aggregate(x=cw[,-8],by=list(cluster=ihc),median)

4.#层次聚类

ihc2<-factoextra::hcut(x=cw[,-8], k=3, hc\_func = "hclust", hc\_method = "complete", hc\_metric = "euclidean", graph=T)

5.table(res$cluster, cw[,8])#判断聚类正确率

6.a=kmeans(scale(cw),3)

7.a$cluster#类别

8.a$centers#类的中心点

9.a$withinss#每类的类内内平方和

10.a$tot.withinss#所有类的类内内平方和，求和

#分层k-means

res.hk<-hkmeans(scale(cw[,-8]),3)

11.hk<-hkmeans(x=cw[,-8], k=3, hc.metric = "euclidean",hc.method = "ward.D2")

12.#确定聚类个数

b<-NbClust(scale(cw[,-8]), distance = "euclidean", min.nc=2,

max.nc=8, method = "complete", index = "all")

13.#确定聚类个数

pamk.best<-pamk(cw[,-8])

14.pamk.best$nc#最佳聚类数

15.pamk.best$pamobject#pam函数最佳运行的输出。

16.v1=data.frame(v[,-1],row.names = v[,1])

resv=hcut(scale(v1), k = 3,stand =TRUE)

17.

## 画图：

1.#绘制热力图

distmap(cc,what="subjects")#对观测值分类

distmap(cc,what="variables")#对变量分类

hm=mix.heatmap(scale(cw[,-8]),rowmar=14)

2.#绘制分层树状图

plot(hh,labels=cw[,8],cex=.6)

id=identify(hh)#手工分成3份，鼠标点三下

plot(ih, main="层次聚类")#画树状图

rect.hclust(ih, k=3)#画出聚类的矩形框

3.fviz\_dend(ihc2, rect = TRUE)#画树形图，并用红色矩形框标出分类情况

4.fviz\_silhouette(ihc2)#画出轮廓图的值

5.fviz\_cluster(ihc2)#主成分聚类的散点图

6.#绘制kmeans聚类图

fviz\_cluster(a,cw[,-8],ellipse.type = "norm")

1. #画出肘方法或弯头法的图

wss=c()

for(k in 1:10)wss[k] = kmeans(x=cw[,-8], centers = k)$tot.withinss

plot(1:10, wss, type="b", xlab="number of clusters",

ylab="Total within-cluster sum of squares",

main="弯头法")

8.#类内平方和

fviz\_nbclust(x=cw[,-8], FUNcluster = kmeans, method="wss")+geom\_vline(xintercept = 3, linetype=2)+labs(subtitle= "Elbow method uder wss")

9.#根据平均轮廓宽度选择最优类个数

fviz\_nbclust(x=cw[,-8], FUNcluster = kmeans, method="gap\_stat")+geom\_vline(xintercept = 3, linetype=2)+labs(subtitle= "Elbow method under gap\_stat")

fviz\_nbclust(w,kmeans,method = "silhouette")+labs(subtitle = "silhouette method")

10.#Gap统计量方法

(z=clusGap(w,FUN=kmeans,5))

plot(z,main = "Gap statistic")

11.#轮廓法

library(cluster)

pr3<-pam(w,3)

si<-silhouette(pr3)

plot(si,col=c("red","green","blue" ))

12.#平均轮廓宽度选择最优聚类个数

fviz\_nbclust(scale(v1),hcut,method = "silhouette")+labs(subtitle = "silhouette method")

13.#分层聚类的树状图、轮廓宽度图、在数据第一、第二主成分的得分图上画出聚类结果

fviz\_dend(resv,rect = TRUE,cex=.3,horiz=T)

fviz\_silhouette(resv)

fviz\_cluster(resv)

1. #围绕中心点的聚类来利用轮廓发确定聚类数目-k中心点法

pamk.best<-pamk(scale(v1))

cat( "estimated number of clusters:",pamk.best$nc,"\n")

plot(pam(v1,pamk.best$nc))

1. #INCA指数

library(ICGE)

d=dist(scale(v[,-1]))

T=vector()

for (i in 2:10)

{part=pam(d,i)$clustering

T[i]=INCAindex(d,part)$Total}

plot(T,type="b" , xlab="Number of clusters" ,ylab="INCA",xlim=c(1.5,10.5))

title("Percentage of objects well classified in the partition")

1. #BIC图，结果示意图

plot(res, what=c("BIC","classification"))

1. #DBSCAN

library(fpc)

set.seed(1010)

par(mfrow=c(2,3))

for(i in c(2,3,4)){

km<- kmeans(wc, i)

plot(wc, pch=km$cluster,col=km$cluster)

title(paste("k-means: k=",i))

}

for(i in c(1,5,7)){

db<- fpc::dbscan(w,eps =0.2,MinPts =i)

plot(wc, pch=db$cluster,col=db$cluster)

title(paste("DBSCAN: Minpts=",i))

}

## 分层聚类（种子）：

### #集群倾向和距离图

cw=read.csv("C:/Users/86133/Desktop/《多元统计分析——R与Python的实现》数据文件/MVAPureData/seeds.csv")  
dim(cw)

## [1] 210 8library(factoextra)

## Loading required package: ggplot2

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

a=get\_clust\_tendency(scale(cw[,-8]),nrow(cw)-1,graph=TRUE)  
a$hopkins\_stat

## [1] 0.8027413

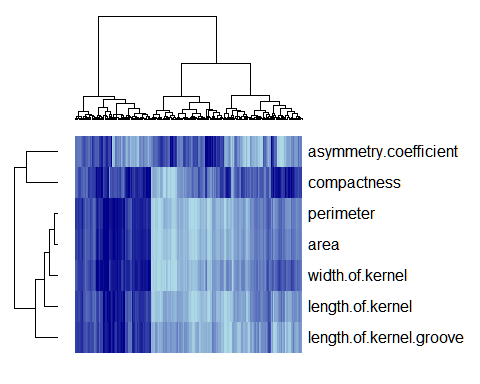
clustertend::hopkins(data=cw[,-8],n=209)

## $H  
## [1] 0.1743926

#绘制热量图  
library(CluMix)

## Registered S3 method overwritten by 'gplots':  
## method from   
## reorder.factor DescTools

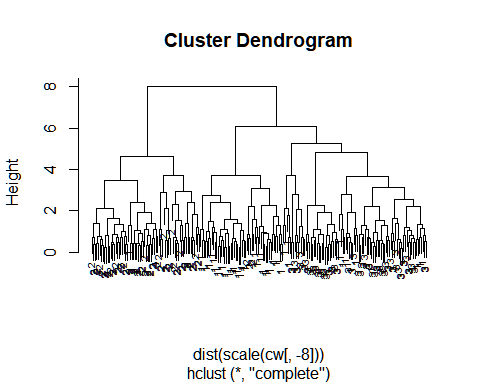
library(extracat)  
hm=mix.heatmap(scale(cw[,-8]),rowmar=14)



### 相关结论：

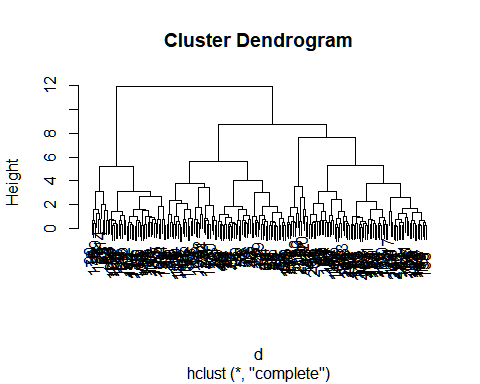
hopkins的统计量为1-0.8027413=0.1972587,有集群倾向

分层聚类，利用完全连接，对观测值进行聚类  
hh=hclust(dist(scale(cw[,-8])),"complete")  
plot(hh,labels=cw[,8],cex=.6)  
id=identify(hh)#手工分成3份，鼠标点三下



### 计算欧氏距离，利用完全连接距离，进行层次聚类

d<-dist(x=cw[,-8],method = "euclidean")  
ih<-hclust(d=d,method = "complete")  
plot(ih)#画树形聚类图



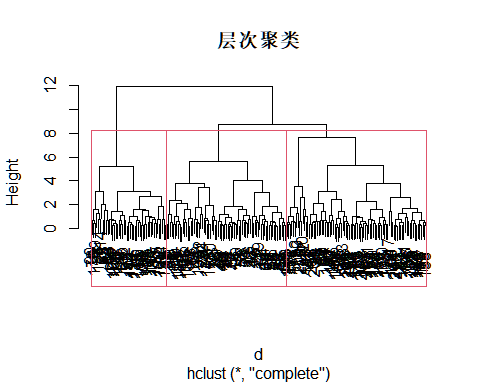
(ihc<-cutree(tree = ih,k=3))#得到每一类的编号

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 2 1 1 2 2 1 1 2 1 1 1 1 1 1  
## [38] 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 1 1 2 1 1 1 3  
## [75] 1 1 1 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 1 3 3 3 3 3 3 1 3 3 3  
## [112] 3 3 3 3 3 3 3 3 3 3 1 1 3 1 3 3 3 3 1 3 3 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2  
## [149] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

ihc1<-aggregate(x=cw[,-8],by=list(cluster=ihc),median)  
ihc1#查看每一类的中位数

## cluster area perimeter compactness length.of.kernel width.of.kernel  
## 1 1 15.260 14.760 0.8796 5.701 3.377  
## 2 2 12.105 13.335 0.8559 5.225 2.879  
## 3 3 18.960 16.420 0.8890 6.259 3.773  
## asymmetry.coefficient length.of.kernel.groove  
## 1 2.956 5.299  
## 2 4.420 5.063  
## 3 3.120 6.148

plot(ih, main="层次聚类")#画树状图  
rect.hclust(ih, k=3)#画出聚类的矩形框



### 相关结论：

利用factoextra包进行层次聚类

ihc2<-factoextra::hcut(x=cw[,-8], k=3, hc\_func = "hclust", hc\_method = "complete",   
 hc\_metric = "euclidean", graph=T)

## Registered S3 method overwritten by 'dendextend':  
## method from   
## rev.hclust vegan

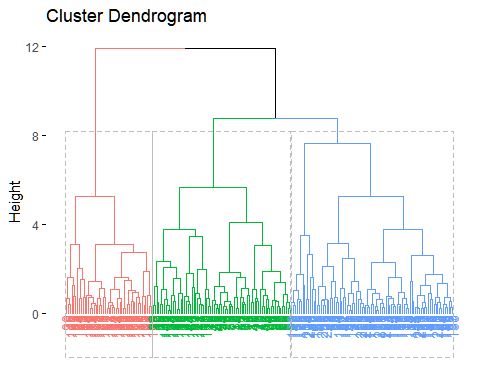
ihc2$cluster

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 2 1 1 2 2 1 1 2 1 1 1 1 1 1  
## [38] 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 1 1 2 1 1 1 3  
## [75] 1 1 1 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 1 3 3 3 3 3 3 1 3 3 3  
## [112] 3 3 3 3 3 3 3 3 3 3 1 1 3 1 3 3 3 3 1 3 3 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2  
## [149] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

ihc2$size

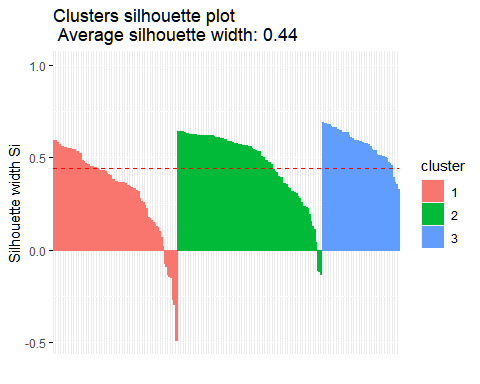
## [1] 75 88 47

fviz\_dend(ihc2, rect = TRUE)#画树形图，并用红色矩形框标出分类情况

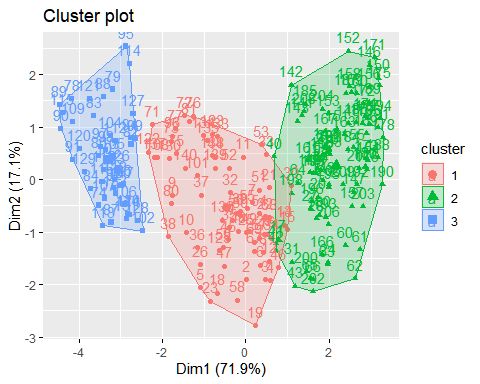


fviz\_silhouette(ihc2)#画出轮廓图的值，有些值是负的，不太理想

## cluster size ave.sil.width  
## 1 1 75 0.33  
## 2 2 88 0.47  
## 3 3 47 0.57

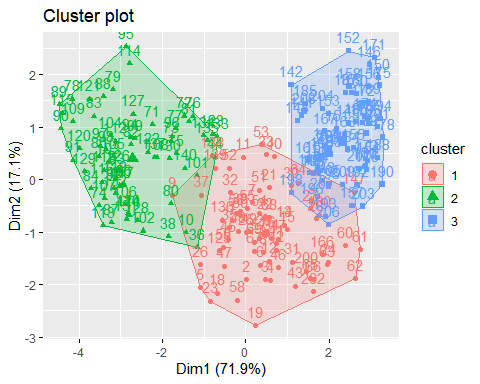


fviz\_cluster(ihc2)#主成分聚类的散点图



#前两个主成分的观测值得分的聚类情况

res<-hcut(scale(cw[,-8]),k=3,stand=TRUE)  
fviz\_cluster(res)



res$cluster

## [1] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1  
## [38] 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 2 2 2 2  
## [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 1 1 2 3 3 3 3 3 3 1 3  
## [149] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3  
## [186] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 3 3 3 3 3 3 3 3

table(res$cluster, cw[,8])

##   
## 1 2 3  
## 1 64 4 5  
## 2 4 66 0  
## 3 2 0 65

### 相关结论：

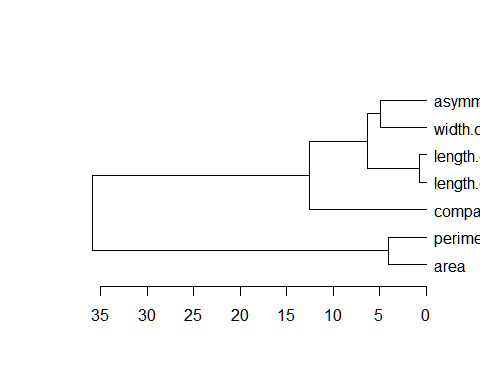
第一类中属于第一类的概率为90.14%

第二类中属于第二类的概率为94.26%

第三类中属于第三类的概率为97.01%

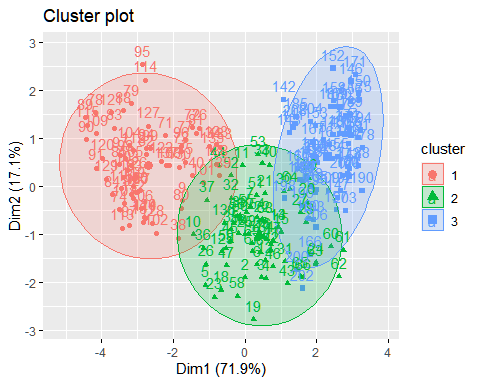
### 利用完全连接距离，对变量聚类

wt=t(cw[,-8])#t表示转置  
ht=hclust(dist(scale(wt)),"complete")  
plot(as.dendrogram(ht),horiz=TRUE)



### k-means

set.seed(1010)  
a=kmeans(scale(cw),3)  
fviz\_cluster(a,cw[,-8],ellipse.type = "norm")



a$cluster

## [1] 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [38] 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1  
## [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 3 3 3 3 3 3 3 3  
## [149] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3  
## [186] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

a$centers#类的中心点

## area perimeter compactness length.of.kernel width.of.kernel  
## 1 1.2098124 1.2212156 0.5160033 1.2062015 1.11261930  
## 2 -0.1878283 -0.2170272 0.3976883 -0.3056103 -0.04082197  
## 3 -1.0219841 -1.0041884 -0.9136916 -0.9005912 -1.07179733  
## asymmetry.coefficient length.of.kernel.groove ID  
## 1 -0.05527907 1.2689066 -0.0349093  
## 2 -0.66847065 -0.6754519 -1.1869160  
## 3 0.72374972 -0.5934547 1.2218253

a$withinss#每类的类内内平方和

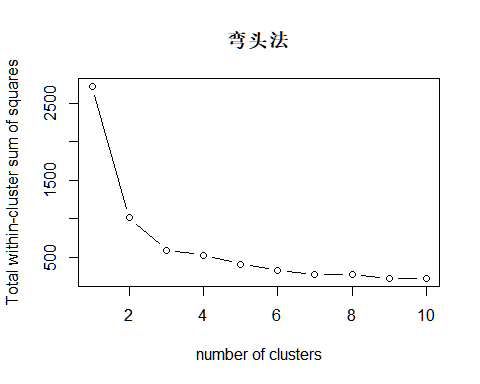
## [1] 157.8478 149.9961 146.6934

a$tot.withinss#所有类的类内内平方和，求和

## [1] 454.5373

### #画出肘方法或弯头法的图

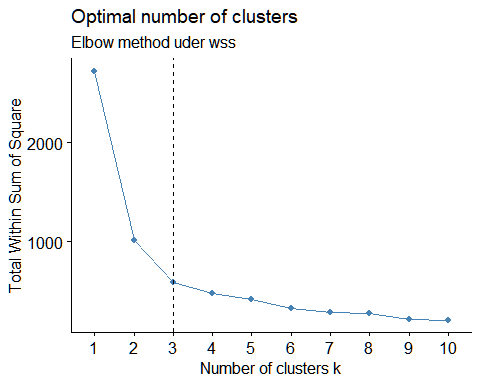
wss=c()  
for(k in 1:10)wss[k] = kmeans(x=cw[,-8], centers = k)$tot.withinss  
plot(1:10, wss, type="b", xlab="number of clusters",   
 ylab="Total within-cluster sum of squares",  
 main="弯头法")



### 相关结论：

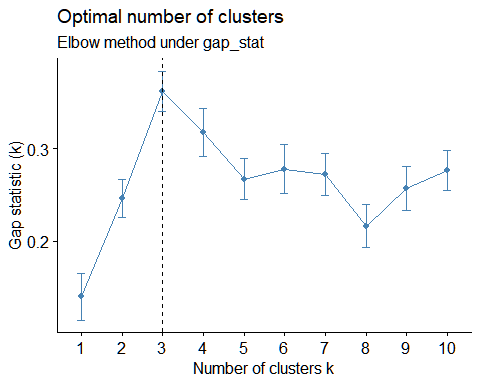
当聚为3类时，tot.withinss变化显著放缓，所以聚为3类

#类内平方和  
fviz\_nbclust(x=cw[,-8], FUNcluster = kmeans, method="wss")+  
 geom\_vline(xintercept = 3, linetype=2)+  
 labs(subtitle= "Elbow method uder wss")



### 根据平均轮廓宽度选择最优类个数

fviz\_nbclust(x=cw[,-8], FUNcluster = kmeans, method="gap\_stat")+  
 geom\_vline(xintercept = 3, linetype=2)+  
 labs(subtitle= "Elbow method under gap\_stat")

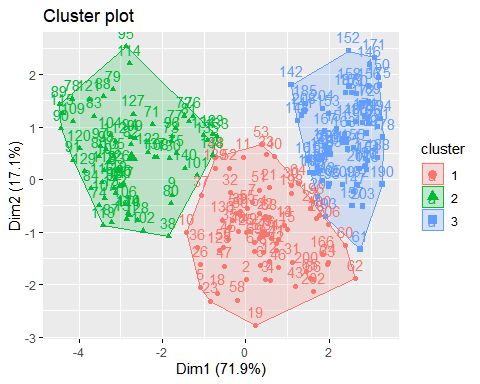


### 相关结论：

聚为3类

### 分层k-means

res.hk<-hkmeans(scale(cw[,-8]),3)  
fviz\_cluster(res.hk)



table(res.hk$cluster, cw[,8])

##   
## 1 2 3  
## 1 65 3 7  
## 2 2 67 0  
## 3 3 0 63

### 相关结论：

第一类中属于第一类的概率为86.67%

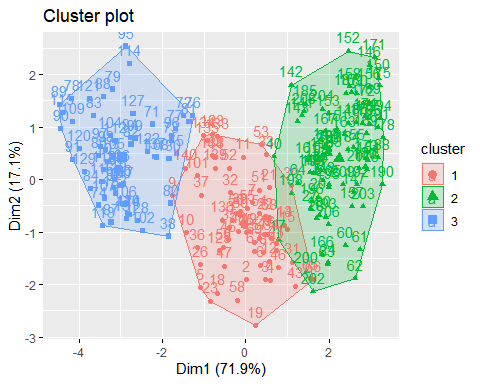
第二类中属于第二类的概率为90.10%

第三类中属于第三类的概率为95.45%

hk<-hkmeans(x=cw[,-8], k=3, hc.metric = "euclidean",hc.method = "ward.D2")  
hk$cluster

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 2 1 1 2 2 1 1 1 1 1 1 1 1 1  
## [38] 3 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 1 1 1 1 1 2 3 3 3 3  
## [75] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3  
## [112] 3 3 3 3 3 3 3 3 3 3 3 1 3 1 3 3 3 3 3 3 3 1 1 1 1 3 1 1 1 2 2 2 2 2 2 2 2  
## [149] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

fviz\_cluster(hk)



table(hk$cluster, cw[,8])

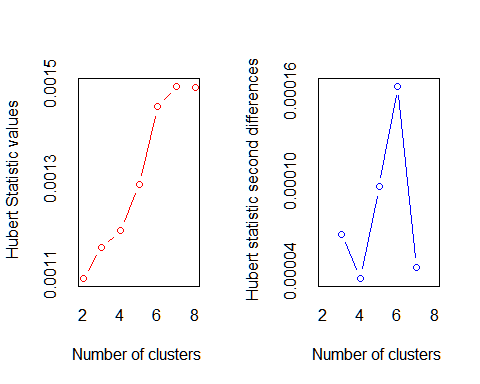
##   
## 1 2 3  
## 1 57 10 0  
## 2 12 0 70  
## 3 1 60 0

### 相关结论：

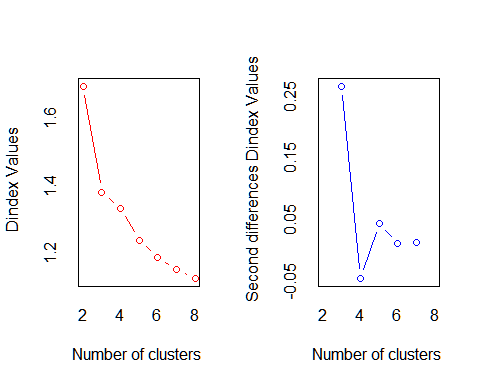
与上述的分层聚类相比分错2个类

### 确定聚类个数

library(NbClust)  
b<-NbClust(scale(cw[,-8]), distance = "euclidean", min.nc=2,  
 max.nc=8, method = "complete", index = "all")



## \*\*\* : The Hubert index is a graphical method of determining the number of clusters.  
## In the plot of Hubert index, we seek a significant knee that corresponds to a   
## significant increase of the value of the measure i.e the significant peak in Hubert  
## index second differences plot.   
##



## \*\*\* : The D index is a graphical method of determining the number of clusters.   
## In the plot of D index, we seek a significant knee (the significant peak in Dindex  
## second differences plot) that corresponds to a significant increase of the value of  
## the measure.   
##   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
## \* Among all indices:   
## \* 7 proposed 2 as the best number of clusters   
## \* 16 proposed 3 as the best number of clusters   
## \* 1 proposed 8 as the best number of clusters   
##   
## \*\*\*\*\* Conclusion \*\*\*\*\*   
##   
## \* According to the majority rule, the best number of clusters is 3   
##   
##   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 相关结论：

聚类的类别为3个

library(fpc)  
pamk.best<-pamk(cw[,-8])  
pamk.best$nc#最佳聚类数

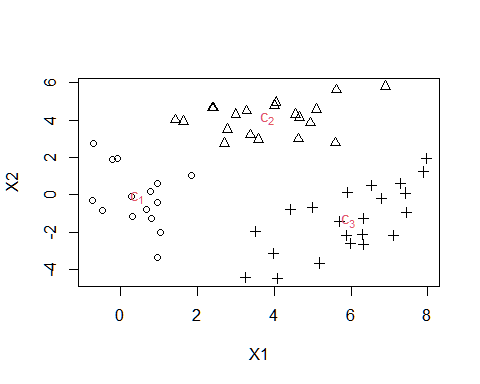
## [1] 2

pamk.best$pamobject#pam函数最佳运行的输出。

## Medoids:  
## ID area perimeter compactness length.of.kernel width.of.kernel  
## [1,] 122 18.14 16.12 0.8772 6.059 3.563  
## [2,] 205 12.37 13.47 0.8567 5.204 2.960  
## asymmetry.coefficient length.of.kernel.groove  
## [1,] 3.619 6.011  
## [2,] 3.919 5.001  
## Clustering vector:  
## [1] 1 2 2 2 1 2 2 2 1 1 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 1 2 2 2 2 2 1 2 2 2 1 1  
## [38] 1 2 2 2 2 2 1 2 2 1 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1  
## [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2  
## [149] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## Objective function:  
## build swap   
## 2.217099 2.020358   
##   
## Available components:  
## [1] "medoids" "id.med" "clustering" "objective" "isolation"   
## [6] "clusinfo" "silinfo" "diss" "call" "data"

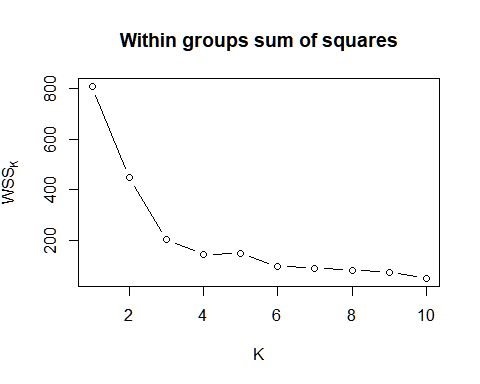
## means聚类：

w=read.csv("C:/Users/86133/Desktop/《多元统计分析——R与Python的实现》数据文件/MVAPureData/kmeansFig.csv")  
a=kmeans(w,3)#确定分成3类  
plot(w ,pch=a$cluster)  
text(a$center ,expression(c[1],c[2],c[3]),col=2)



### **类内平方和wss**

wss=vector()  
for(i in 1:10)wss[i]=kmeans(w,centers=i)$tot.withinss  
plot(1:10,wss,type="b",xlab="K",ylab=expression(WSS[K]),main="Within groups sum of squares")

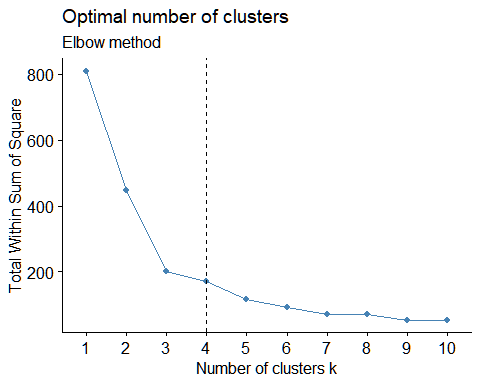


### 相关结论：

k=3处wss变化缓慢，选择3类

library(factoextra)

fviz\_nbclust(w,kmeans, method = "wss")+geom\_vline(xintercept = 4,linetype =2)+labs(subtitle = "Elbow method")



### 相关结论：

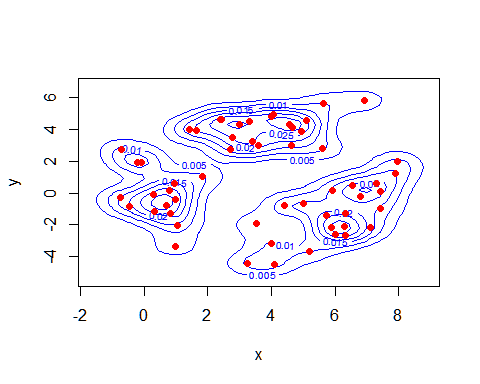
选择4类

### 非参数密度估计图

library(KernSmooth)

## KernSmooth 2.23 loaded  
## Copyright M. P. Wand 1997-2009

est=bkde2D(w, bandwidth=rep(1,2)\*.6)  
contour(est$x1,est$x2,est$fhat, col = "blue",xlab="x" , ylab="y")  
points(w,col = "red",pch = 16)



### 基于模型聚类

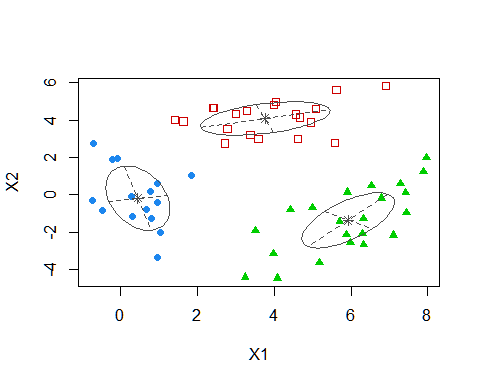
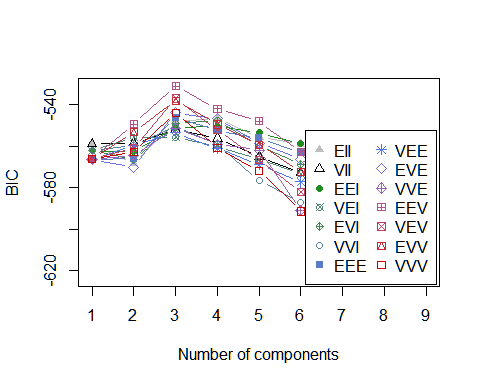
library(mclust)

## Package 'mclust' version 5.4.7  
## Type 'citation("mclust")' for citing this R package in publications.

res =Mclust(w)  
summary(res)

## ----------------------------------------------------   
## Gaussian finite mixture model fitted by EM algorithm   
## ----------------------------------------------------   
##   
## Mclust EEV (ellipsoidal, equal volume and shape) model with 3 components:   
##   
## log-likelihood n df BIC ICL  
## -239.3084 57 13 -531.1766 -532.03  
##   
## Clustering table:  
## 1 2 3   
## 15 20 22

plot(res, what=c("BIC","classification"))#BIC图，结果示意图



### 相关结论：

最适合聚类数目为k=3的EEV模型

#查看其中三类包含的种类

w[res$classification==1,1]

## [1] 0.78773341 0.81867894 -0.20464153 0.95241303 0.96580033 -0.08487109  
## [7] -0.71486499 0.96624080 -0.46230137 0.68605062 -0.73876017 0.31882166  
## [13] 1.05196788 0.28592085 1.83786541

w[res$classification==2,1]

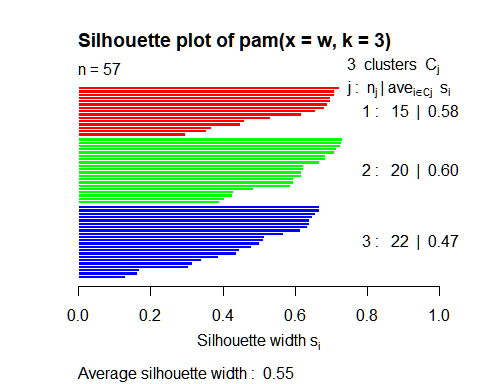
## [1] 1.429204 6.903041 4.626362 4.666472 2.714808 4.557171 3.005091 2.401909  
## [9] 3.384864 1.641833 2.425423 4.050945 3.290588 5.095851 2.783941 5.591346  
## [17] 4.001319 3.592455 4.953661 5.619862

w[res$classification==3,1]

## [1] 7.434011 5.709957 5.992984 7.426577 6.302171 4.082676 4.990624 7.107920  
## [9] 6.332461 7.958556 7.294028 3.988984 7.878695 4.416487 5.915106 3.510812  
## [17] 6.333177 5.186667 6.539969 3.244055 5.878109 6.792990

### 轮廓法

library(cluster)  
pr3<-pam(w,3)  
si<-silhouette(pr3)  
plot(si,col=c("red","green","blue" ))

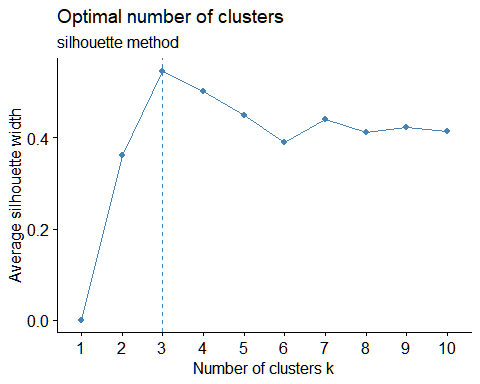


### 相关结论：

分为3类

### 平均轮廓宽度选择最优聚类个数

library(factoextra)  
fviz\_nbclust(w,kmeans,method = "silhouette")+labs(subtitle = "silhouette method")



#最高点为类别

## 人造数据聚类：

### 读取数据、分类及计算距离、画图

ww=read.csv("C:/Users/86133/Desktop/《多元统计分析——R与Python的实现》数据文件/MVAPureData/classclust.csv")  
library(rpart.plot)

## Loading required package: rpart

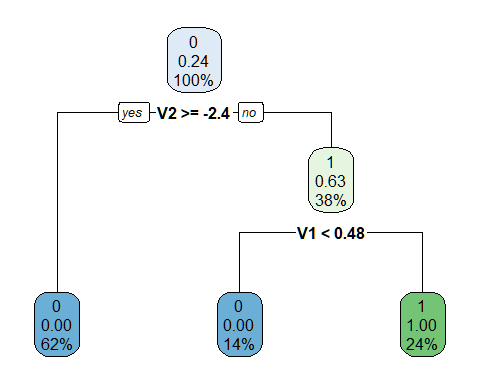
(a=rpart(factor(V3)~. ,ww))

## n= 100   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 100 24 0 (0.7600000 0.2400000)   
## 2) V2>=-2.43056 62 0 0 (1.0000000 0.0000000) \*  
## 3) V2< -2.43056 38 14 1 (0.3684211 0.6315789)   
## 6) V1< 0.4789247 14 0 0 (1.0000000 0.0000000) \*  
## 7) V1>=0.4789247 24 0 1 (0.0000000 1.0000000) \*

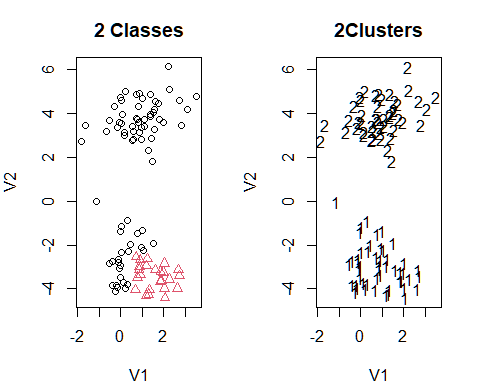
table(factor(ww[,3]),predict(a,ww,type="class"))

##   
## 0 1  
## 0 76 0  
## 1 0 24

rpart.plot(a)



b=kmeans(ww[,-3],2)  
  
par(mar=c(4,4,3,2))  
layout(t(1:2))  
plot(ww[,-3] ,pch=(ww[,3]+1),col=ww[,3]+1)  
title("2 Classes")  
  
plot(ww[,-3],type="n")  
text(ww[,-3],label=b$cluster)  
title("2Clusters")



### 相关结论：

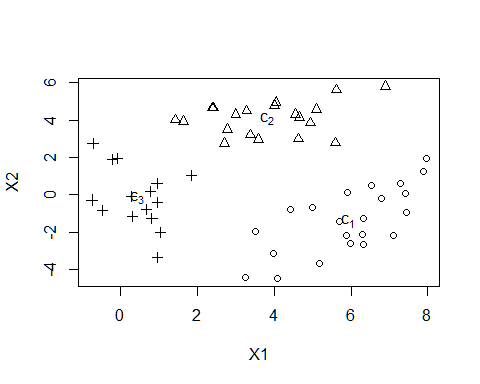
由决策树可知，

图1将数据分类为2类，图2将其聚类为2类

### k-means

w=read.csv("C:/Users/86133/Desktop/《多元统计分析——R与Python的实现》数据文件/MVAPureData/kmeansFig.csv")  
a=kmeans(w,3)#确定分成3类  
plot(w, pch=a$cluster)  
text(a$center , expression(c[1] ,c[2] ,c[3]),co1=2)

## Warning in text.default(a$center, expression(c[1], c[2], c[3]), co1 = 2):  
## "co1"不是图形参数



### 相关结论：

将数据分类为3类

### DBSCAN

wc=read.csv("C:/Users/86133/Desktop/《多元统计分析——R与Python的实现》数据文件/MVAPureData/3g1.csv")  
library(fpc)  
set.seed(1010)  
par(mfrow=c(2,3))  
for(i in c(2,3,4)){  
 km<- kmeans(wc, i)  
 plot(wc, pch=km$cluster,col=km$cluster)  
 title(paste("k-means: k=",i))  
}  
for(i in c(1,5,7)){  
 db<- fpc::dbscan(w,eps =0.2,MinPts =i)  
 plot(wc, pch=db$cluster,col=db$cluster)  
 title(paste("DBSCAN: Minpts=",i))  
 }

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '26'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '27'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '28'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '29'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '30'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '31'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '26'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '27'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '28'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '29'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '30'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '31'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '26'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '27'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '28'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '29'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '30'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '31'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '26'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '27'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '28'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '29'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '30'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '31'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '26'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '27'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '28'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '29'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '30'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '31'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '26'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '27'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '28'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '29'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '30'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '31'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '26'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '27'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '28'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '29'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '30'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '31'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '26'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '27'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '28'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '29'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '30'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '31'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '26'这样一个值

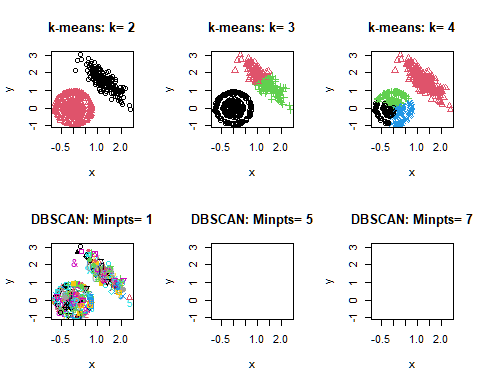
## Warning in plot.xy(xy, type, ...): 目前没有实现pch '27'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '28'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '29'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '30'这样一个值

## Warning in plot.xy(xy, type, ...): 目前没有实现pch '31'这样一个值



### 相关结论：

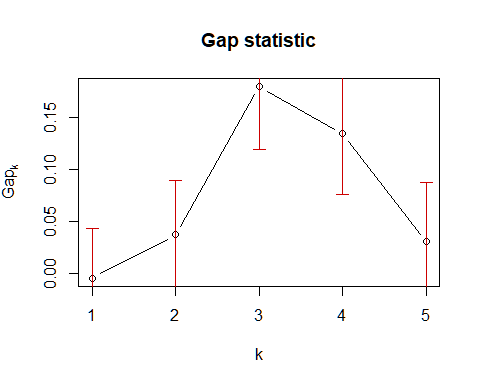
最好分成2类，分成3类和4类都没意义

### Gap统计量方法

library(cluster)  
(z=clusGap(w,FUN=kmeans,5))

## Clustering Gap statistic ["clusGap"] from call:  
## clusGap(x = w, FUNcluster = kmeans, K.max = 5)  
## B=100 simulated reference sets, k = 1..5; spaceH0="scaledPCA"  
## --> Number of clusters (method 'firstSEmax', SE.factor=1): 3  
## logW E.logW gap SE.sim  
## [1,] 4.222013 4.217360 -0.004652755 0.04781745  
## [2,] 3.871177 3.908422 0.037244749 0.05190102  
## [3,] 3.472781 3.652249 0.179468137 0.06040747  
## [4,] 3.319095 3.453090 0.133995030 0.05794706  
## [5,] 3.281521 3.312126 0.030605448 0.05684811

plot(z,main = "Gap statistic")



### 相关结论：

k=3时gap值最大，分为3类

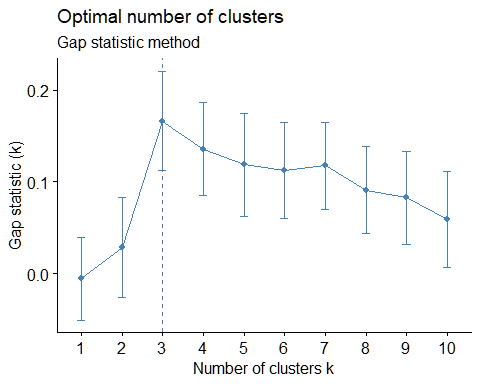
### 平均轮廓宽度选择最优聚类个数

library(factoextra)

## Loading required package: ggplot2

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

fviz\_nbclust(w,kmeans, nstart = 10,method = "gap\_stat", nboot=50)+  
labs(subtitle = "Gap statistic method")



#最高点为类别

### 相关结论：

分为3类