[R 语言入门基础 3](#_Toc520812921)

[1. 统计软件比较 3](#_Toc520812922)

[2. R语言入门和安装 4](#_Toc520812923)

[R的源起 4](#_Toc520812924)

[R 特点 4](#_Toc520812925)

[如何安装、加载包 5](#_Toc520812926)

[3. R的基本数据类型 6](#_Toc520812927)

[R基本的数据类型 6](#_Toc520812928)

[常量 6](#_Toc520812929)

[缺失值NA 7](#_Toc520812930)

[R基本数据类型及其转换 7](#_Toc520812931)

[4. R的数据结构 9](#_Toc520812932)

[R中数据结构 9](#_Toc520812933)

[向量(Vector) 9](#_Toc520812934)

[矩阵(matrix) 14](#_Toc520812935)

[数组(array) 19](#_Toc520812936)

[列表(list) 20](#_Toc520812937)

[因子(factor) 22](#_Toc520812938)

[数据框(data.frame) 24](#_Toc520812939)

[5. 外部数据导入 28](#_Toc520812940)

[读入R包中自带的数据 28](#_Toc520812941)

[读取外部数据 29](#_Toc520812942)

[读取TXT 29](#_Toc520812943)

[读取EXCEL 30](#_Toc520812944)

[SPSS 31](#_Toc520812945)

[SAS 31](#_Toc520812946)

[6. 程序控制 32](#_Toc520812947)

[循环 32](#_Toc520812948)

[条件 32](#_Toc520812949)

[R 进行描述性分析 37](#_Toc520812950)

[1. 变量统计特征探索 37](#_Toc520812951)

[1.1 集中趋势的度量 37](#_Toc520812952)

[1.2 离散程度的度量 44](#_Toc520812953)

[1.3 偏态和峰态的度量 46](#_Toc520812954)

[2. 变量的可视化探索 50](#_Toc520812955)

[图形展示三步 50](#_Toc520812956)

[软件实现 50](#_Toc520812957)

[ggplot2包介绍 57](#_Toc520812958)

[R 进行推断性分析 62](#_Toc520812959)

[1. 参数估计 62](#_Toc520812960)

[总体参数与样本统计量的对比 62](#_Toc520812961)

[抽样估计的过程 62](#_Toc520812962)

[参数估计的分类 62](#_Toc520812963)

[2. 假设检验 67](#_Toc520812964)

[假设检验在统计方法中的地位 67](#_Toc520812965)

[假设检验的过程 68](#_Toc520812966)

[原假设(null hypothesis) 68](#_Toc520812967)

[双侧检验与单侧检验(假设的形式) 68](#_Toc520812968)

[假设检验中的两类错误 69](#_Toc520812969)

[两类错误对数据挖掘的启示 70](#_Toc520812970)

[一个总体均值的检验(作出判断) 70](#_Toc520812971)

[利用P值进行决策 71](#_Toc520812972)

[软件实现 72](#_Toc520812973)

[3. 两个样本检验 73](#_Toc520812974)

[两个总体参数的检验 73](#_Toc520812975)

[两独立样本T检验 73](#_Toc520812976)

[配对样本T检验 75](#_Toc520812977)

[小结 76](#_Toc520812978)

[非参数检验 77](#_Toc520812979)

[非参数检验拓展 78](#_Toc520812980)

[卡方独立性检验 79](#_Toc520812981)

[4. 方差分析 80](#_Toc520812982)

[问题来源 80](#_Toc520812983)

[什么是方差分析 80](#_Toc520812984)

[方差分析的步骤 83](#_Toc520812985)

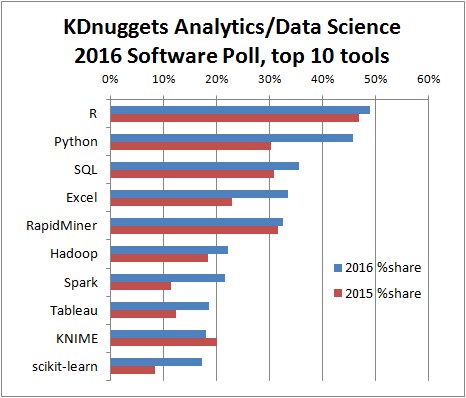
[实例演练 85](#_Toc520812986)

[多重比较软件实现 86](#_Toc520812987)

[R 结语 87](#_Toc520812988)

# R 语言入门基础

## 1. 统计软件比较



<https://www.kdnuggets.com/>

<https://www.kdnuggets.com/2016/06/r-python-top-analytics-data-mining-data-science-software.html>

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 软件名称 | 操作方式 | 介绍 | 是否开源 | 使用场景 | 数据处理能力 | 统计方法丰富程度 | 使用人员 |
| SAS | 编程 | 源于上世纪60年代对生物统计分析的需求, 在医药、金融等对数据分析质量要求较高的行业运用较多 | 否 | 大型商业数据、高复杂度 | 极强, 多表 | 滞后学术前沿2-3年 | 数据工程师 |
| Python | 编程 | 最初设计为科学家提供的一种脚本式语言。该语言粘合性很好, 使其兼顾了简单易用和执行高效的特点。作为程序开发语言被广为接受。 | 是 | 大数据、网络、非结构数据 | 一般, 多表 | 和R相似, 滞后于R | 程序员与大数据处理人员 |
| R | 编程 | 统计分析功能强大, 编程风格简练 | 是 | 小数据、低复杂度 | 一般, 多表 | 紧跟学术前沿 | 有理工或社科类背景研究人员 |
| Matlab | 编程(菜单可以忽略) | 用于科研的分析软件, 以数学建模为主 | 否 | 小数据、低复杂度 | 一般, 多表 | 紧跟学术前沿 | 有理工背景的研究人员 |
| SPSS | 菜单(编程可以忽略) | 早期为满足社会统计的需求, 后改为满足常用商业分析的需求 | 否 | 小型数据、低复杂度 | 弱, 单表 | 基本统计方法 | 业务分析人员 |
| Stata | 编程+菜单 | 专著与学术数据分析 | 否 | 小型数据、低复杂度 | 弱, 单表 | 紧跟社科类学术前沿 | 社科类科研人员 |

使用不同的软件来做统计分析时, 底层的原理都是一样的, 只是语言的实现上有所不同

R & python

It reflects that there are two primary routes into data science: from statistics/science in general where use of R is more common, and from programming/software engineering where python is more common. -------Hadley Wickham

查看安装包的分类

<http://mirrors.ustc.edu.cn/CRAN/web/views/>

可以通过搜索别人的博客知道完成某个分析需要的包

## 2. R语言入门和安装

### R的源起

R是S语言的一种实现。S语言是由 AT&T贝尔实验室开发的一种用来进行数据探索、统计分析、作图的解释型语言。最初S语言的实现版本主要是S-PLUS。SPLUS是一个商业软件, 它基于S语言, 并由MathSoft公司的统计科学部进一步完善。后来Auckland大学的Robert Gentleman 和 Ross Ihaka 及其他志愿人员开发了一个R系统。R的使用与S-PLUS有很多类似之处。

### R 特点

1. 多领域的统计资源

涵盖统计学, 经济学, 社会学, 生物学, 心理学, 空间分析等

1. 跨平台

R可以在UNIX, Windows, Mac OS系统运行

1. 开源软件

运行效率可能没有商业软件高

如何安装R及Rstudio

https://www.r-project.org/

<https://www.rstudio.com/>

Rstudio

1. 开源的IDE, 将常用的窗口整合在一起, 无需彼此之间切换
2. 需要另外安装

### 如何安装、加载包

#######################R基础

#安装包

install.packages("arules")

# 也可以在Rstudio中安装包, 如果自定义了R的安装路径, 要注意在安装包时指定安装到R所在的路径中.

install.packages("sampling", lib="C:/R/R-3.4.4/library")

# 更新包

update.packages("arules")

#加载包, 必须要先安装才能加载, 必须在加载后才能使用包

library(arules)

#require(arules)

ls("package:arules")

#卸载包, 从运行环境中移除包

detach("package:arules")

#删除包

remove.packages("arules")

# 查看所有已经安装的包, 以列表方式显示包名

t1 <- .packages(all.available=**T**)

t1

#查看已安装的包, 显示包的简要信息

t2 <- installed.packages()

t2

# 查看所有可以安装的包

t <- available.packages()

# 查看所有可安装的包的数量,nrow查看行数

nrow(t)

# 查看所有已加载的包

sessionInfo()

# 查看帮助信息, value即为输出的内容

?glm

?mean

?factor

help(options)

# Displays the argument names and corresponding default values of a function or primitive. 查看函数表达式及其参数名和相应的默认值

args(glm)

# Get or set the formal arguments of a function.查看或设置某个函数所有的参数

formals(glm)

#查看函数的实例

example(glm)

# 列出所有已安装包自带的数据集

data()

# 只查看arules包自带的数据集

data(package="arules")

## 3. R的基本数据类型

### R基本的数据类型

数值型(numeric)

字符型(character)

逻辑型(logical)

复数型(complex)

原始型(raw)

默认值(missing value)

### 常量

• 常量为: 数值型、逻辑型和字符型三种,

• 数值型数据又可以分为整型、单精度、双精度等

例如, 123, 123.45, 1.2345e30 是数值型常量；"Weight", "李明"是字符型；

逻辑假值写为F或FALSE。逻辑真值写为T或TRUE(注意区分大小写, 写t或true都没意义) .

函数的首字母大写, 其它字母都小写

### 缺失值NA

• 数据可以取缺失值, 用符号NA代表缺失值, not available

• 任何数与缺失值的运算结果仍为缺失值。

x <- c(1, **NA**, 2)

x

*# 任何数与NA相乘为NA*

2\*x

*# [1] 2 NA 4*

*# is.na(x) 返回x是否缺失值(返回值T或F)*

is.na(x)

*# [1] FALSE TRUE FALSE*

*# 任何数与缺失值NA的运算结果仍为缺失值*

sum(x)

*# [1] NA*

### R基本数据类型及其转换

|  |  |  |
| --- | --- | --- |
| 数据类型 | 辨别函数 | 转换函数 |
| character | is.character() | as.character() |
| complex | is.complex() | as.complex() |
| double | is.double() | as.double() |
| integer | is.integer() | as.integer() |
| logical | is.logical() | as.logical() |
| NA | is.na() | as.na() |
| numeric | is.numeric() | as.numeric() |

可以认为double和integer是numeric的两种特殊情况.

?mode

*# Get or set the type or storage mode of an object.# mode(x)*

*# mode(x) <- value*

*# storage.mode(x)*

*# storage.mode(x) <- value*

x <-3

x

*# 查看基本的数据类型*

mode(x)

*# [1] "numeric"*

*# 判断是否为整型*

is.integer(x)

*# [1] FALSE*

*# 转换为整型, 只是把结果输出为整形的, 并不会在原来的x上进行修改*

x1 <- as.integer(x)

x1

is.integer(x1)

*# [1] TRUE*

*# 字符型, 单引号与双引号皆可*

x <- "你好"

x <- '你好'

mode(x)

*# [1] "character"*

*# 逻辑型*

x <- **TRUE**

mode(x)

*# [1] "logical"*

*# NA的数据类型为logical*

x <- **NA**

mode(x)

*# [1] "logical"*

is.na(x)

*# [1] TRUE*

## 4. R的数据结构

### R中数据结构

• 向量(vector)

• 数组(array)

• 列表(list)

• 因子(factor)

• 矩阵(matrix)

• 数据框(data.frame)

**# ctrl+shift+c进行注释**

### 向量(Vector)

#### 向量定义

• 向量是有相同基本类型的元素序列, 即一维数组, 但使用不同的基本类型也不会报错

• 使用函数c() 定义, 它把若干个数值或字符串组合为一个向量

# 定义向量

# 混用不用的基本类型也不会报错,会自动进行基本数据类型的转换

x<-c("vb",1,**NA**,3)

x

# [1] "vb" "1" NA "3"

# 查看向量的数据类型

class(x)

# [1] "character"

# 一维数值型向量

x<-c(1,2,3)

x

class(x)

# [1] "numeric"

x <- c(1:3, 10:13)

x

# [1] 1 2 3 10 11 12 13

# 字符串向量

x1<-c("A","B","C")

x1

class(x1)

# 逻辑型向量

x2<-c(**TRUE**,**FALSE**,**TRUE**,**FALSE**)

x2<-c(**T**,**F**,**T**,**F**)

x2

class(x2)

#### 向量元素的调用

• 访问1个元素: 用x[i]的格式, x[4], 中括号[]表示调用

• 访问多个元素: x[c(1, 2, 1) ], x[-3], x[-c(1, 3) ], x[x>10]

# 调用元素

x <- c(1:3, 10:13)

x

# [1] 1 2 3 10 11 12 13

x[4]

# [1] 10

# 同时调用多个元素

x[c(1,2,4)]

# [1] 1 2 10

# 错误的写法, 必须要把元素位置写在向量中. x[1,2]表示二维向量

# x[1,2]

# 排除第3个元素

x[-3]

# [1] 1 2 10 11 12 13

# 排除多个元素

x[-c(1,3)]

# [1] 2 10 11 12 13

# 取x中的每个元素与10比较大小,得到与x等长的逻辑型向量

x>10

# [1] FALSE FALSE FALSE FALSE TRUE TRUE TRUE

# 把x>10的结果作为参数, 只输出值大于10的元素

x[x>10]

# [1] 11 12 13

#### 向量的四则运算

• 可以对向量进行加(＋) 减(－) 乘(\*) 除(/) 、乘方(^) 运

算, 其含意是对向量的每一个元素进行运算

# 两个向量的四则运算是对应位置上元素的运算, 两个向量的长度必须相同

x<-c(1,2,3)

y<-c(4,5,6)

x\*y

# [1] 4 10 18

x/y

# [1] 0.25 0.40 0.50

# 向量的平方等于两个向量相乘

x^2

# [1] 1 4 9

y^x

# [1] 4 25 216

#### 向量的其它运算

• 可以用向量作为函数自变量

函数length(x) , sqrt、log、exp、sin、cos、tan等函数都可以用向量作自变量, 结果是对向量的每一个元素取相应的函数值

min, max, sd, sum, var, sort, order

• which.min() which.max()

t<-c(3,1,5,9,7)

t

length(t)

mode(t)

*# 返回最小值*

min(t)

*# 返回最大值*

max(t)

*# 返回极差,数据的分布区间*

range(t)

*# 返回向量中的最大值的位置*

which.max(t)

*# [1] 4*

*# 返回向量中的最大值*

t[which.max(t)]

*# [1] 9*

which.min(t)

*# [1] 2*

t[which.min(t)]

*# [1] 1*

#### 向量的常用函数

|  |  |  |  |
| --- | --- | --- | --- |
| 函数 | 用途 | 实例(X<-c(1, 3, 5, 4) ) | 结果 |
| rev() | 反排序 | rev(X) | 4 5 3 1 |
| rank() | 求秩 | rank(x) | 1 2 4 3 |
| append(x, values, after=length(x) ) | 追加 | append(X, 7, after=2) | 1 3 7 5 4 |
| replace(x, list, values) | 替换 | replace(X, c(1, 4) , c(8, 8) ) | 8 3 5 8 |
| paste(, sep=) | 连接 | paste("y", 1:5, sep="") | "y1" "y2" "y3" "y4" "y5" |
| match(x, table) /pmatch() | 匹配/部分匹配 | match(c(1, 2) , X) | 1 NA |
| all() | 判断所有 | all(X>c(2, 3, 4, 3) ) | FALSE |
| any() | 判断部分 | any(X>c(2, 3, 4, 3) ) | TRUE |

rank() 返回从小到大的元素的位置

paste("y", 1:5, sep="") 把"y"与向量1:5中的所有元素进行连接, 得到长度为5的字符串向量

match(x, table) /pmatch() 类似于excel中的Vlookup函数

x<-c(1,3,5,4)

*# 反排序*

rev(x)

*# [1] 4 5 3 1*

*# 求秩,返回元素在向量中从小到大排序的位置*

rank(x)

*# [1] 1 2 4 3*

append(x,7,after=2)

*# [1] 1 3 7 5 4*

*# 把向量x的第1,4号元素替换为8,8*

replace(x,c(1,4),c(8,8))

*# [1] 8 3 5 8*

*# 把"y"与向量1:5中的所有元素进行连接, 得到长度为5的字符串向量*

paste("y",1:5,sep="")

*# [1] "y1" "y2" "y3" "y4" "y5"*

*# 从x中查找c(4,5,7)中的元素,返回匹配的元素在x中的位置,查找不到的元素返回NA*

match(c(4,5,7),x)

*# [1] 4 3 NA*

*# x中对应位置上的元素是否全部大于c(2,3,4,3)对应位置的元素*

all(x>c(2,3,4,3))

*# [1] FALSE*

*# x中对应位置上的元素是否至少有1个大于c(2,3,4,3)对应位置的元素*

any(x>c(2,3,4,3))

*# [1] TRUE*

#### 产生有规律的数列

• 等差数列

1:n产生从1到n的整数列

1:n-1不是代表1到n-1, 而是先生成向量1:n, 再用1:n中的每个元素减去1, 得到1个新的向量

1:3-1

[1] 0 1 2

seq(from=0, to=2, by=0.7)

# 产生有规律的数列

?seq

# seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)), length.out = NULL, along.with = NULL, ...)

t<-1:10

# [1] 1 2 3 4 5 6 7 8 9 10

t1 <- 1:10-1

# [1] 0 1 2 3 4 5 6 7 8 9

t<-seq(1,10,by=2)

# [1] 1 3 5 7 9

t<-seq(1,by=3,length=10)

# [1] 1 4 7 10 13 16 19 22 25 28

seq(from=0, to=2, by=0.7)

# [1] 0.0 0.7 1.4

对x重复多少次, 返回1个新的向量

rep(x, times, each=)

# 重复向量c(1:3)4次, 得到一个新向量

rep(c(1:3),4)

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

# 对向量c(1:3)中的每个元素都重复4次, 得到一个新向量

rep(c(1:3),each=4)

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

# 对第1个向量c(1:3)中的元素分别重复第2个向量c(1:3)中的对应位置的元素次数

rep(c(1:3),c(1:3))

# [1] 1 2 2 3 3 3

numeric(n) 可以产生一个长度为n的零向量, 作用就是在内存中开辟一个定长的空间, 方便后面的运行.

t<-numeric(8)

t

# [1] 0 0 0 0 0 0 0 0

length(t)

#### 向量取子集

m<-seq(1,20,by=2)

m

# [1] 1 3 5 7 9 11 13 15 17 19

m[m>10]

# [1] 11 13 15 17 19

subset(m,m>5)

# [1] 7 9 11 13 15 17 19

m[which(m>5)]

# [1] 7 9 11 13 15 17 19

### 矩阵(matrix)

#### 矩阵定义

矩阵是数据用行和列排列的长方形表格, 每个单元格是相同的数据类型.矩阵是二维数组, 应用广泛

matrix(data=NA, nrow=, ncol=, byrow=FALSE, dimnames=NULL)

data项为必要的矩阵因素

nrow表示行数

ncol表示列数

dimnames表示行和列的名称

byrow表示排列元素时是否按照行进行, 默认为是FALSE, 即按列排列, 先排第1行, 再排列第2行, 如此下去

# 矩阵

A <- matrix(1:12, nrow=4, ncol=3, byrow=**T**)

A

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

# [1,] 1 2 3

# [2,] 4 5 6

# [3,] 7 8 9

# [4,] 10 11 12

A <- matrix(1:12, nrow=4, ncol=3)

A

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

# [1,] 1 5 9

# [2,] 2 6 10

# [3,] 3 7 11

# [4,] 4 8 12

# 当向量c(1,0)中的元素个数不够时, 就重复c(1,0)中的元素

B <- matrix(c(1,0), nrow=2, ncol=3, byrow=**T**)

B

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

# [1,] 1 0 1

# [2,] 0 1 0

矩阵两个维: 行维和列维。比如, 设x为2行3列矩阵, 它的行维可以定义一个长度为2的字符向量作为每行的名字, 它的列维可以定义一个长度为3的向量作为每列的名字, 属性dimnames是一个列表, 列表的每个成员是一个维名字的字符向量或NULL。

例如:

# 定义3行2列的矩阵, 给列和行指定名称, dimnames先指定行的名称, 再指定列的名称

x <- matrix(1:6, ncol=2, dimnames=list(c("one", "two", "three"), c("First", "Second")), byrow=**T**)

x

# First Second

# one 1 2

# two 3 4

# three 5 6

#### 矩阵元素的调用

A[i, j]得到第i行第j列的元素

A[i, ]表示第i行的元素

A[, j]表示第j列的元素

A <- matrix(1:12,4,3)

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

*# [1,] 1 5 9*

*# [2,] 2 6 10*

*# [3,] 3 7 11*

*# [4,] 4 8 12*

*# 返回A矩阵的转置矩阵*

t(A)

*# [,1] [,2] [,3] [,4]*

*# [1,] 1 2 3 4*

*# [2,] 5 6 7 8*

*# [3,] 9 10 11 12*

*# 返回矩阵A的第3行2列的元素*

A[3,2]

*# [1] 7*

*# 返回矩阵A第3行的元素*

A[3,]

*# [1] 3 7 11*

A[,2]

*# [1] 5 6 7 8*

*# 返回多少行*

nrow(A)

*# [1] 4*

*# 返回多少列*

ncol(A)

*# [1] 3*

*# 维度(行,列)*

dim(A)

*# [1] 4 3*

#### 矩阵运算

• 函数t(A) 返回矩阵A的转置。nrow(A) 为矩阵A的行数, ncol(A) 为矩阵A的列数。

• 矩阵之间进行普通的加减乘除四则运算, 即数组的对应元素之间进行运算, 所以注意A\*B不是矩阵乘法而是矩阵对应元素相乘。

• 要进行矩阵乘法, 使用运算符%\*%, A%\*%B表示矩阵A乘以矩阵B(当然要求A的列数等于B的行数) 。

• 其它矩阵运算还有solve(A, b) 解线性方程组 Ax= b, solve(A) 求方阵A的逆矩阵, svd() 计算奇异值分解, qr() 计算QR分解, eigen() 计算特征向量和特征值。详见随机帮助, 例如:

• > ?qr

• 函数diag() 的作用依赖于其自变量。

• diag(vector) 返回以自变量(向量) 为主对角元素的对角矩阵。

• diag(matrix) 返回由矩阵的主对角元素组成的向量。

• diag(k) (k为标量) 返回k阶单位阵。

t <- c(1,2,3,4)

*# diag(vector) 返回以自变量(向量) 为主对角元素的对角矩阵*

diag(t)

*# [,1] [,2] [,3] [,4]*

*# [1,] 1 0 0 0*

*# [2,] 0 2 0 0*

*# [3,] 0 0 3 0*

*# [4,] 0 0 0 4*

A <- matrix(1:12,4,3)

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

*# [1,] 1 5 9*

*# [2,] 2 6 10*

*# [3,] 3 7 11*

*# [4,] 4 8 12*

*# diag(matrix) 返回由矩阵的主对角元素组成的向量*

diag(A)

*# [1] 1 6 11*

*# diag(k) (k为标量) 返回k阶单位阵。*

diag(4)

*# [,1] [,2] [,3] [,4]*

*# [1,] 1 0 0 0*

*# [2,] 0 1 0 0*

*# [3,] 0 0 1 0*

*# [4,] 0 0 0 1*

#### 矩阵合并与拉直

可以看成是矩阵的横向追加和纵向追加

• 函数cbind() 把其自变量横向拼成一个大矩阵, rbind() 把其自变量纵向拼成一个大矩阵。

• cbind() 的自变量是矩阵或者看作列向量的向量, 自变量的高度应该相等(对于向量, 高度即长度, 对于矩阵, 高度即行数) 。

• rbind的自变量是矩阵或看作行向量的向量, 自变量的宽度应该相等(对于向量, 宽度即长度, 对于矩阵, 宽度即列数) 。

• 如果参与合并的自变量比其它自变量短则循环补足后合并。

例如:

A <- matrix(1:4,nrow=2,ncol=2,byrow=**T**)

*# [,1] [,2]*

*# [1,] 1 2*

*# [2,] 3 4*

B <- matrix(5:8,nrow=2,ncol=2,byrow=**T**)

*# [,1] [,2]*

*# [1,] 5 6*

*# [2,] 7 8*

*# 把AB按行合并起来, AB上下拼接到一起, 列数必须相同, 否则会报错*

rbind(A,B)

*# [,1] [,2]*

*# [1,] 1 2*

*# [2,] 3 4*

*# [3,] 5 6*

*# [4,] 7 8*

*# 把AB按列合并起来, AB左右拼接到一起, 行数必须相同*

cbind(A,B)

*# [,1] [,2] [,3] [,4]*

*# [1,] 1 2 5 6*

*# [2,] 3 4 7 8*

A <- matrix(1:6,nrow=2,ncol=3,byrow=**T**)

cbind(A,B)

*# 报错rbind即按行合并, 上下合并, 列数必须相同*

rbind(A,B)

*# 矩阵与一个数按列合并, 在原矩阵的右边或左边追加1列*

cbind(A,3)

*# [,1] [,2] [,3] [,4]*

*# [1,] 1 2 3 3*

*# [2,] 4 5 6 3*

*# 矩阵与一个数按行合并, 在原矩阵的下边或上边追加1行*

rbind(A,5)

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

*# [1,] 1 2 3*

*# [2,] 4 5 6*

*# [3,] 5 5 5*

### 数组(array)

数组既可以看成是带有多个下标的且类型相同的元素的集合, 也可以看做是向量和矩阵的推广, 一维数组就是向量, 二维数组就是矩阵。

• array() 函数

array(x, dim=length(x) , dimnames=NULL) ,

x是第一自变量, 表示数组的元素值组成的向量。

dim为维度参数, 可省略, 省略时作为一维数组。

dimnames属性可以省略, 不省略时是一个长度与维数相同的列表(list, 见后面) , 列表的每个成员为一维的名字。

• 调用方式类似矩阵, 如y[1,2,3]

> y <- array(1:18, dim = c(2, 3, 3) )

> y

, , 1

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

[1, ] 1 3 5

[2, ] 2 4 6

, , 2

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

[1, ] 7 9 11

[2, ] 8 10 12

, , 3

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

[1, ] 13 15 17

[2, ] 14 16 18

### 列表(list)

#### 列表(list) 定义

列表是一种特别的对象集合, 它的元素也由序号(下标) 区分, 但是各元素的类型可以是任意对象, 不同元素不必是同一类型。元素本身允许是其它复杂数据类型, 比如, 列表的一个元素也允许是列表。

list(变量1=分量1, 变量2=分量2)

x<-c(1,2)

y<-c("as","asds")

LST <- list(class=x,sex=y)

LST

*# $class*

*# [1] 1 2*

*#*

*# $sex*

*# [1] "as" "asds"*

*# list中每个元素的类型可以不同*

m <-list(name="张三丰",age=80,scores=c(85,76,90))

m

#### 列表元素的引用

• 列表元素用"列表名[[下标]]"的格式引用。

• 列表不同于向量, 我们每次只能引用一个元素。

• 注意: "列表名[下标]"或"列表名[下标范围]"的用法也是合法的, 但其意义与用两重括号的记法完全不同, 两重记号取出列表的一个元素, 结果与列表中该元素类型相同, 如果使用一重括号, 则结果是列表的一个子列表(结果类型仍为列表) 。

#### 命名引用

m <-list(name="张三丰",age=80,scores=c(85,76,90))

*# $name*

*# [1] "张三丰"*

*#*

*# $age*

*# [1] 80*

*#*

*# $scores*

*# [1] 85 76 90*

m[[1]]

*# [1] "张三丰"*

class(m[[1]])

*# [1] "character"*

m[1]

*# $name*

*# [1] "张三丰"*

class(m[1])

*# [1] "list"*

m[[3]]

*# [1] 85 76 90*

class(m[[3]])

*# [1] "numeric"*

m[3]

*# $scores*

*# [1] 85 76 90*

class(m[3])

*# [1] "list"*

m[[3]][2]

*# [1] 76*

#### 位置引用

m['name']

*# $name*

*# [1] "张三丰"*

class(m['name'])

*# [1] "list"*

m$name

*# [1] "张三丰"*

m$scores

*# [1] 85 76 90*

#### 修改列表

• 列表的元素可以修改, 只要把元素引用赋值即可。

• 如果被赋值的元素原来不存在, 则列表延伸以包含该新元素。

m$age <- 45

# 可以把一个列表元素修改为任何类型

m$age <- c(19, 29, 31)

# 给列表添加变量

m$sex <- "男"

m[[5]] <- 161

m

*# $name*

*# [1] "张三丰"*

*#*

*# $age*

*# [1] 19 29 31*

*#*

*# $scores*

*# [1] 85 76 90*

*#*

*# $sex*

*# [1] "男"*

*#*

*# [[5]]*

*# [1] 161*

### 因子(factor)

对应分类型数据, 因子是一种特殊的字符型向量, 其中每一个元素取一组离散值中的一个, 而因子对象有一个特殊属性levels表示这组离散值。

定义

factor(x=character() , levels, lables="", exclude=NA, ordered=is.ordered(x) )

# 定义有2个level的因子, 因子的标签分别为female和male, 1表示female, 2表示male

f<-factor(c(1,2,1,2),levels=c(1,2),labels=c('female','male'))

f

# [1] female male female male

# Levels: female male

levels(f)

# [1] "female" "male"

• 可以自行指定各离散取值水平(levels) , 不指定时由x的不同值来求得。

• labels可以用来指定各水平的标签, 不指定时用各离散取值的对应字符串。

• exclude参数用来指定要转换为缺失值(NA) 的元素值集合。如果指定了levels, 则因子的第i个元素当它等于水平中第j个时元素值取"j", 如果它的值没有出现在levels中则对应因子元素值取NA。

• ordered取真值时表示因子水平是有次序的(按编码次序) 。

• 可以用is.factor() 检验对象是否因子, 用as.factor() 把一个向量转换成一个因子。

# 定义有2个level的因子, 因子的标签分别为female和male, 1表示female, 2表示male

f<-factor(c(1,2,1,2),levels=c(1,2),labels=c('female','male'))

f

# [1] female male female male

# Levels: female male

levels(f)

# [1] "female" "male"

x <- c(1,0,1,1,0)

# 提取x中不同的值

unique(x)

# [1] 1 0

# sort默认从小到大排序

sort(unique(x))

# [1] 0 1

# 从大到小排序

sort(unique(x),decreasing=**T**)

# [1] 1 0

# exclue=NA, 不排除NA, order=F表示不是有序的因子

x=c(1,0,1,1,**NA**,0)

y <- factor(x,levels=sort(unique(x),decreasing=**T**),

labels=c("男", "女"),exclude=**NA**,order=**F**)

y

# [1] 男 女 男 男 <NA> 女

# Levels: 男 女

x=c(1,0,1,1,**NA**,0)

# order=F表示有序的因子,按编码次序

y=factor(x,levels=c(1,0),labels=c("男", "女"),exclude=**NA**,order=**T**)

y

# [1] 男 女 男 男 <NA> 女

# Levels: 男 < 女

• 因子的基本统计是频数统计, 用函数table() 来计数。

sex = factor(c("男", "女", "男", "男", "女"))

sex

# [1] 男 女 男 男 女

# Levels: 男 女

table(sex)

res.tab <- table(sex)

res.tab

# sex

# 男 女

# 3 2

table() 的结果是一个带元素名的向量, 元素名为因子水平, 元素值为该水平出现的频数。

• 可以用两个或多个因子进行交叉分类。比如, 性别(sex) 和职业(job) 交叉分组可以用table(sex, job) 来统计每一交叉类的频数, 结果为一个矩阵, 矩阵带有行名和列名, 分别为两个因子的各水平名。

• 因子可以用来作为另外的同长度变量的分类变量。比如, 假设上面的sex是5个学生的性别, 而

> h <- c(165, 170, 168, 172, 159) 是这5个学生的身高, 则> tapply(h, sex, mean)

可以求按性别分类的身高平均值。

sex = factor(c("男", "女", "男", "男", "女"))

h <- c(165, 170, 168, 172, 159)

# 对h变量使用sex因子进行分组, 并求分组后的平均值

tapply(h, sex, mean)

# 男 女

# 168.3333 164.5000

### 数据框(data.frame)

• 数据框是R中类似SAS, SPSS数据集的一种数据结构。它通常是矩阵形式的数据, 但各列可以是不同类型的。数据框每列是一个变量, 每行是一个样品的观测值。

• 数据框的主要用途是保存统计建模需要的数据。统计建模功能都需要以数据框为输入数据。

• 是一种特殊的列表对象, class 属性为"data.frame" , 列表成员必须是向量(数值型、字符型、逻辑型) 、因子、数值型矩阵、列表, 或其它数据框。

• 可以用矩阵形式显示, 可以用对矩阵的下标引用方法来引用其元素或子集。

#### 数据框定义

• 数据框可以用data.frame() 函数生成, 其用法与list() 函数相同, 各自变量变成数据框的成分, 自变量可以命名, 成为变量名。例如:

d <- data.frame(name=c("赵", "钱", "孙", "李", "王"),

age=c(20,21,22,21,20),

height=c(170,171,175,165,181),

gender=c("男", "女", "男", "女", "男" ))

d

*# name age height gender*

*# 1 赵 20 170 男*

*# 2 钱 21 171 女*

*# 3 孙 22 175 男*

*# 4 李 21 165 女*

*# 5 王 20 181 男*

• 如果一个列表的各个成分满足数据框成分的要求, 可以用as.data.frame() 函数强制转换为数据框。比如, 上面的d如果先用list() 函数定义成了一个列表, 就可以强制成为一个数据框。

• 一个矩阵可以用data.frame() 转换为一个数据框, 如果它原来有列名则其列名被作为数据框的变量名, 否则系统自动为矩阵的各列起一个变量名(如X1, X2) 。

#### 数据框引用

• 引用一: 与引用矩阵元素的方法相同, 可以使用下标或下标向量, 也可以使用名字或名字向量。如d[1:2, 2:3]。

• 引用二: 数据框的各变量也可以用按列表引用(即用双括号[[]] 或$符号引用) 。

• 数据框的变量名由属性names定义, 数据框的各行也可以定义名字, 可以用rownames属性定义。如:

d <- data.frame(name=c("赵", "钱", "孙", "李", "王"),

age=c(20,21,22,21,20),

height=c(170,171,175,165,181),

gender=c("男", "女", "男", "女", "男" ))

d

*# name age height gender*

*# 1 赵 20 170 男*

*# 2 钱 21 171 女*

*# 3 孙 22 175 男*

*# 4 李 21 165 女*

*# 5 王 20 181 男*

*# 类似于矩阵的引用方式*

d[1,2]

*# [1] 20*

d[3,3]

*# [1] 175*

*# 类似于列表的引用方式*

d$name

d[['name']]

d[[1]]

*# [1] 赵 钱 孙 李 王*

*# Levels: 李 钱 孙 王 赵*

*# 使用name来引用*

names(d)

*# [1] "name" "age" "height" "gender"*

rownames(d)

*# [1] "1" "2" "3" "4" "5"*

colnames(d)

*# [1] "name" "age" "height" "gender"*

*# 查看数据集的第4列, 即gender的内容*

table(d[4])

*# 男 女*

*# 3 2*

table(d[[4]])

*# 男 女*

*# 3 2*

*# 第3列元素按照第4列元素来进行分组, 并对每一组求均值. 即把身高按照年龄进行分组, 并求均值*

tapply(d[[3]],d[[4]], mean)

*# 男 女*

*# 175.3333 168.0000*

*# 报错, d[3],d[4]为列表, 必须要使用[[]]的引用方式*

tapply(d[3],d[4], mean)

*# Error in tapply(d[3], d[4], mean) : 参数的长度必需相同*

d[[3]]/d[[2]]

*# [1] 8.500000 8.142857 7.954545 7.857143 9.050000*

#### attach() 函数

• 在引用时用"数据框名$变量名"的记法较麻烦, attach() 函数可以把数据框"连接"入当前的名字空间。

例如,

r <- d$height /d$age

r

*# [1] 8.500000 8.142857 7.954545 7.857143 9.050000*

*# attach(d) 可以把数据框引入当前的命名空间。*

attach(d)

*# The following objects are masked from d (pos = 3):*

*#*

*# age, gender, height, name*

*#*

*# The following objects are masked from d (pos = 4):*

*#*

*# age, gender, height, name*

r <- height /age

r

*# [1] 8.500000 8.142857 7.954545 7.857143 9.050000*

• 后一语句将在当前工作空间建立一个新变量r, 它不会自动进入数据框d, 要把新变量赋值到数据框中, 可以用

> d$r <- height / age这样的格式。

*# 给d添加1列, 值为height与age的比值*

d$r <- height / age

detach()

d

*# name age height gender r*

*# 1 赵 20 170 男 8.500000*

*# 2 钱 21 171 女 8.142857*

*# 3 孙 22 175 男 7.954545*

*# 4 李 21 165 女 7.857143*

*# 5 王 20 181 男 9.050000*

*# 删除r列, NULL表示没有, NA表示缺失值*

d$r <- **NULL**

d

*# name age height gender*

*# 1 赵 20 170 男*

*# 2 钱 21 171 女*

*# 3 孙 22 175 男*

*# 4 李 21 165 女*

*# 5 王 20 181 男*

• 为了取消连接, 只要调用detach() (无参数即可) 。

• 注意: R中名字空间的管理是比较独特的。它在运行时保持一个变量搜索路径表, 在读取某个变量时到这个变量搜索路径表中由前向后查找, 找到最前的一个；在赋值时总是在位置1赋值(除非特别指定在其它位置赋值) 。

• attach() 的缺省位置是在变量搜索路径表的位置2 ,

detach() 缺省也是去掉位置2。

• 所以, R编程的一个常见问题是当你误用了一个自己并没有赋值的变量时有可能不出错, 因为这个变量已在搜索路径中某个位置有定义, 这样不利于程序的调试, 需要留心。

• attach() 既可以连接数据框, 也可以连接列表。

#### 查看属性的方法

• mode()

• typeof()

• str()

• attr(, "") 查看或某个属性, 第二个参数不能为空

• class()

• attributes(obj) 查看对象属性

• attributes(<object>) ——查看对象object各种属性组成的列表

• attr(<object>, "name") ——存取对象object的名为name的属性

## 5. 外部数据导入

• txt

• csv

• excel

• sas

• spss

• 数据库

### 读入R包中自带的数据

*# 查看所有R自带的数据集*

data()

*# 直接输入某个数据名即可查看对应的数据集*

CO2

airquality

*# 查看某个某个包的数据并加装*

*# 方法一*

*# arules做关联规则时经常使用, 如可以分析哪些商品可以一起出现*

data(package="arules")

*# 读取arules包中的Income数据*

data(Income,package="arules")

*# 方法二, 先加载包, 再使用其中的数据*

library(arules)

data(Income)

detach()

### 读取外部数据

|  |  |  |  |
| --- | --- | --- | --- |
| 文件格式 | package | 读入函数 | 输出函数 |
| CSV | base | read.table read.csv | write.table write.csv |
| TXT | base | read.table | write.table |
| SAS | haven, foreign | read\_sas read\_xport | write\_sas |
| SPSS | haven, foreign | read\_spss read.spss | write\_spss |
| EXCEL | openxlsx, readxl | read.xlsx read\_excel | write.xlsx |
| 数据库 | sqldf, RODBC | sqldf() sqlQuery() | sqldf() sqlSave() |

base包为R自带的包, 不需要安装, 其它的都要安装.

R语言其实没有自已的保存数据的格式文件, 想要把数据保存取文件中, 必须要使用上面的方法导出为其它格式的文件.

### 读取TXT

#### read.table

函数: read.table(file, header=FALSE, sep="", stringAsFactors=) 函数以数据框的格式读入数据, 适合读取混合模式的数据, 但是要求每列的数据数据类型相同。

file参数: 这是必须的, 可以是相对路径或者绝对路径(注意: Windows下路径要用斜杠'/'或者双反斜杠'\\') 。

header参数: 默认为FALSE即数据框的列名为V1, V2..., , 设置为TRUE时第一行作为列名。

data1<-read.table('item.csv') #默认header=FALSE

data2<-read.table('item.csv', header=TRUE)

sep参数: 分隔符, 默认为空格。可以设置为逗号(comma) sep=', ', 分号(semicolon) sep=';'和制表符(tab) 。

stringAsFactors默认是将字符变量转成因子变量. 因子对应于统计中的分类变量.

read.csv、read.csv2、read.delim是read.table函数的包装, 分隔符分别对应逗号, 分号, 制表符, 同样接受read.table所有参数。

# 读取TXT文件

# 查看默认工作路径

getwd()

# 设置默认工作路径

setwd("D:\\R\_edu\\data")

# 设置工作路径之后再读取文件时就不用再指定工作路径的路径了

t1 <- read.table("CLASS.txt",header=**TRUE**)

# 跳过几列

# row.names设置行的名字

# na.strings=""如果在数值变量中发现空格则为缺失

t2 <- read.table("CLASS.txt",header=**TRUE**,colClasses =c("character", "character", "integer", "NULL", "NULL"))

t2

#### read.csv

read.csv函数header参数默认为TRUE, 不同于read.table

#字符型数据读入时自动转换为因子

data5<-read.csv('item.csv', stringAsFactors=FALSE)

#如果数据集中含有中文, 直接导入很有可能不识别中文, 这时加上参数fileEncoding='utf-8'

read.csv('data.csv', fileEncoding='utf-8')

### 读取EXCEL

library(readxl)

#读取Excel

read\_excel("old\_excel.xls")

read\_excel("new\_excel.xlsx")

#sheet参数, 指定sheet名或者数字

read\_excel("excel.xls", sheet=2)

read\_excel("excel.xls", sheet="data")

• # If NAs are represented by something other than blank cells, set the na argument

• read\_excel("excel.xls", na = "NA")

# EXCEL

library(openxlsx)

t1 <- read.xlsx("hsb2.xlsx",sheet="hsb2")

t1 <- read.xlsx("hsb2.xlsx",sheet=1)

t2<-read.xlsx("hsb2.xls",sheet=1)

class(t1)

library(readxl)

?read\_excel

t2<read\_excel("hsb2.xls","hsb2")

detach()

### SPSS

# SPSS

library(haven)

w <- read\_spss("D:/R\_edu/data/hsb2.sav")

w

# 读入的文件有tbl\_df, tbl, data.frame三种格式. 是在数据框的基础上延伸出来的处理速度比较高的table格式

class(w)

# 转换为数据框格式

w1 <- as.data.frame(w)

class(w1)

### SAS

# SAS

library(haven)

w <- read\_sas("D:/R\_edu/data/class.sas7bdat")

names(w)

library(Hmisc)

describe(w)

class(w)

w1 <- as.data.frame(w)

#### 从数据库中读取数据

# 数据库

library(RODBC)

# 连接数据库

chan <- odbcConnect("mysqltest", uid="root", pwd="123")

# 查看数据库中所有的表

sqlTables(chan)

sqlTypeInfo(chan)

# 提取某张表

t1 <- sqlFetch(chan,"sc")

t1

#查询

sqlQuery(chan, "select \* from sc ")

sqlQuery(chan, "select \* from sc where cno=1 and grade>80")

## 6. 程序控制

• 循环

• If-else

• Apply系列

### 循环

• for (n in x) {expr}

R中最基本的是for循环, 其中n为循环变量, x通常是一个序列。n在每次循环时从x中顺序取值, 代入到后面的expr语句中进行运算。

x <- c(1,2,4)

for (n in x){

print (n^2)

}

while (condition) {expr}

当不能确定循环次数时, 我们需要用while循环语句。在condition条件为真时, 执行大括号内的expr语句

x <- 1

while(x<3){

print(x)

x <- x+1

}

### 条件

#### if条件

• if (conditon) {expr1} else {expr2}

if语句用来进行条件控制, 以执行不同的语句。若condition条件为真, 则执行expr1, 否则执行expr2。ifesle() 函数也能以简洁的方式构成条件语句。

if (test\_expression) {

statement

}

x <- 5

if(x>0) {

print("Positive number")

}

if(test\_expression) {

statement1

}else{

statement2

}

if <- 5

if(x>0) {

print("Non-negative number")

}else{

print("Negavite number")

}

if(x > 0) print("Non-negative number") else print("Negative number")

if(test\_expression1) {

statement1

}else if(test\_expression2) {

statement2

}else if(test\_expression2) {

statement3

}else{

statement4

}

x <- 0

if (x < 0) {

print("Negative number")

} else if (x > 0) {

print("Positive number")

} else {

print("Zero")

}

#### switch

在R语言中创建switch语句的基本语法是

switch(expression, case1, case2, case3....)

expression表达式的结果为1, 就运行case1, 结果为2, 就运行case2, etc.

如果表达式的值不是字符串, 则被强制转化为整数。

在switch内可有任意数量的case语句。 每个case语句后跟要比较的值和冒号。

如果整数的值在1和nargs() - 1(最大参数数) 之间, 则对条件的相应元素进行求值并返回结果。

x <- 2

switch (x,

"1 st prize",

"2 st prize",

"3 st prize"

)

#### apply系列-循环的替代

在R中一般不提倡使用循环, 因为循环会比较耗费资源, 所以推荐以下面的方法替代

|  |  |  |
| --- | --- | --- |
| 作用目标 | 在每个元素上应用 | 在子集合上应用 |
| array/dataframe /matrix | apply | tapply |
| list | lapply | by |
| vector | lapply | tapply/by |

#####apply系列

# apply(array, margin, FUN, ...)

# array为矩阵或向量, margin为边缘, FUN为作用函数

m <- matrix(c(1:12), nrow=3)

m

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

# [1,] 1 4 7 10

# [2,] 2 5 8 11

# [3,] 3 6 9 12

# 1表示行, 把sum函数作用到m的每一行上, 对每一行求和

apply(m, 1, sum)

# [1] 22 26 30

# 2表示列, 对每一列求和

apply(m, 2, sum)

# [1] 6 15 24 33

# 3维的可以分别作用在3个维度上

a <- array(c(1:12), dim=c(2,3,2))

a

apply(a,1,sum)

apply(a,2,sum)

apply(a,3,sum)

# tapply(array, indices, margin, FUN=NULL, ...)

# 在子集合上进行运算. array为要操作的对象, indices为维度,

# 按indices中的值分组，把相同值对应下标的array中的元素形成一个集合，应用到FUN。

# 类似于group by indices的操作。如果FUN返回的是一个值，tapply返回vector；

# 若FUN返回多个值，tapply返回list。vector或list的长度和indices中不同值的个数相等。

setwd("D:\\R\_edu\\data")

cre <- read.csv("creditcard.csv")

str(cre)

# 按Creditability信用好坏对CreditAmount进行分组

tapply(cre$CreditAmount, cre$Creditability)

# 按Creditability信用好坏对CreditAmount进行分组, 并计算每组的均值

tapply(cre$CreditAmount, cre$Creditability,mean)

# 0 1

# 3938.127 2985.443

# 按Creditability信用好坏对CreditAmount进行分组, 并计算每组的5个特征值

tapply(cre$CreditAmount, cre$Creditability,fivenum)

# $`0`

# [1] 433.0 1350.0 2574.5 5154.0 18424.0

#

# $`1`

# [1] 250.0 1375.0 2244.0 3637.5 15857.0

tapply(cre$CreditAmount, cre[,c('Occupation', 'Sex.Marital.Status')], sum,na.rm=**TRUE**)

# by(dataframe, INDICES, FUN, ..., simplify=TRUE)

# by可以当成dataframe上的tapply。indices应当和dataframe每列的长度相同。

# 返回值是by类型的object。若simplify=FALSE，本质上是个list。

by(cre$CreditAmount, cre$Creditability, mean)

# lapply(list, FUN, ...)

# 在list上逐个元素调用FUN。可以用于dataframe上，

# 因为dataframe是一种特殊形式的list。例

lst <- list(a=c(1:5), b=c(6:10))

lapply(lst, mean)

# sapply(list, FUN, ..., simplify, USE.NAME=TRUE)

# sapply(\*, simplify = FALSE, USE.NAMES = FALSE) is equivalent to lapply(\*).

# 比lapply多了一个simplify参数。如果simplify=FALSE，则等价于lapply。

# 否则，在上一种情况的基础上，将lapply输出的list简化为vector或matrix

?sapply

lst <- list(a=c(1:5), b=c(6:10))

r1 <- sapply(lst, mean)

r1

class(r1)

mode(r1)

r2 <- sapply(lst, fivenum)

r2

class(r2)

mode(r2)

R 拓展学习

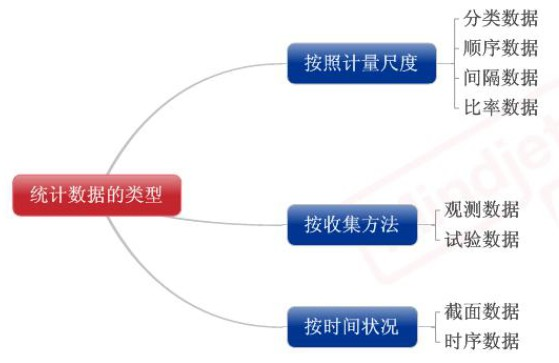
tibble类型

# R 进行描述性分析

## 1. 变量统计特征探索

### 1.1 集中趋势的度量

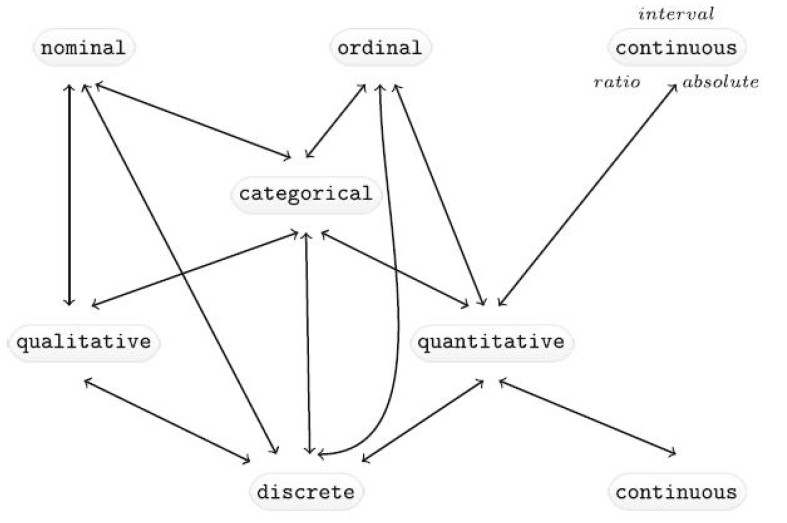
#### 数据的分类



间隔数据和比率数据合起来为数值型数据???

不同的数据类型使用不同的方式进行描述

#### 类型与取值之间的关系



|  |  |  |
| --- | --- | --- |
| nominal | 名义上的 |  |
| ordinal | 序数 | 顺序数据 |
| continuous | 连续 | 连续数据 |
| categorical | 分类 | 类别数据?? |
| qualitative | 定性 |  |
| quantitative | 定量 |  |
| discrete | 离散 |  |
| continuous | 连续 |  |

#### CREDITCRAD数据集介绍

德国信用卡数据

|  |  |  |
| --- | --- | --- |
| 序号 | 英文名称 | 中文说明 |
| [1] | Creditability | 信用水平(好/差) |
| [2] | Account Balance | 账户余额 |
| [3] | Duration | 注册时间 |
| [4] | Payment Status of Previous Credit | 上笔贷款支付状态 |
| [5] | Purpose | 贷款目的 |
| [6] | Credit Amount | 贷款金额 |
| [7] | Value Savings | 存款状态 |
| [8] | Length of current employment | 当前工作年数 |
| [9] | Instalment percent | 收入偿债比 |
| [10] | Sex.Marital.Status | 性别婚姻状态 |
| [11] | Guarantors | 担保人数量 |
| [12] | Duration in Current address | 当前地址年数 |
| [13] | Most valuable available asset | 贵重资产 |
| [14] | Age | 年龄 |
| [15] | Concurrent Credits | 当前信用评分 |
| [16] | Type of apartment | 住房类型 |
| [17] | No of Credit at this Bank | 授信银行数 |
| [18] | Occupation | 职业 |
| [19] | No of dependents | 家庭人数 |
| [20] | Telephone | 电话 |
| [21] | Foreign Worker | 是否为外国 |

#### 软件实现

• 连续转离散

不同的数据类别的描述方法及参数都不一样.

cut(x, breaks, labels = NULL, include.lowest =FALSE, right = TRUE, dig.lab = 3, ordered\_result =FALSE, ...)

x为待转换连续性变量

breaks为数字或者数值变量, 数字表示等分数, 如5表示对x进行5等分, 数值变量表示边界分割. c(1, 3, 6, 7) 表示4个分割点, 把数据分为3个区间, 1-3之间的数, 3-6之间的数, 6-7之间的数.

labels可以为每个区间进行命名, 默认没有名称.

include.lowest第一个区间包含左端点／最后一个区间包含右端点. 对于左开右闭区间, (1, 3], (3, 6], (6, 7], include.lowest表示第1个区间包含左端点1, 即第1个区间为[1, 3]

right=TRUE表示左开右闭, 反之左闭右开

ordered\_result返回有序因子

|  |  |  |
| --- | --- | --- |
| 数据类型 | 辨别函数 | 转换函数 |
| character | is.character() | as.character() |
| complex | Is.complex() | as.complex() |
| double | is.double() | as.double() |
| integer | is.integer() | as.integer() |
| logical | is.logical() | as.logical() |
| NA | is.na() | as.na() |
| numeric | is.numeric() | as.numeric() |

# 设置默认的工作路径

setwd("D:\\R\_edu\\data")

# 读取csv文件

cre <- read.csv("creditcard.csv")

# 查看cre总体的数据类型. "data.frame"

class(cre)

# 查看每个字段的基本数据类型和结构

str(cre)

# names查看数据框data.frame中每个字段的名称, 即所有的变量

names(cre)

# 数值变量转变为类别变量, 把年龄分为几组.

# 方法1

# 先创建一个agegrade的空字段, 再根据Age字段的大小依次对agegrade字段进赋值.

cre$agegrade<- **NA**

# 注意NA与NULL的区别, NA是空值, NULL表示删除该字段

# cre$agegrade <- NULL

cre$agegrade[cre$Age<=45] <- "young"

cre$agegrade[cre$Age>45 & cre$Age<=60] <-"middle"

cre$agegrade[cre$Age>60] <- "old"

# 如果不想每次执行时都使用写cre, 可以使用attach把数据加载到系统变量中, 执行完成后再使用detach即可.

# 方法2

# within函数更间接

within(cre,{

agegrade<- **NA**

agegrade[Age<=45] <- "young"

agegrade[Age>45 & Age<=60] <-"middle"

agegrade[Age>60] <- "old"

})

# 方法3

# 把年龄分为几组. 在cre中创建了一个新的字段Agel, 并把分组的结果写入到这个字段中

cre$Age1 <- cut(cre$Age,

breaks=c(min(cre$Age),45,60,max(cre$Age)),

labels=c("young","middle","old"),

right=**TRUE**,)

# 对上面不同方法得到的数据进行对比, 看结果是不是完全相同.

all.equal(cre$Age1,as.factor(cre$agegrade))

t2 <- cre[cre$Age1==as.factor(cre$agegrade),]

class(cre$Age1)

class(cre$agegrade)

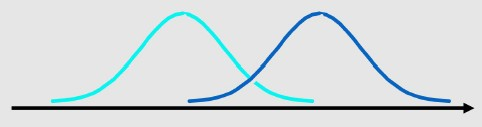
?all.equal

str(cre)

#### 数据分布的特征(图形)

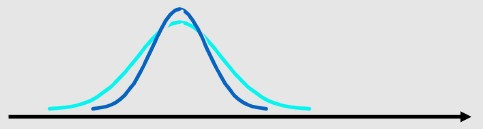
集中趋势

(位置)



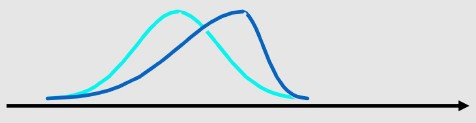
离中趋势

(分散程度)

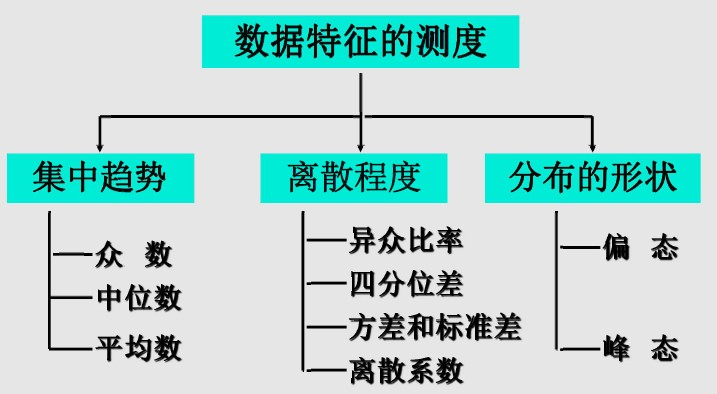


偏态和峰态

(形状)



#### 数据分布特征的测度(统计量)



#### 数据类型与集中趋势测度值

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据类型和所适用的集中趋势测度值 | | | | |
| 数据类型 | 分类数据 | 顺序数据 | 间隔数据 | 比率数据 |
| 适  用  的  测  度  值 | ※众数 | ※中位数 | ※平均数 | ※平均数 |
| — | 四分位数 | 众数 | 调和平均数 |
| — | 众数 | 中位数 | 几何平均数 |
| — | — | 四分位数 | 中位数 |
| — | — | — | 四分位数 |
| — | — | — | 众数 |

#### 软件实现

• table()

• prob.table(table对象)

• 均值

求均值时数据的分布不能太极端了, 否则均值的代表性就会很差. 可以把异常值移除掉, 得到的均值称为截尾平均数. 设置trim = 0.1即表示移除左边和右边10%的数据, 只使用中间80%的数据求平均值.

na.rm是否移除NA值, 默认是跳过缺失值NA, 默认为FALSE, 一定要改为na.rm=TRUE.

mean(x, trim = 0, na.rm = FALSE, ...)

• 中位数

median(x, na.rm = FALSE)

• 众数

which.max求出现次数最多的数的位置??

names(table(x) ) [which.max(table(x) ) ]

• 百分位数

probs = seq(0, 1, 0.25) , 从0到1, 每隔0.25求一个分位数, 即求4分位数.

quantile(x, probs = seq(0, 1, 0.25) , na.rm = FALSE, names = TRUE, type = 7, ...)

# 分别求每个字段的均值

mean(cre$Age)

# [1] 35.542

mean(cre$CreditAmount)

# [1] 3271.248

# 使用apply来处理多个字段的均值, 向量c(14,6)表示求第6列和14列的均值.

apply(cre[,c(14,6)],2,mean)

# Age CreditAmount

# 35.542 3271.248

#中位数

median(cre$CreditAmount)

# [1] 2319.5

# 对类别型数据使用列表分析, 以列表的形式查看贷款的目的

table(cre$Purpose)

# 0 1 2 3 4 5 6 8 9 10

# 234 103 181 280 12 22 50 9 97 12

# 使用概率表查看占比, 查看贷款目的的占比, properbility即概率. 注意prop.table的参数是table

prop.table(table(cre$Purpose))

# 0 1 2 3 4 5 6 8 9 10

# 0.234 0.103 0.181 0.280 0.012 0.022 0.050 0.009 0.097 0.012

# 基于单变量的列表分析, 对类别型数据

table(cre$Age1)

# young middle old

# 814 139 45

# 基于多变量的列表分析.

table(cre$Age1,cre$Creditability)

# 0 1

# young 252 562

# middle 37 102

# old 10 35

table(cre$Purpose,cre$Creditability)

# 0 1

# 0 89 145

# 1 17 86

# 2 58 123

# 3 62 218

# 4 4 8

# 5 8 14

# 6 22 28

# 8 1 8

# 9 34 63

# 10 5 7

# 二元的概率表. margin设置行百分比和列百分比, margin=2时, 每一列的百分比之和为1, margin=1时, 每一行的百分比之和为1

prop.table(table(cre$Purpose,cre$Creditability),margin=2)

# 0 1

# 0 0.296666667 0.207142857

# 1 0.056666667 0.122857143

# 2 0.193333333 0.175714286

# 3 0.206666667 0.311428571

# 4 0.013333333 0.011428571

# 5 0.026666667 0.020000000

# 6 0.073333333 0.040000000

# 8 0.003333333 0.011428571

# 9 0.113333333 0.090000000

# 10 0.016666667 0.010000000

prop.table(table(cre$Purpose,cre$Creditability),margin=1)

# 0 1

# 0 0.3803419 0.6196581

# 1 0.1650485 0.8349515

# 2 0.3204420 0.6795580

# 3 0.2214286 0.7785714

# 4 0.3333333 0.6666667

# 5 0.3636364 0.6363636

# 6 0.4400000 0.5600000

# 8 0.1111111 0.8888889

# 9 0.3505155 0.6494845

# 10 0.4166667 0.5833333

### 1.2 离散程度的度量

#### 离散程度的度量

分类数据: 异众比率

顺序数据: 四分位差

数值型数据: 方差和标准差

相对位置的度量: 标准分数

相对离散程度: 离散系数

#### 相对位置的度量: 标准分数(standard score)

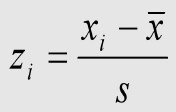
1. 也称标准化值

2. 对某一个值在一组数据中相对位置的度量

3. 可用于判断一组数据是否有离群点

4. 用于对变量的标准化处理

5. 计算公式为



#### 经验法则

经验法则表明: 当一组数据对称分布时

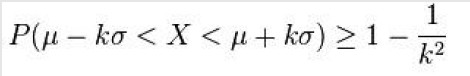
l 约有68%的数据在平均数加减1个标准差的范围之内

l 约有95%的数据在平均数加减2个标准差的范围之内

l 约有99%的数据在平均数加减3个标准差的范围之内

#### 切比雪夫不等式(Chebyshev's inequality )

1. 如果一组数据不是对称分布, 经验法则就不再使用, 这时可使用切比雪夫不等式, 它对任何分布形状的数据都适用



2. 切比雪夫不等式提供的是"下界", 也就是"所占比例至少为多少"

3. 对于任意分布形态的数据, 根据切比雪夫不等式, 至少有1-1/k2的数据落在k个标准差之内。其中k是大于1的任意值, 但不一定是整数

#### 切比雪夫不等式(Chebyshev's inequality )

对于k=2, 3, 4, 该不等式的含义是

l 至少有75%的数据落在平均数加减2个标准差的范围之内

l 至少有89%的数据落在平均数加减3个标准差的范围之内

l 至少有94%的数据落在平均数加减4个标准差的范围之内

#### 相对离散程度: 离散系数

(coefficient of variation)

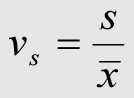
l 标准差与其相应的均值之比

l 对数据相对离散程度的测度

l 消除了数据水平高低和计量单位的影响

l 用于对不同组别数据离散程度的比较

l 计算公式为



#### 数据类型与离散程度测度值

|  |  |  |  |
| --- | --- | --- | --- |
| 数据类型和所适用的集中趋势测度值 | | | |
| 数据类型 | 分类数据 | 顺序数据 | 数值型数据 |
| 适  用  的  测  度  值 | ※异众比率 | ※四分位差 | ※方差或标准差 |
| — | 异众比率 | ※离散系数(比较时用) |
| — | — | 平均差 |
| — | — | 极差 |
| — | — | 四分位差 |
| — | — | 异众比率 |

#### 软件实现

# 方差

var(cre$CreditAmount)

# [1] 7967927

# 标准差

sd(cre$CreditAmount)

# [1] 2822.752

# 极差

range(cre$CreditAmount)

# [1] 250 18424

# 标准分数, 每个数减去均值除以标准差

scale(cre$CreditAmount)

# IQR四分位数 IQR(x)

IQR(cre$CreditAmount)

# [1] 2606.75

# 离散系数 sd(x) /mean(x)

sd(cre$CreditAmount) /mean(cre$CreditAmount)

# [1] 0.8628975

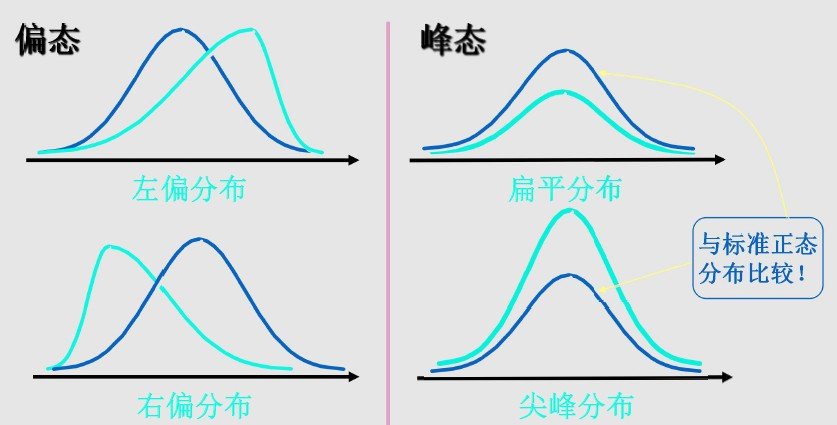
# 汇总输出

summary(cre$CreditAmount)

# Min. 1st Qu. Median Mean 3rd Qu. Max.

# 250 1366 2320 3271 3972 18424

### 1.3 偏态和峰态的度量



#### 偏态(skewness)

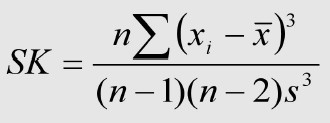
1. 统计学家Pearson于1895年首次提出

2. 数据分布偏斜程度的测度

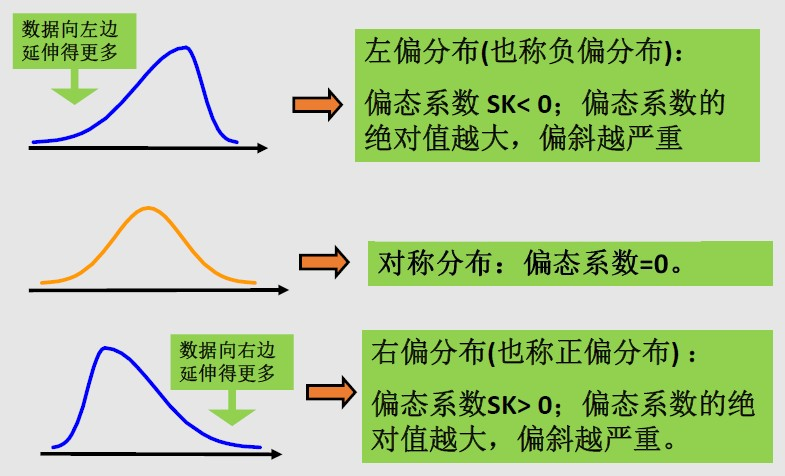
l 偏态系数=0为对称分布

l 偏态系数> 0为右偏分布

l 偏态系数< 0为左偏分布



#### 偏态系数的含义



#### 峰态(kurtosis)

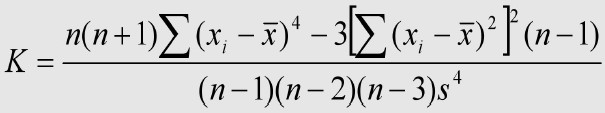
1. 统计学家Pearson于1905年首次提出

2. 数据分布扁平程度的测度

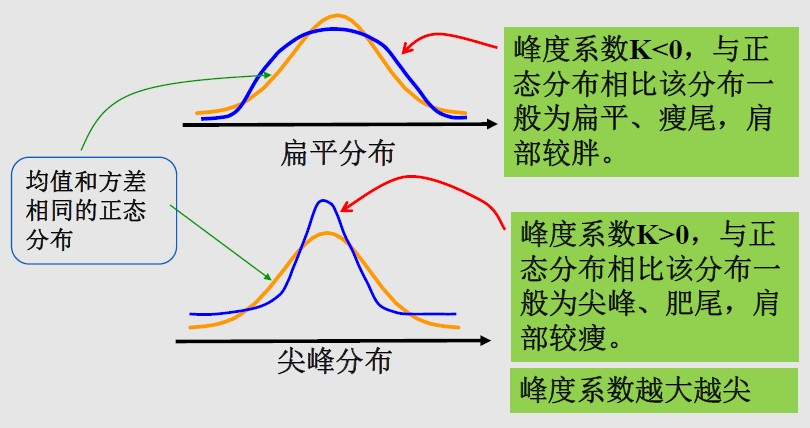
l 峰态系数=0扁平峰度适中

l 峰态系数<0为扁平分布

l 峰态系数>0为尖峰分布



#### 峰度系数的含义



#### 软件实现

library(Hmisc)

t1 <-Hmisc::describe(cre)

t1

#汇总统计量+偏度+峰度

library(psych)

t1 <- describe(cre$CreditAmount)

t1

# vars n mean sd median trimmed mad min max range skew kurtosis se

# X1 1 1000 3271.25 2822.75 2319.5 2754.59 1627.15 250 18424 18174 1.94 4.25 89.26

library(e1071)

# 求峰态

kurtosis(cre$CreditAmount)

# [1] 4.250531

# 求偏度

skewness(cre$CreditAmount)

# [1] 1.943749

# 自定义函数计算统计量

myStat <- **function**(x){

n <- length(x) #样本数据个数

m <- mean(x) #均值

me <- median(x) #中位数

mo <- names(table(x))[which.max(table(x))] #众数

sd <- sd(x) #标准差

v <- var(x) #方差

r <- max(x) - min(x) #极差

cv <- 100 \* sd/m #变异系数

R1 <- quantile(x,0.75) - quantile(x,0.25) #四分位差

sm <- sd/sqrt(n) #标准误

g1 <- n/((n-1)\*(n-2)\*sd^3)\*sum((x-m)^3)/sd^3 #偏度系数

g2 <- ((n\*(n+1))/((n-1)\*(n-2)\*(n-3))\*sum((x-m)^4)/sd^4 -(3\*(n-1)^2)/((n-2)\*(n-3))) #峰度系数

# 使用data.frame就可以把数据返回

data.frame(N=n,Mean=m,Median=me,Mode=mo,

Std\_dev=sd,Variance=v,Range=r,

CV=cv,R1=R1,SM=sm,Skewness=g1,

Kurtosis=g2,

row.names=1)

}

# 调用函数完成计算

myStat(cre$CreditAmount)

# N Mean Median Mode Std\_dev Variance Range CV R1 SM Skewness

# 1 1000 3271.248 2319.5 1258 2822.752 7967927 18174 86.28975 2606.75 89.26325 8.668145e-11

# Kurtosis

# 1 4.292481

## 2. 变量的可视化探索

### 图形展示三步

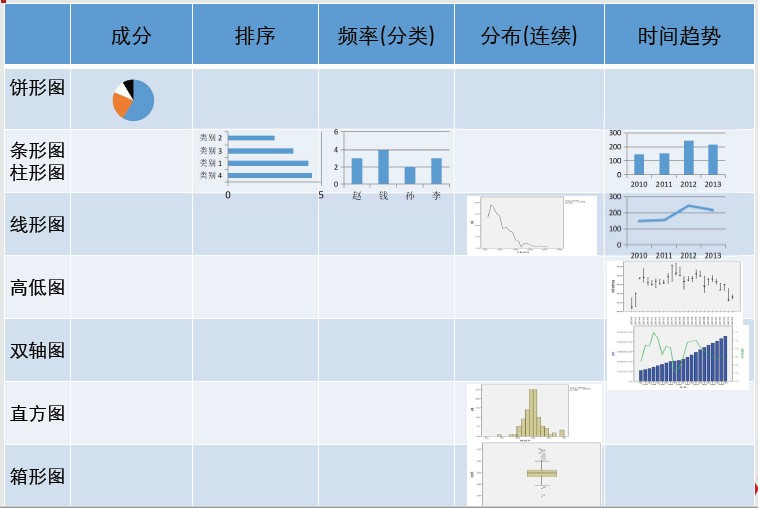
第一步 明确你要表达的信息

第二步 确定相对关系

第三步 选择图表形式

数据 > 信息 > 相对关系 > 图形

通过不同的图表表达特定对比



饼图: 不同成份之间的对比

排序: 条形图或柱形图

频率, 分类: 条形图, 线形图

连续分布: 线形图, 直方图, 箱形图

时间趋势: 条形图, 柱形图, 线形图, 高低图, 双轴图

### 软件实现

对比Excel中每个图形的绘制和图形的设置

#### 饼图(pie)

pie(x, labels = names(x) , radius = 0.8, col = NULL)

x 数组如C(1, 3) , 数组中的每个值决定了饼图扇区的大小, 比例为c[i]/sum(c)

labels 饼图扇区的描述, R中要显示扇区的比例, 或者数字

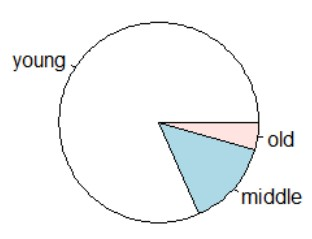
radius 饼图半径

col 颜色填充

# 饼图

t1 <- table(cre$Age1)

pie(t1)



#### 柱状图(barplot)

barplot(height, beside = FALSE, horiz = FALSE, col = NULL, xlim =NULL, log = "", axes = TRUE, offset = 0 )

beside 默认值为 FALSE, 表示累计柱状图

horiz表示横向还是纵向

col设置颜色

xlimX轴的取值范围

log表示是否对坐标轴取对数

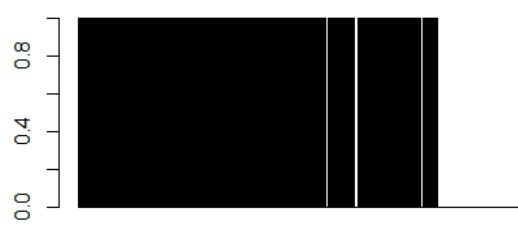
axes表示是否含有坐标轴

offset表示Y轴的开始位置

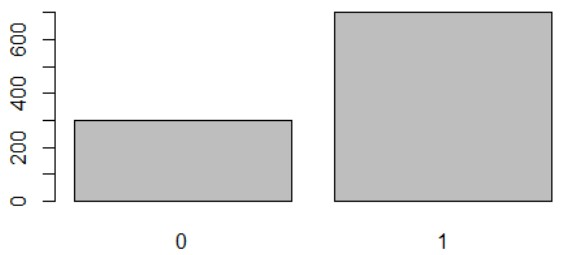
# 柱状图

?barplot

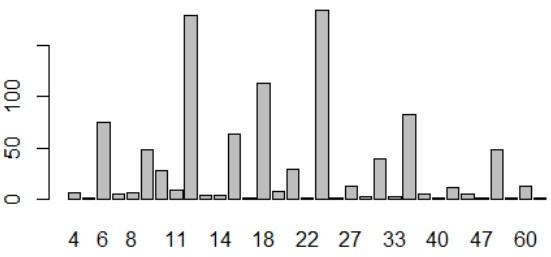
barplot(cre$Creditability)



barplot(table(cre$Creditability))



barplot(table(cre$Duration))



#### 直方图(hist)

hist(x, breaks = "Sturges", freq = NULL, density = NULL, col = NULL, xlab= xname, ylab, axes = TRUE, labels = FALSE, nclass = NULL)

break指定一个向量, 给出不同的断点; 或者给出区间数. 如果要分成5组, breaks等于5即可.

freq: 默认值为TRUE , y轴显示的是每个区间内的频数, FALSE, 代表显示的是频率

labels: 显示在每个柱子上方的标签

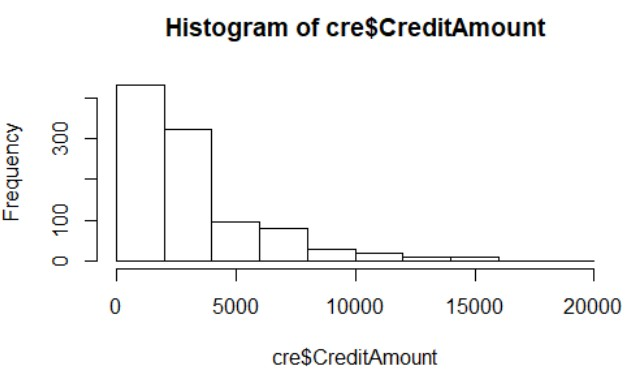
axes: 逻辑值, 是否显示轴线

col: 柱子的填充色

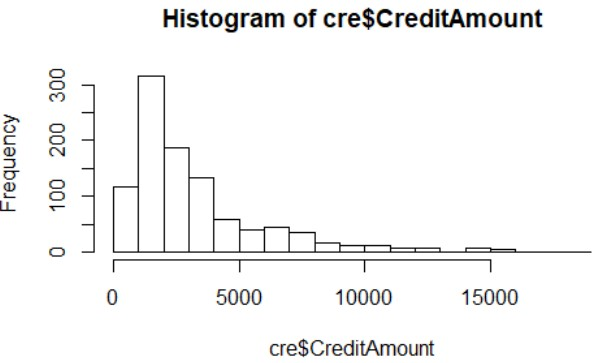
density: 用线条填充柱子, 数字代表线的密集

# 直方图

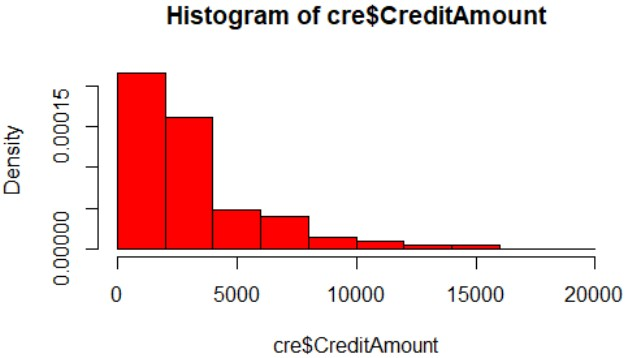
h <- hist(cre$CreditAmount)



h <- hist(cre$CreditAmount,breaks=20)



hist(cre$CreditAmount,freq=FALSE,col="red")



#### 线图(plot)

plot(x, y, col=, type=, pch)

x y代表x轴, y轴

col代表颜色

type代表连接类型 (l代表line, p代表points, b代表both, s代表阶梯型)

pch代表点的类型

此函数为泛型函数

# 线型图

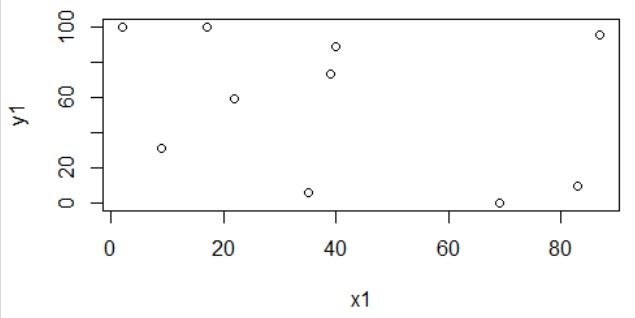
# round为取整, runif模型产生随机数. 产生10个[0-100]之间的随机数

x1 <- round(runif(10, min = 0, max = 100))

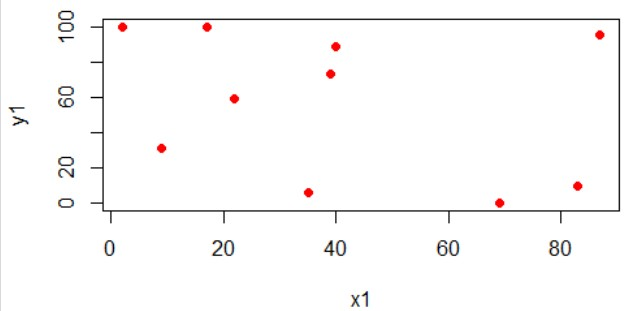
y1 <- round(runif(10, min = 0, max = 100))

z1 <- seq(1:10)

plot(x1,y1)

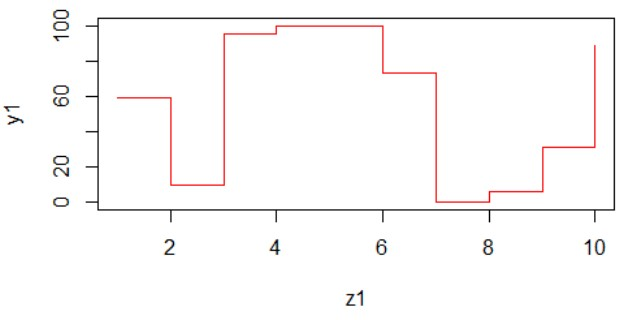


plot(x1,y1,col="red",pch=19)



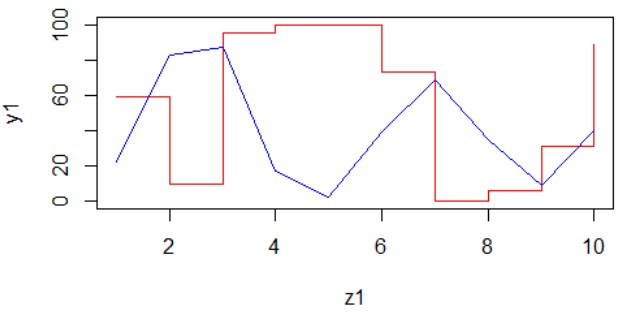
# 也就是说plot是泛型函数, 它有很多子函数, 是调用子函数来完成绘图功能的.

plot(z1,y1,type="s",col="red")

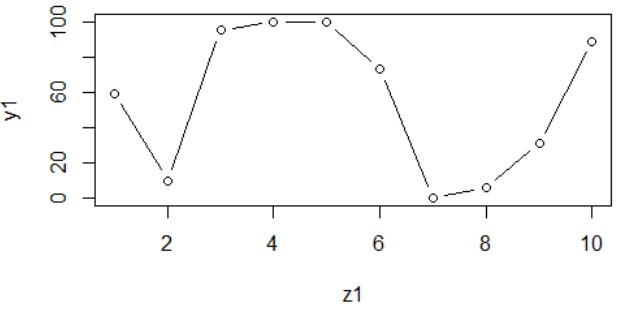


# 同时绘制多条线, 添加第2条线时就使用lines来完成了.

lines(z1,x1,type="l",col="blue")



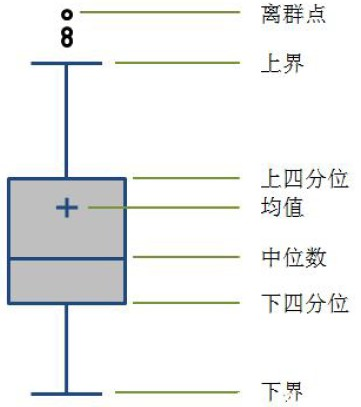
plot(z1,y1,type="b")



?plot

#### 箱线图

"盒式图"或叫"盒须图" , 其绘制须使用常用的统计量, 最适宜提供有关数据的位置和分散的参考, 尤其在不同的总体数据时更可表现其差异。



上界为Q3+1.5IQR, 下界=Q1-1.5IQR, 上界与最大值max对比, 取二者中最小值为上界, 下界与min对比, 取二者中的最大值为下界, 在上界和下界这个区间之外的数就是离群点

boxplot(x, ...)

## S3 method for class 'formula'

boxplot(formula, data = NULL, ..., subset, na.action = NULL,

drop = FALSE, sep = ".", lex.order = FALSE)

## Default S3 method:

boxplot(x, ..., range = 1.5, width = NULL, varwidth = FALSE,

notch = FALSE, outline = TRUE, names, plot = TRUE,

border = par("fg"), col = NULL, log = "",

pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),

horizontal = FALSE, add = FALSE, at = NULL)

其中的range就是求上下界时的1.5, 可以设置为其它值.

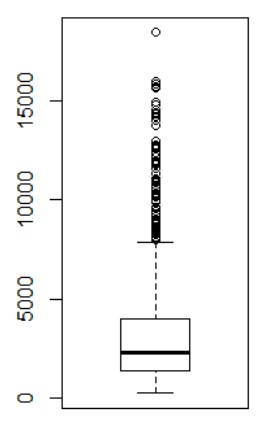
使用boxplot()绘制箱线图, 同时会产生一些统计量, 把离群点等显示出来.

• boxplot.stats(x, coef=1.5, do.conf=TRUE, do.out=TRUE)

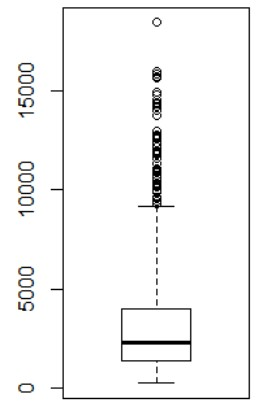
不用绘制箱型图, 使用boxplot.stats直接计算出离群值.

# 箱线图

b <- boxplot(cre$CreditAmount)

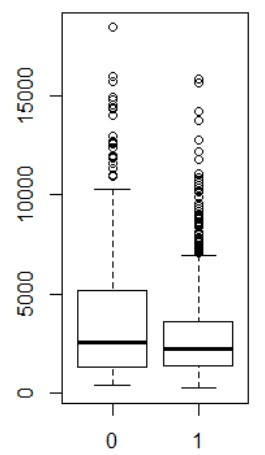


b <- boxplot(cre$CreditAmount,range=2)



b <- boxplot(cre$CreditAmount~cre$Creditability)

b



b1 <- boxplot.stats(cre$CreditAmount,coef=3)

# 把coef设置为3, 离群值会减少

b1

?boxplot

### ggplot2包介绍

• Hadley Wickham编写

• 基于wilkinson图形语法规则

• 允许创建创新性和新颖的图形

可以绘制多图层图形, 每一个图层实现不同的功能.

#### qplot

qplot(x, y = NULL, data, facets = NULL, geom = "auto", xlim, ylim, log = "", stat = NULL, position = NULL)

quick plot

facets为分组分面绘制. 如想要绘制年龄与刷卡金额之间的关系, 但要对不同的年龄分开绘制图形, facets就可以指定为性别.

geom取值为point(散点图) , smooth为平滑曲线, boxplot为盒形图, line(线性图) , histogram为直方图, density为密度曲线, bar为绘制条线图

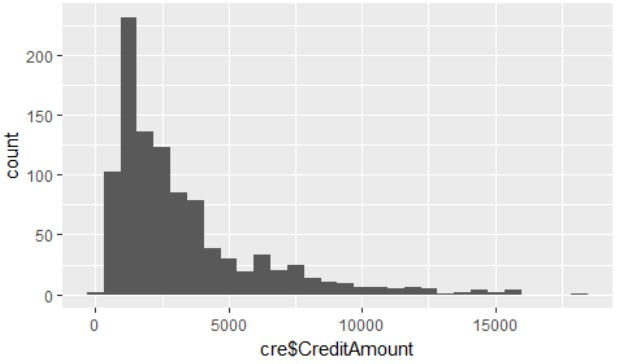
xlim, ylim设置x轴和y轴的取值范围.

log为x轴或者y轴是否取对数

library(ggplot2)

?mpg

qplot(cre$CreditAmount,geom="histogram")



#### ggplot

ggplot(data = mpg) +

geom\_point(mapping = aes(x = displ, y = hwy) ) +

facet\_grid(drv ~ cyl)

<https://studfiles.net/preview/6267809/page:135/>

可以绘制的图形

Table 19.2 Geom functions

|  |  |  |
| --- | --- | --- |
| Function | Adds | Options |
| geom\_bar() | Bar chart | color, fill, alpha |
| geom\_boxplot() | Box plot | color, fill, alpha, notch, width |
| geom\_density() | Density plot | color, fill, alpha, linetype |
| geom\_histogram() | Histogram | color, fill, alpha, linetype, binwidth |
| geom\_hline() | Horizontal lines | color, alpha, linetype, size |
| geom\_jitter() | Jittered points | color, size, alpha, shape |
| geom\_line() | Line graph | colorvalpha, linetype, size |
| geom\_point() | Scatterplot | color, alpha, shape, size |
| geom\_rug() | Rug plot | color, side |
| geom\_smooth() | Fitted line | method, formula, color, fill, linetype, size |
| geom\_text() | Text annotations | Many; see the help for this function |
| geom\_violin() | Violin plot | color, fill, alpha, linetype |
| geom\_vline() | Vertical lines | color, alpha, linetype, size |

Table 19.3 Common options for geom functions

|  |  |
| --- | --- |
| Option | Specifies |
| color | Color of points, lines, and borders around filled regions. |
| fill | Color of filled areas such as bars and density regions. |
| alpha | Transparency of colors, ranging from 0 (fully transparent) to 1 (opaque) . |
| linetype | Pattern for lines (1 = solid, 2 = dashed, 3 = dotted, 4 = dotdash, 5 = longdash, 6 = twodash) . |
| size | Point size and line width. |
| shape | Point shapes (same as pch, with 0 = open square, 1 = open circle, 2 = open triangle, and so on) . See figure 3.4 for examples. |
| position | Position of plotted objects such as bars and points. For bars, "dodge" places grouped bar charts side by side, "stacked" vertically stacks grouped bar charts, and"fill" vertically stacks grouped bar charts and standardizes their heights to be equal. For points, "jitter" reduces point overlap. |
| binwidth | Bin width for histograms. |
| notch | Indicates whether box plots should be notched (TRUE/FALSE) . |
| sides | Placement of rug plots on the graph ("b" = bottom, "l" = left, "t" = top, "r" = right, "bl" = both bottom and left, and so on) . |
| width | Width of box plots. |

library(ggplot2)

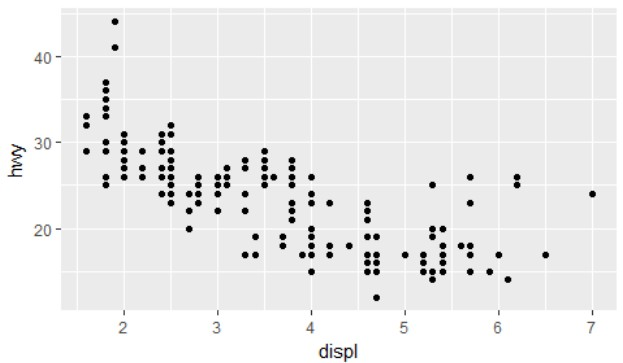
# 创建一个空模板

ggplot(data = mpg)

#加入点

ggplot(data = mpg) +

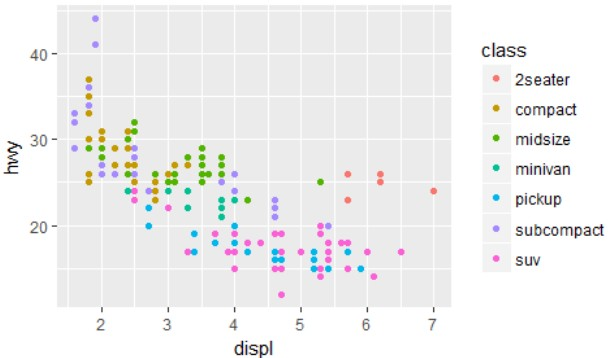
geom\_point(mapping = aes(x = displ, y = hwy))



#加入颜色

ggplot(data = mpg) +

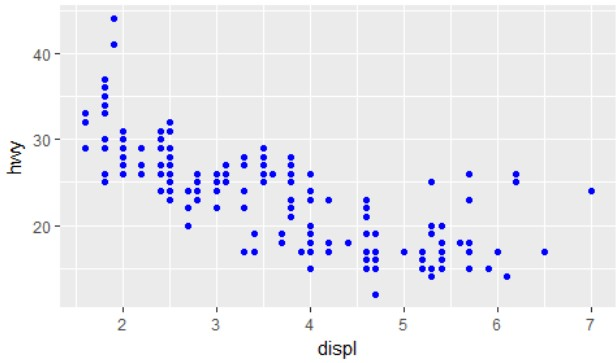
geom\_point(mapping = aes(x = displ, y = hwy, color = class))



#指定颜色

ggplot(data = mpg) +

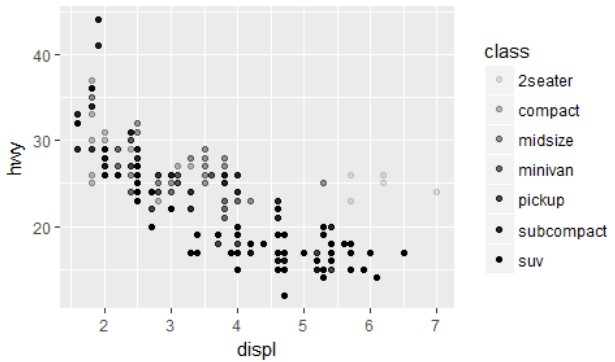
geom\_point(mapping = aes(x = displ, y = hwy), color = "blue")



# 控制透明度

ggplot(data = mpg) +

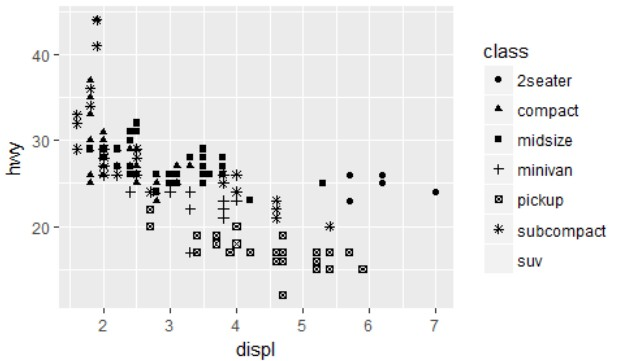
geom\_point(mapping = aes(x = displ, y = hwy, alpha = class))



# 控制形状

ggplot(data = mpg) +

geom\_point(mapping = aes(x = displ, y = hwy, shape = class))



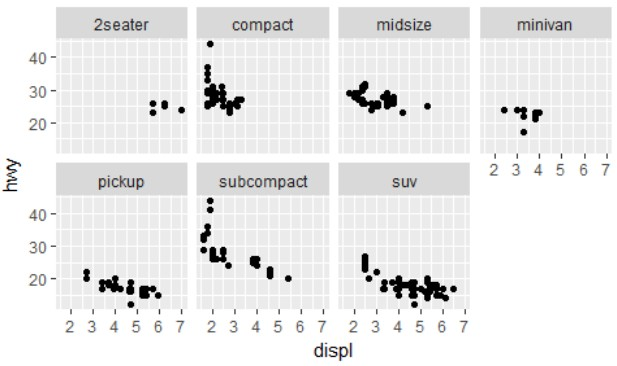
#分面

#一个维度

ggplot(data = mpg) +

geom\_point(mapping = aes(x = displ, y = hwy)) +

facet\_wrap(~ class, nrow = 2)

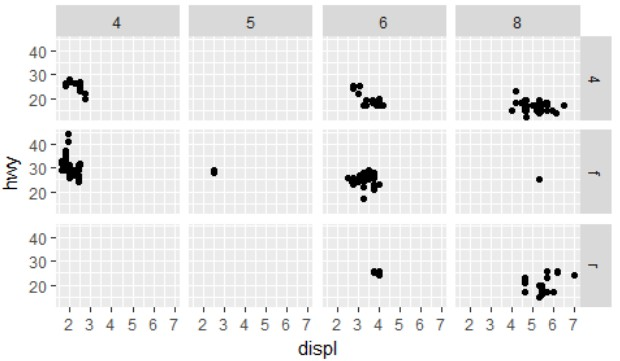


#两个维度

ggplot(data = mpg) +

geom\_point(mapping = aes(x = displ, y = hwy)) +

facet\_grid(drv ~ cyl)

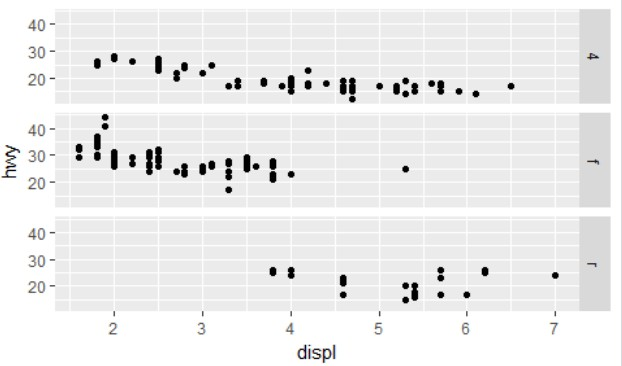


#每个drv水平的独立图，配置成一个单列

ggplot(data = mpg) +

geom\_point(mapping = aes(x = displ, y = hwy)) +

facet\_grid(drv ~ .)



#每个cyl水平的独立图，配置成一个单行

ggplot(data = mpg) +

geom\_point(mapping = aes(x = displ, y = hwy)) +

facet\_grid(. ~ cyl)



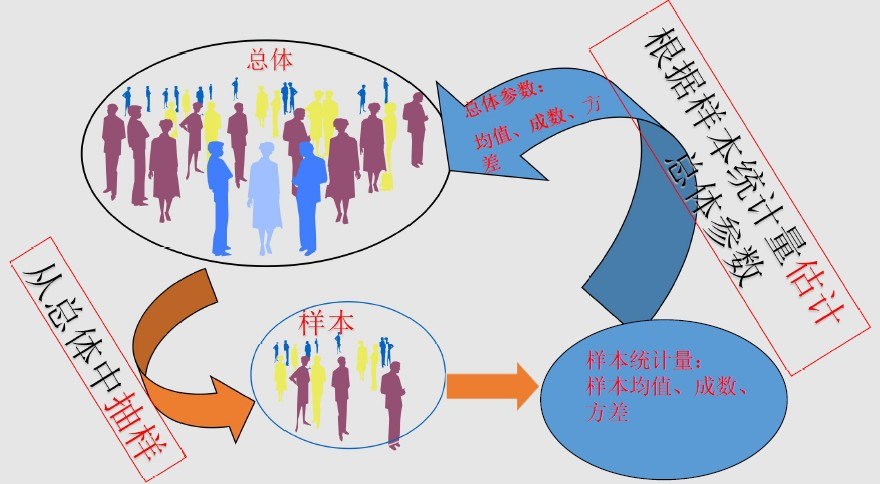
# R 进行推断性分析

## 1. 参数估计

### 总体参数与样本统计量的对比

|  |  |  |  |
| --- | --- | --- | --- |
| 总体参数 |  | 样本统计量 |  |
| 反映总体数量特征的指标 | 符号 | 反映样本数量特征的指标 | 符号 |
| 总体容量 | N | 样本容量 | n |
| 总体均值 | μ | 样本均值 |  |
| 总体成数(比例) | π | 样本成数(比例) | p |
| 总体方差 | σ2 | 样本方差 | s2 |
| 总体标准差 | σ | 样本标准差 | s |

### 抽样估计的过程



总体 > 从总体中抽样 > 样本 > 样本统计量, 样本均值, 成数, 方差 > 根据样本统计量估计总体参数 > 总体参数: 均值, 成数, 方差

### 参数估计的分类

估计方法:

点估计

点估计有两种方式, 第一种方法为矩估计, 用样本的方差来估计总体的方差, 用样本的均值估计总体的均值等, 这种方法是矩估计. 第二种方法是极大似然估计. 抽取一个样本来估计总体的参数, 得到的总体的参数值会使当前使用的估计总体参数的样本出现的可能性最大. 存在即合理, 既然现在得到了这个样本, 要用这个样本来对总体的参数进行估计, 那么得到的结果使当前样本出现的概率最大也是合理的.

区间估计

#### 点估计

点对点, 用样本单一的统计量来估计总体的统计量, 如用样本的均值来估计总体的均值.

优点: 简便、易行、原理直观, 它能够提供总体指标的具体估计值, 可以作为行动决策的数量依据。

缺点: 任何点估计不是对就是错, 点估计没有表明抽样估计的误差, 更没有指出误差在一定范围内的概率保证程度有多大。

**点估计软件实现**

# 查看所有自带的数据集

data()

# 查看cars数据集的所有变量及其基本数据结构, 其中有50个样本.

str(cars)

# dist为不同速度下的刹车距离. 随着速度的越来越大, 刹车距离越来越远

# 用样本的统计量估计总体的, 就是点估计

mean(cars$dist)

median(cars$dist)

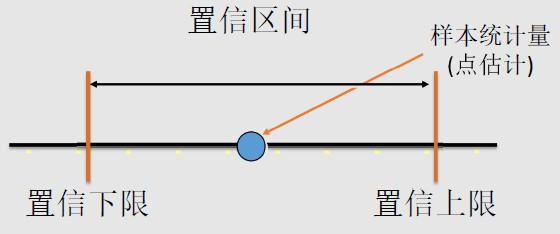
var(cars$dist)

#### 区间估计(interval estimate)

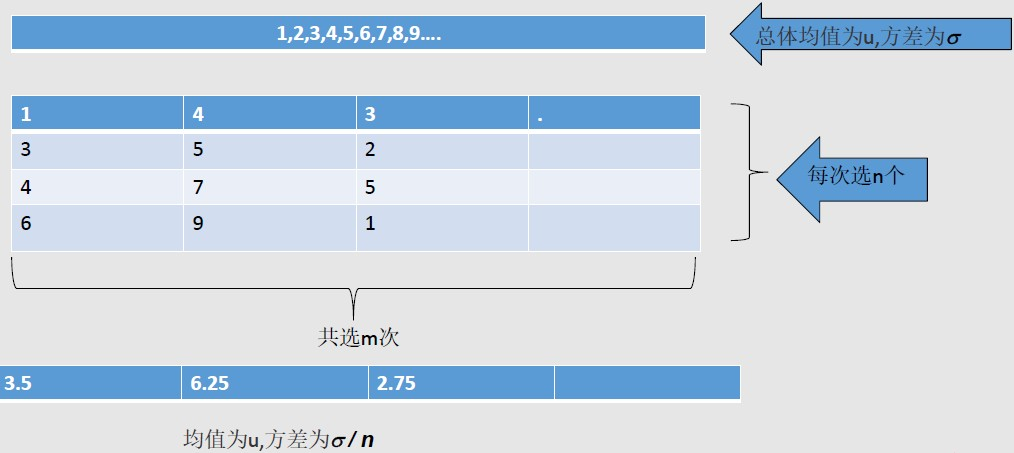
1. 在点估计的基础上, 给出总体参数估计的一个区间范围, 该区间由样本统计量加减抽样误差而得到的

2. 根据样本统计量的抽样分布能够对样本统计量与总体参数的接近程度给出一个概率度量

• 比如, 某班级平均分数在75～85之间, 置信水平是95%, 即可靠性程度, 有95%的把握.



#### 例子



样本均值的分布是指m次抽样的均值分布.从均值为μ, 方差为σ的总体中抽取出n个样本, 计算样本均值, 进行m次抽样, 得到m个样本的均值. 当抽样的样本量接近总体样本量时~~次数非常大时??~~, 对m次抽样的均值再取平均值, 得到的结果与总体的均值是一致的. 但是样本均值分布的方差变成了σ2/n

得到了m个样本均值后, 如何构建95% 的置信区间?

把m个样本均值从小到大排列, 再取下2.5%和上97.5%的分位数, 两个分位数之间的数据占所有样本的95%, 这里的95%就是置信区间, 有95%的样本落在置信区间中. 对于特定的分布, 给出置信水平/区间1-α, 就可以计算此置信水平/区间对应的分位点Zα/2, 每一个分位点都对应一个置信水平/区间. 同样, 给定了分位点, 也可以计算出对应的置信水平/区间. 还可以由样本参数估计总体参数, 例如, 对于正态分布, 如果已知样本量n, 样本均值x-bar和总体(样本)的标准差σ, 就可以由下面的公式计算出总体的均值所在的区间. 这就是总体均值的区间估计.

使用Excel计算分位数.

**软件实现**

**样本均值, 样本方差与总体均值, 总体方差的关系.**

# 模拟样本均值的分布情况

?rnorm

# 产生1000个均值为0, 方差为1的标准正态分布

x <- rnorm(1000,0,1)

# 定义一个函数, 从x中抽取n个样本计算平均值, 一共抽取m次.

sim.fun <- function (m,n){

sam <- 1:m

for (i in 1:m) {

sam[i] <- mean(sample(x,n))

}

sam

}

# 调用函数, 抽取100次, 每次抽取16个样本.

t1 <- sim.fun(m=100,n=16)

# 随着样本量的增多, 样本均值无限接近总体均值, 改变m,n值查看不同结果

mean(t1)

# 样本标准差等于总体标准差除以根号n, 1/sqrt(16)=0.25, 计算出来的值接近0.25, 改变m,n值查看不同结果

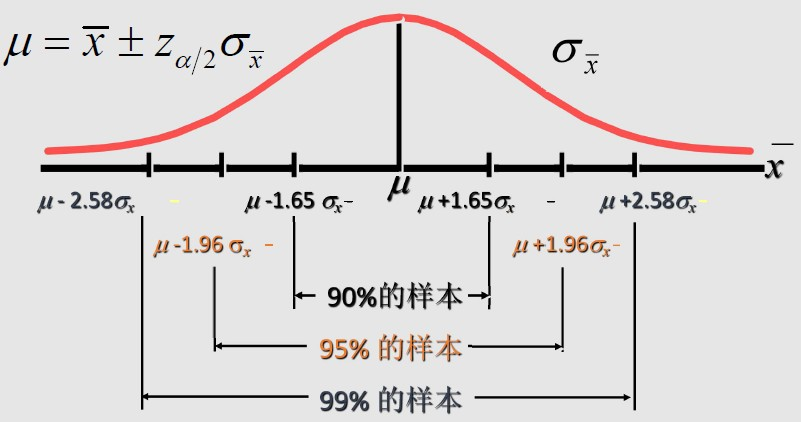
sd(t1)

t1 <- sim.fun(m=100,n=36)

mean(t1)

sd(t1)

#### 区间估计的图示



图中的1.65, 1.96, 2.58就是每一个置信水平/区间对应的分位点Zα/2

#### 与置信水平对照表

|  |  |
| --- | --- |
| Zα/2 | 置信水平 |
| 1.00 | 68.27% |
| 1.64 | 90% |
| 1.96 | 95% |
| 2.00 | 95.45% |
| 3.00 | 99.735 |

#### 置信水平

1. 将构造置信区间的步骤重复很多次, 置信区间包含总体参数真值的次数所占的比例称为置信水平

假如一次抽样中得到身高均值的95%的置信区间为[160, 170], 并不是说这个区间有95%的把握包含真实值, 或说真实值有95%的可能落在[160, 170]这个区间中. 因为真实的值身高分布和身高均值是已经确定的, 只是我们不能知道不能获得罢了, 也就是说, 这个区间要么包含真实的身高均值, 要么不包含真实的身高均值, 并不是说95%的把握包含真实的均值. 95%的置信区间指的是, 如果进行多次抽样, 得到多次身高均值95%的置信区间, 如[158, 164], [162, 170], [161, 169], 等等, 这些区间中的95%的次数能够覆盖到真实值.

2. 表示为 (1-α) % α为是总体参数未在区间内的比例, 称为显著性水平

3. 常用的置信水平值有 99%, 95%, 90%, 相应的显著性水平α为0.01, 0.05, 0.10

#### 置信区间 (confidence interval)

1. 由样本统计量所构造的总体参数的估计区间称为置信区间

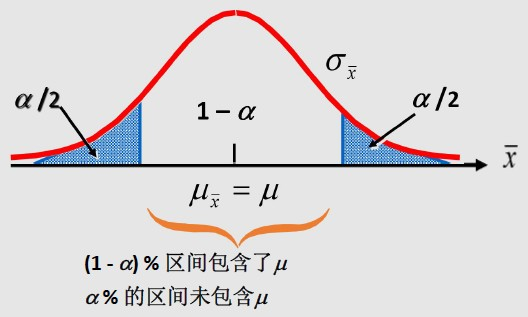
2. 统计学家在某种程度上确信这个区间会包含真正的总体参数, 所以给它取名为置信区间

3. 用一个具体的样本所构造的区间是一个特定的区间, 我们无法知道这个样本所产生的区间是否包含总体参数的真值

• 我们只能是希望这个区间是大量包含总体参数真值的区间中的一个, 但它也可能是少数几个不包含参数真值的区间中的一个

#### 置信区间与置信水平

#### 均值的抽样分布



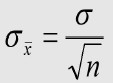
#### 影响区间宽度的因素



X-bar为区间的中心, 区间的宽度由Zα/2σx决定

1. 总体数据的离散程度, 用 σ 来测度

2. 样本容量, 样本容量越大, 区间越窄. 100个同学, 每次取5个求均值, 取100次, 区间会很宽, 而如果每次取95个同学求均值, 取100次, 区间会很窄, 很接近于总体的均值. 如果抽取的样本占总体样本量很多时, 能用很小的区间获得一个很高的置信水平.



3. 置信水平 (1-α) , 影响 z 的大小

#### 软件实现-区间估计

# 区间估计-置信区间的计算

# x为输入的数据, sigma为方差, alpha为显著性水平, 1-alpha为置信区间.

conf.int=function(x,sigma,alpha){

# 计算均值

mean=mean(x)

# 计算x中数据个数

n=length(x)

# 求出来z值, 即显著性水平alpha对应的分位点

z=qnorm(1-alpha/2,mean=0,sd=1,lower.tail=TRUE)

# 得出置信区间, 为某置信区间或显著性水平对应的刹车距离

c(mean-sigma\*z/sqrt(n),mean+sigma\*z/sqrt(n))

}

# 调用函数, 计算置信区间, 方差为4, 显著性水平为0.01, 对应99%的置信区间, alpha越小, 置信区间越宽, 修改alpha值查看效果

conf.int(cars$dist,4,0.1)

conf.int(cars$dist,4,0.05)

conf.int(cars$dist,4,0.01)

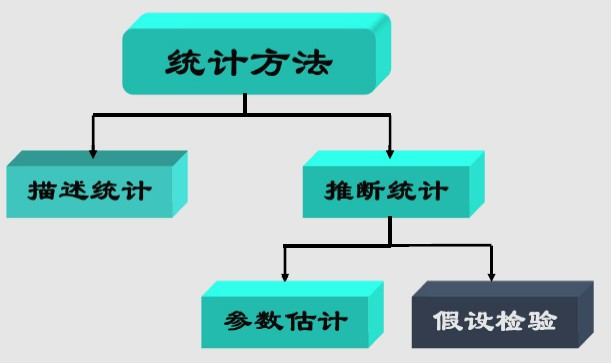
# 产生100个[20,80]之间均匀分布的随机数. 并计算置信水平

t1 <- runif(100,20,80)

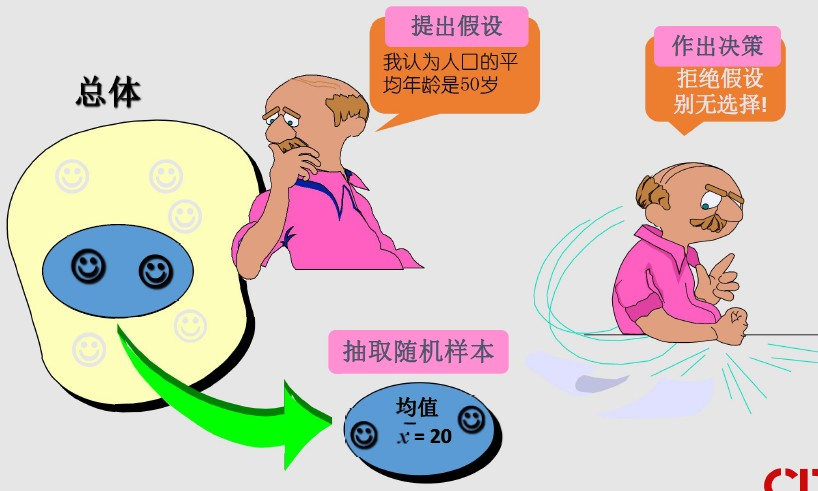
conf.int(t1,40,0.1)

## 2. 假设检验

### 假设检验在统计方法中的地位



### 假设检验的过程



第1步: 提出假设

第2步: 从总体中抽取随机样本, 计算其样本统计量

第3步: 作出决策, 拒绝原假设或无法拒绝原假设

### 原假设(null hypothesis)

1. 研究者想收集证据予以反对的假设

2. 又称"0假设"

3. 总是有符号 =, ≤ 或 ≥. 原假设中总是有等号. 只要看到了等号, 就知道原假设是什么, 要拒绝什么.

4. 表示为 H0

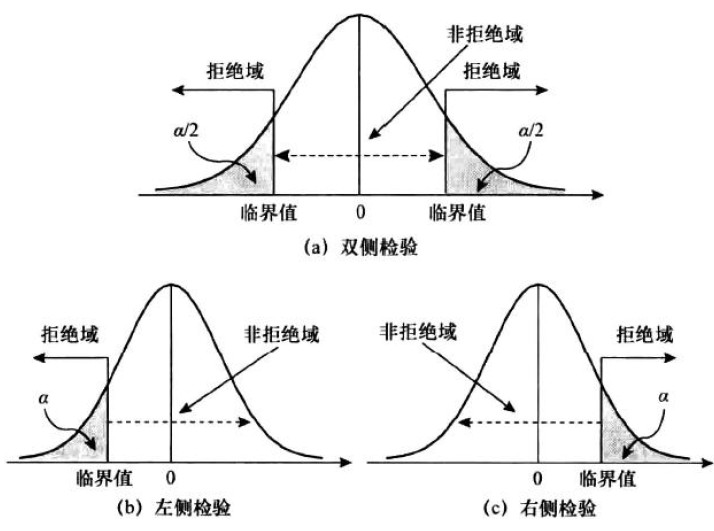
• H0: μ = 某一数值

• 指定为符号 =, ≤ 或 ≥

• 例如, H0: μ = 10cm

### 双侧检验与单侧检验(假设的形式)

|  |  |  |  |
| --- | --- | --- | --- |
| 假设 | 双侧检验 | 单侧检验 | |
| 左侧检验 | 右侧检验 |
| 原假设 | H0: μ = μ0 | H0: μ≥ μ0 | H0: μ≤ μ0 |
| 备择假设 | H1: μ ≠ μ0 | H1: μ < μ0 | H1: μ > μ0 |



### 假设检验中的两类错误

|  |  |  |
| --- | --- | --- |
| 推断\实际 | 真 | 假 |
| 真 | 正确 | 取伪错误 |
| 假 | 弃真错误 | 正确 |

1. 第Ⅰ类错误(弃真错误)

• 原假设为真时拒绝原假设

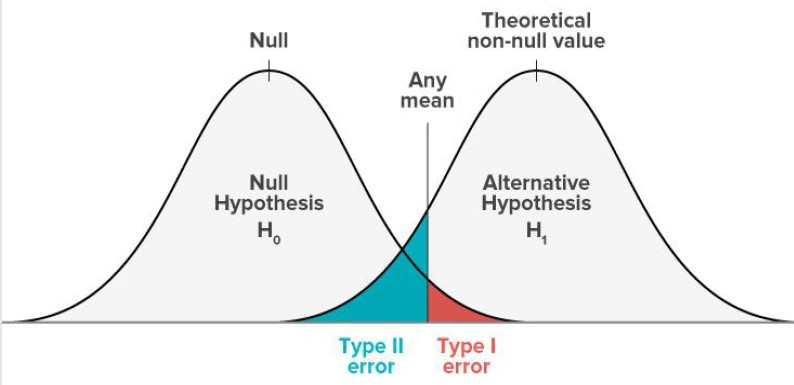
• 第Ⅰ类错误的概率记为α

• 被称为显著性水平

2. 第Ⅱ类错误(取伪错误)

• 原假设为假时未拒绝原假设

• 第Ⅱ类错误的概率记为β (Beta)



### 两类错误对数据挖掘的启示

• 准确率为99.9的模型到底好不好？

准确率只是模型的一个指标, 不是绝对的指标. 还需要考虑其它的指标. 如捕捉率, 召回率等其它指标.

以保险为例, 1W个客户, 其中有9990都是正常的投保人, 只有10个是为了骗保而投的保险, 想要把这10个人给找出来. 建立了一个模型, 假定所有人都是好人, 这个模型的准确率要达到99.0%才能把9990个人都覆盖. 虽然99.9%的概率已经很高了, 但对决策却没有什么帮助. 建模的目的是要把10个骗保的人给找出来, 得出的99.9%的概率对这个目的的实现没有任何帮助.

所以不要唯准确率是从.

• 如何在两类错误之间取舍

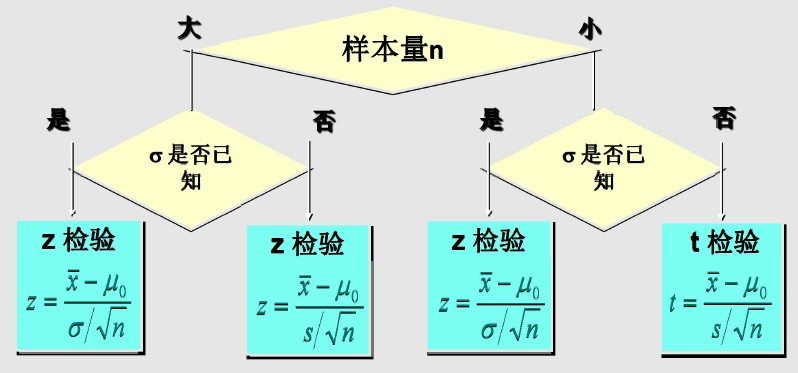
在样本量一定的情况下, 无法同时减小α和β, α的减小必然会伴随着β的增大.

α和β并不是等价的, 在有些情况下, 要尽可能的降低α, 而有些情况则要尽可能的降低β. 以得了癌症没有告诉病人和没有得癌症告诉病人得了癌症为例, 得了癌症没有告诉病人就是弃真错误, 没有得却告诉病人得了癌症就是取伪错误. 此时第I类错误的代价更大, 就要尽可能的降低α, 减少第I类错误发生的概率.

以银行的信用卡审批为例, 对一个信用好的人没有通过, 即发生了弃真错误, 银行可能只会损失几十几百元, 而如果通过了一个信用不好的人, 即发生了取伪错误, 就有可能被会套走几万元, 此时第II类错误的代价更大, 就要尽可能降低β, 减少第II类错误发生的概率.

想要同时减小α和β, 就要增大样本量, 即再去收集信息. 收集的信息越多, 就越能降低这两类的错误.

### 一个总体均值的检验(作出判断)



### 利用P值进行决策

在确定P值后, 根据给定的显著性水平, 进行拒绝原假设或不拒绝原假设的决策.

1. 当P<α时, 则在显著性水平下拒绝原假设.

2. 当P>=α时, 则在显著性水平下不拒绝原假设.

用P值作检验, 可以准确地知道检验的显著性水平, 实际上P值就是犯弃真错误的概率 , 也就是检验的真实显著性水平.

**区间估计怎么操作.**



给出一个正态分布, 中心μ加减标准差

给出一个α/2值, α/2是一个小概率,通常可以取1%, 5%, 10%, 在正轴上, 这个小概率对应着曲线下面一个点Zα/2到正无穷的面积. 也就是说, 在密度函数上切下一刀, 为了保证曲线下这一点到正无穷的面积为α/2, 这一刀应该切在哪里, 所切的位置就是Zα/2. Zα/2称为分位数, 分位数乘以方差σ, 就叫作误差, 点估计加减误差就得到了区间.

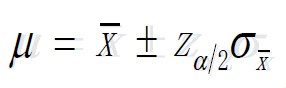
90%的样本会落在正负1.65σ范围内, 即正负1.65σ范围内曲线下的面积占总面积的90%. 正负1.65σ之外的样本各为5%, 曲线面积各占5%.

95%的样本会落在正负1.96σ范围内, 即正负1. 96σ范围内曲线下的面积占总面积的95%. 正负1. 96σ之外的样本各为2.5%, 曲线面积各占2.5%.

99%的样本会落在正负2.58σ范围内, 即正负2.58σ范围内曲线下的面积占总面积的99%. 正负2.58σ之外的样本为0.5%, 曲线面积为0.5%.

随着对概率的要求越来越高, 区间就会变得越来越宽.

想要得到一个更确切的结果, 点估计的区间就会变大.

置信区间的公式, , 只要给定了置信水平, 对应的α值就会确定, 误差Zα/2σ就会确定, 一次抽样的宽度就会确定

当把置信水平由95%变为99%, 那么, 每次抽样的抽样区间就会变得更宽, 以使得99%的抽样区间都能覆盖总体的参数. 置信水平越高, 区间就会变得越宽, 每次抽样的区间范围会变得更大.

如钥匙找不到了, 95%的可能性在家里, 99%的可能性在小区里, 100%的可能性在整个城市里.

**置信水平(confidence level)**

1. 将构造置信区间的步骤重复很多次, 置信区间包含总体参数真值的次数所占的比例称为置信水平

2. 表示为(1-α)%

• α为总体参数未在区间内的比例, 中间的比例, 如90%, 为1-α.

3. 常用的置信水平值有99%,95%,90%, 相对应的α分别为0.01, 0.05, 0.10

置信水平就是落在置信区间中的概率.

区间估计中的α

给出一个α/2值, α/2是一个小概率, 通常可以取1%, 5%, 10%, 在正轴上, 这个小概率对应着曲线下面一个点Zα/2到正无穷的面积. Zα/2称为分位数, 分位点到正无穷的面积为α/2

### 软件实现

某汽车厂商销售的一款SUV, 宣传称百公里油耗为8.6L, 20位车主测试数据如下, 其宣传是否可信(数据集caroil)

# 假设检验

# 正态检验

?pnorm

# 自定义函数进行正态检验.

u.test<-**function**(a,mu,sigma,alternative="twoside"){

# Se为方差

Se=sigma/sqrt(length(a))

# 计算u统计量

u=(mean(a)-mu)/Se

# 判断是双边检验还是单边检验

**if** (alternative=="twoside") {

p=2\*(1-pnorm(abs(u)))

} **else** **if** (alternative=="less") {

p=pnorm(u)

} **else** {

p=1-pnorm(u)

}

# 返回计算得到的u值和p值

**return**(list(u=u, p=p))

}

# 检验汽车的百公里油耗

# 设置工作目录

setwd("D:\\R\_edu\\data")

# 读取数据

oil <- read.csv("caroil.csv")

# 对数据进行统计

summary(oil)

# 调用函数, 进行统计检验.

# 假设8.4是声称的油耗, 2.1为厂方或卖方给出的总体标准差.

u.test(oil$oil,8.6,2.1,alternative="twoside")

# $u

# [1] 1.717513

#

# $p

# [1] 0.08588545

# 这是一个双侧检验, 等号一定在原假设上, 即原假设为真实的油耗等于声称的油耗, 假设显著性水平为α=0.1, 对应的Z值即临界值为1.65, 对应的置信水平为90%. 通过计算得到的p值为0.085, 小于0.1, 即落在了拒绝区中, 所以就要拒绝原假设, 即认为在0.1的显著性水平下, 油耗不是8.6. 但是如果显著性水平为0.05, 得到的p值为0.085, 大于0.05, 没有落在拒绝区中, 所以就不能拒绝原假设, 就认为二者的方差相等.

## 3. 两个样本检验

### 两个总体参数的检验

• 两个独立总体. 如两个班的成绩.

• 两配对总体

### 两独立样本T检验

两独立样本T检验的目的是利用来自两个正态总体的独立样本, 推断两个总体的均值是否存在显著差异。如要对两个班的成绩进行对比, 分别计算出总体的均值即可, 为什么还要使用两个独立样本的假设检验呢? 注意假设检验不是要计算两个总体之间有没有差异, 而是计算它们之间有没有显著性差异, 或说是有没有本质上的差异, 对于两个班级的成绩来说, 就是检验这两个班级作为两个群体他们的成绩是不是已经不在一个Level上了.

当样本的总体是非正态分布时, 应考虑选择非参数检验的方法。但当两个样本的容量足够大时, 一般认为大于30或50时, 也可以用Z′检验(Z检验的近似检验) 。

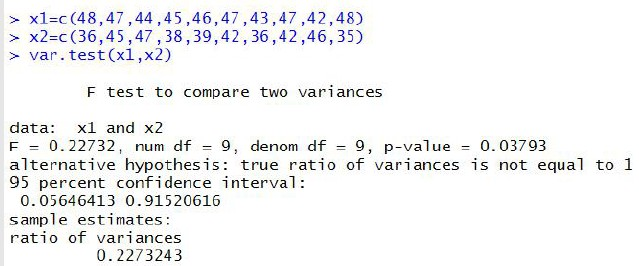
#### 实例演练

• 检验两销售团队的业绩是否有明显差异

• step1 正态性检验

• setp2 检验方差

方差是数据的离散程度, 反映的是代表值的代表性, 如果数据的离散程度很大, 代表值如均值的代表性就会很差. 为了验证两个销售团队的业绩是否有显著性差异, 就首先要保证做对比的参数是从这两个团队中选择出来的代表, 能够代表这两个团队参赛. 并且只有这两个团队选择出来的代表值的代表能力相当, 才能进行比较. 如平均值相等收入差距很小的国家或地区与平均值相等但收入差距很大的国家或地区之间就不具有可比性, 它们的方差相差很大, 就不能进行方差分析. 只有保证方差没有显著性差异, 才能进行方差分析.

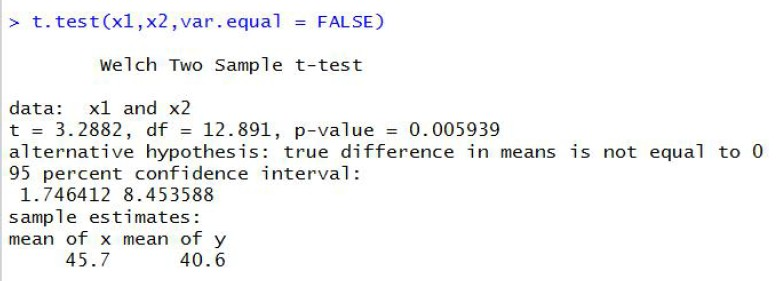


如果条件很宽泛, 如在α=0.01的显著性水平上, 就不能拒绝原假设, 不能认为二者的方差具有显著性差异, 即认为二者的方差相等. α越小, 对应的Z分位点的值就越大, 对应的置信区间就越宽. 落在置信区间中的概率即置信水平就越大. 就可以认为是可以接受的条件就更宽泛.

如果显著性水平α由0.01变为0.05, 此时的置信区间就会变窄, 可接受的值就会越少, 可接受的条件就会变得很严格,就要拒绝原假设, 即认为二者的方差不相等.

• step3 均值检验

如果上一步方差检验中得到的方差相等, equal = True, 如果方差不等, equeal = False.



在方差不等的情况下, 得到的p值小于0.05, 就会拒绝原假设, 认为两个团队的值具有显著性差异.

# 两个样本的检验-独立样本T检验

# 检验两个团队的销售业绩. T检验的数据要符合正态分布

x1=c(48,47,44,45,46,47,43,47,42,48)

x2=c(36,45,47,38,39,42,36,42,46,35)

# 首先进行描述性统计分析, 对数据整体上有一个了解

mean(x1)

mean(x2)

# 使用qq图看数据是否大致在一条直线上来大致判断是不是符合正态分布

qqnorm(x1)

qqline(x1)

qqnorm(x2)

qqline(x2)

# 使用ks检验来判断数据是否符合正态分布

# 得到的p值为0.66, 不能拒绝数据符合正态分面的原假设,

ks.test(x1,"pnorm",mean=mean(x1),sd=sd(x1))

# 得到的p值为0.97

ks.test(x2,"pnorm",mean=mean(x2),sd=sd(x2))

# 使用shapiro检验来判断是否符合正态分布

shapiro.test(x1)

shapiro.test(x2)

# 专门用来做正态检验的包

library(nortest)

# AD正态性检验

# pearson卡方正态性检验

ad.test(x1)

ad.test(x2)

#检验方差是否一致

# ?var.test

var.test(x1,x2)

# 根据方差相等的检验结果，进行均值比较

t.test(x1,x2,var.equal = **FALSE**)

# 根据方差不相等的检验结果，进行均值比较

t.test(x1,x2,var.equal = **TRUE**)

# 检验两总体是否一致, 有很多种分布, 检验两个总体是不是来自相同的分布. 在0.1的水平上拒绝原假设, 即认为不是来自同一分布的.

ks.test(x1,x2)

### 配对样本T检验

在数据分析时经常会遇到这样的问题:

某种教学方法是否对教学有效, 也就是确实能提高学生成绩；

某种训练是否对接受训练的人的某一身体机能有改善作用；

或者某一种药物对某种病的治疗是否有效果等等。

针对以上问题, 我们通常就会采取配对样本T检验的方法进行分析。

• t.test(x, y = NULL, alternative = c("two.sided", "less", "greater") , mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95, ...)

如果是独立样本的T检验, paired = FALSE, 如果是配对样本的T检验, paired = TRUE.

#### 实例演练

• 某种训练方式减肥效果可以至少减肥8.5kg以上

# 配对T样本的检验

before = c(94.5,101,110,103.5,97,88.5,96.5,101,104,116.5)

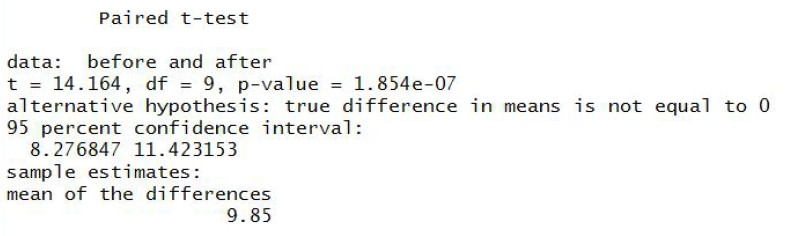
after = c(85,89.5,101.5,96,86,80.5,87,93.5,93,102)

t.test(before,after,paired=T)

t.test(before-8.5,after,paired=T)

# 得到的结果为0.08414, 在0.05的水平上接受原假设.

配对的意思就是两个样本对应位置上的元素一一对应, 如94.5对应85, 101对应89.5…



由于宣称的是至少减肥8.5kg以上, 得到的区间是[8.276847, 11.423153], 8.5落在了这个区间中.

### 小结

|  |  |  |  |
| --- | --- | --- | --- |
| 方法 | 检验的目的 | 方法的前提条件 | 举例 |
| 单样本T检验 | 检验样本均值与已知总体均值之间是否存在差异 | 总体服从正态分布 | 某地区高考数学成绩与全国数学高考成绩均值是否存在显著差异 |
| 独立样本T检验 | 总体正态分布下, 两个独立样本均值之间是否存在显著差异 | 进行比较的两个样本是独立的, 并且服从正态分布 1.F检验方差齐性 2.T检验 | 某大学随机抽取若干个大学一年级学生, 分析他们的大学入学考试成绩在性别上是否存在显著差异。 |
| 配对样本T检验 | 两配对样本的均值之间是否存在显著差异 | 1.配对要求两组同质, 受试对象配成对子或同一受试对象分别接受两种不同的处理 2.样本来自的两个总体必须服从正态分布 | 1.针对实验前学习成绩和智商相同的两组学生, 分别进行不同教学方法的训练, 比较参与实验的两组学生的学习成绩是否存在显著差异。2.算法优化前后, 算法性能是否有显著差异。 |

### 非参数检验

表1-1 四种测量层次及适宜的统计量

|  |  |  |  |
| --- | --- | --- | --- |
| **测量层次** | **数学性质** | **描述统计量** | **适合的统计检验** |
| 定类尺度  分类数据 | ＝、≠ | 众 数 | 非参数检验 |
| 频 率 |
| 列联系数 |
| 定序尺度  顺序数据 | ＝、≠ | 中位数 |
| 百分位数 |
| ＜、＞ | Kendall τ |
| Spearman r |
| 定距尺度 | ＝、≠ | 平均值 | 参数检验与非参数检验 |
| ＜、＞ | 方 差 |
| ＋、－ | Pearson r |
| 定比尺度 | ＝、≠ | 几何均值 |
| ＜、＞ | 标准差系数 |
| ＋、－ | 多重相关系数 |
| ×、÷ |  |

对不同类型的数据, 采用的统计方法是不同的. 对于分类数据和顺序数据, 一般使用的是非参数检验.

而定距数据和定比数据, 是根据计算的描述统计量的不同选择使用参数检验或非参检验, 如检验一组数据的众数是否等于特定值, 使用非参数检验, 验证公司工资的中位数是否为2W, 使用非参数检验. 定距数据和定比数据为数值型数据.

如果数值型数据不符合正态分布的话, 就要使用非参数检验.

进行检验的一个原则: 能用参数检验就用尽量使用参数检验, 实在不能使用参数检验时才使用非参数检验.

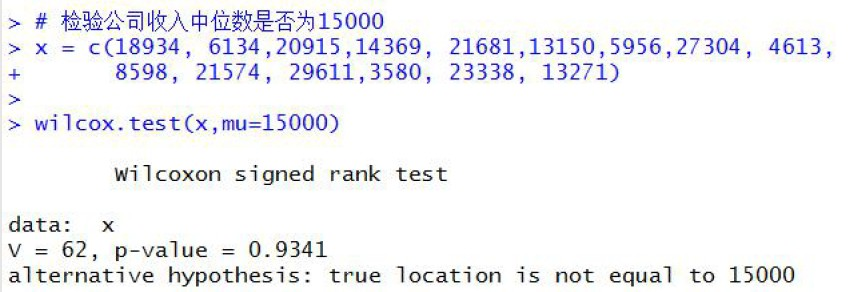
拿到数据之后, 首先要检验数据是否服从正态分布. 如果不服从正态分布, 就对数据进行转换, 如开根号, 取对数, 取平方等, 如果经过转换之后的数据服从正态分布, 就对转换之后的数据使用正态分布进行参数检验. 如果无论使用什么方法都不能把数据转换为服从正态分布的数据, 就只能使用非参数检验. 非参数检验的课程.

如果拿到数据不进行检验或转换, 可以使用正态分布检验或转换后使用正态分布而不使用正态分布, 那么使用非参数检验得的结果一般不会太精确.

### 非参数检验拓展

#### 单样本非参数检验

在对单样本进行T检验的时候, 如果样本均值不服从正态分布, 此时就要使用单样本的非参数检验了, 但是就不是对均值进行检验了, 而是对中位数进行非参数检验, 判断中位数是不是15000.



# 非参数检验

# 检验公司收入中位数是否为15000

# 首先要检验数据是否服从正态分布. 如果不服从正态分布, 就对数据进行转换, 如开根号, 取对数, 取平方等, 如果经过转换之后的数据服从正态分布, 就对转换之后的数据使用正态分布进行参数检验. 如果无论使用什么方法都不能把数据转换为服从正态分布的数据, 就只能使用非参数检验.

x = c(18934, 6134,20915,14369, 21681,13150,5956,27304, 4613,

8598, 21574, 29611,3580, 23338, 13271)

wilcox.test(x,mu=15000)

#### 独立样本非参数检验

如果不满足独立样本t检验的条件, 就要使用独立样本非参数检验和两个独立样本的非参数检验. 下面两种方法都是针对两个独立样本的非参数检验的, 只是数据的格式不一样.

如两个班级的成绩在一列中上下排列. 第一列是成绩, x1, x2, x3, … , 第二列是成绩对应的班级. 1班, 1班, …, 2班, 2班, 2班, …, 就使用第1种方法.

# independent 2-group Mann-Whitney U Test

wilcox.test(y~A)

# where y is numeric and A is A binary factor

如果两个班级的成绩分别在在两列中, 如第一列是一班的成绩, 第2列是二班的成绩, 就使用第2种方法.

# independent 2-group Mann-Whitney U Test

wilcox.test(y, x) # where y and x are numeric

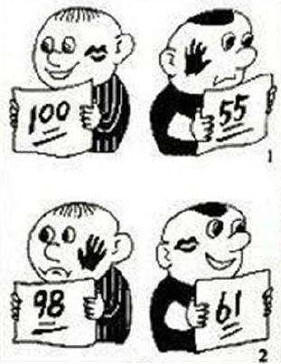
#### 配对样本非参数检验

如果不满足前面的配对样本t检验的条件, 就要使用配对样本非参数检验.

# dependent 2-group Wilcoxon Signed Rank Test

wilcox.test(y1, y2, paired=TRUE) # where y1 and y2 are numeric

### 卡方独立性检验



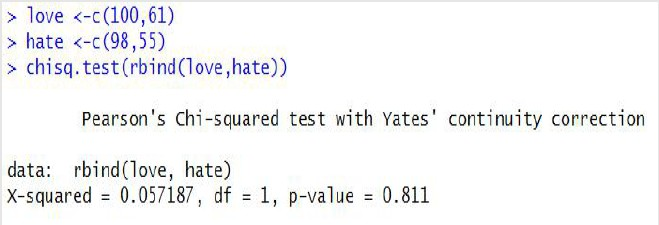
挨不挨打与头发长短之间的关系.

• 检验两个分类变量之间的关联性

如大学一个班级中性别的分布与本地和外地的分布有没有关联. 习惯上认为男生更倾向于选择外地的大学. 女生更倾向于选择本地的大学. 检验这种习惯上的思想是否可靠.

• chsiq.test()

使用R进行分析时一定要明白为什么要做这种检验, 明白其原理, 并知道如何对结果进行解读, 转化为业务的理解. 具体的代码写起来很简单, 也不需要自己花心思去调优.



love <-c(100,61)

hate <-c(98,55)

chisq.test(rbind(love,hate))

## 4. 方差分析

### 问题来源

#### 不同超市位置的销售额是否有显著差异？



#### 为什么不用两两检验？

若做两两的比较, 则需要做3次比较。这样做不仅繁琐, 而且每次检验犯第Ι类错误的概率都是α, 作多次检验会使犯第Ι类错误的概率相应地增加, 检验完成时, 犯第Ι类错误的概率会大于α。如有5个样本, 就要进行C(5,2) = 10次比较. 假设每次检验犯第Ι类错误的概率都是α=0.1, 不犯错的概率就是0.9, 10次检验不犯错误的概率就是0.9^10 = 0.348678, 10次检验至少犯1次错误的概率就是1-0.9^10= 0.651322, 即至少会犯一次错误的概率就会很大.

方差分析是检验多个总体均值是否存在显著差异的统计推断方法, 是假设检验在多个总体问题的进一步扩展。

### 什么是方差分析

方差分析(ANOVA) (analysis of variance)

1. 方差分析的基本原理是在20世纪20年代由英国统计学家Ronald A.Fisher在进行实验设计时为解释实验数据而首先引入的

方差来源于实验科学 DOE, design of experiment, 实验设计. 如设计不同的日照时间和日照强弱, 温度和湿度, 看对植物生长的影响.

2. 检验多个总体均值是否相等

• 通过分析数据的误差判断各总体均值是否相等

3. 研究分类型自变量对数值型因变量的影响

• 一个或多个分类型自变量, 超市的位置.

• 两个或多个 (k 个) 处理水平或分类, 如可以再添加上工作日和非工作日的分类.

• 一个数值型因变量, 不同位置超市的销售额

4. 有单因子方差分析和双因子方差分析

• 单因子方差分析: 涉及一个分类的自变量, 只考虑超市位置

• 双因子方差分析: 涉及两个分类的自变量, 考虑超市位置和工作日与非工作日.

#### 重要概念

在方差分析中, 所要检验的对象称为因素或因子(Factor), 即要检验的变量.

因子的不同表现称为水平或处置(Treatment), 三个不同的超市位置就称为不同的水平.

每个因子水平下得到的样本数据称为观察值。如销售额就是观察值.

#### 方差分析的基本假定

1. 正态性(normality) 。每个总体都应服从正态分布, 即对于因子的每一个水平, 其观测值是来自正态分布总体的简单随机样本

• 检验总体是否服从正态分布的方法有很多, 包括对样本数据作直方图、茎叶图、箱线图、正态概率图做描述性判断, 如果变换之后也不符合正态分布, 也可以进行非参数检验等. 非参检验中也有与方差分析类似的方法, 称为k个独立个体的非参检验.

2. 方差齐性(homogeneity variance) 。各个总体的方差必须相同, 对于分类变量的个水平, 有

3. 独立性(independence) 。每个样本数据是来自因子各水平的独立样本(该假定不满足对结果影响较大). 每个超市之间互不影响. 但实际中比较难以确定样本的独立性, 可以了解一下取数的规则, 如商业区取周一之后, 对应居民区的选择是否可以随意选择. 如果商业区取周一的数据之后, 居民区就不能取周一的数据, 就不能认为二者的数据是独立的.

在具体做方差分析的过程中, 要一步步的进行检验. 看是否满足条件.

#### 方差分析的基本原理

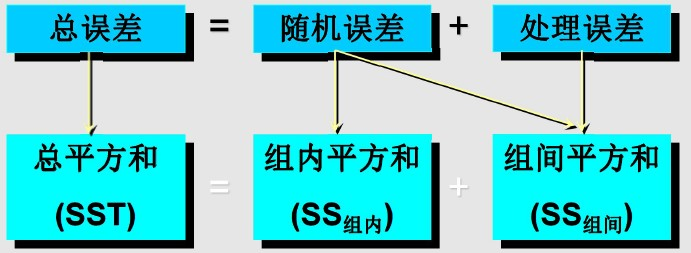
误差分解

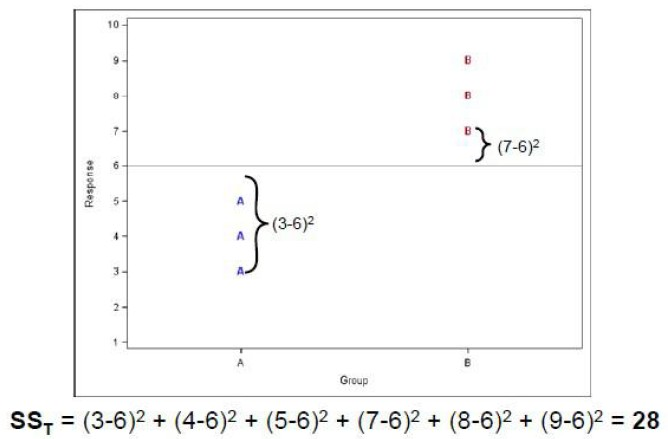
误差平方和的分解及其关系:

同一个超市每天的销售额也有波动, 这种波动称为随机误差.

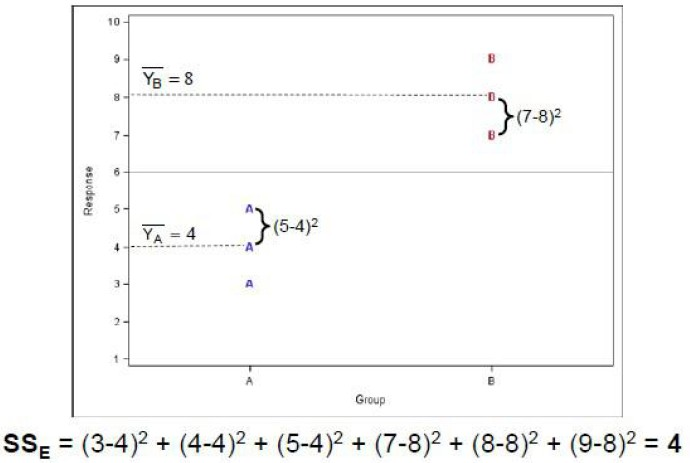
因为超市位置不同造成的销售额间的差异称为处理误差. treatment

如果一个超市的随机误差与处理误差相差无几, 如商业区销售额的随机波动和商业区和居民区之间的销售差值差不多, 那么超市位置对销售额的影响就可以忽略了. 所以要把误差分为随机误差和处理误差, 如果处理误差远远大于随机误差, 就可以认为不同的超市位置对销售额的影响是显著性的. 否则如果随机误差大于或等于处理误差, 那么就认为不同位置对超市的销售额的影响是不显著的.

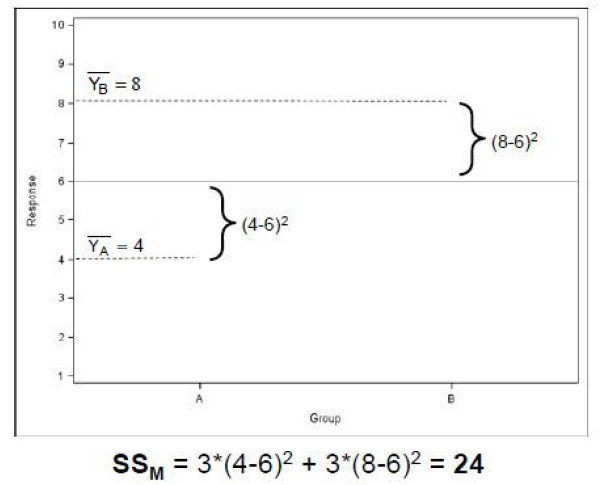




A超市3天的销售额为3,4,5, B超市3天的销售额为7,8,9, AB超市总的均值为6. 所以总的误差为SST = 28.



随机误差SSE为各组内误差之和



处理误差SSM, (4-6), (8-6)为小组均值-总均值, 3为样本量

这样就把总均值28分解为随机误差4与处理误差24.

### 方差分析的步骤

#### 提出假设

1. 一般提法

• H0: μ1 = μ2 = … = μk

• 自变量对因变量没有显著影响, 不同位置超市的销售额是相同的.

• H1 : μ1, μ2, …, μk 不全相等

• 自变量对因变量有显著影响, 不同位置超市的销售额不全相同.

2. 注意: 拒绝原假设, 只表明至少有两个总体的均值不相等, 并不意味着所有的均值都不相等

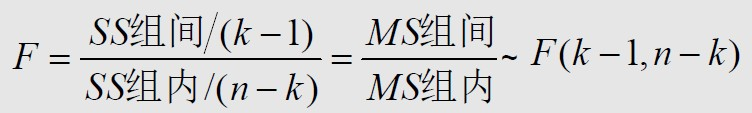
#### 构造检验的统计量F

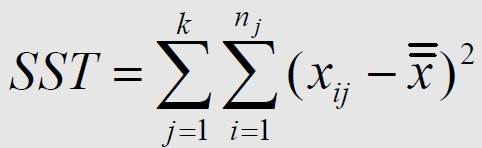
1. 将组间均方和MS组间除以组内均方和MS组内即得到所需要的检验统计量F

2. 当H0为真时, 二者的比值服从分子自由度为k-1、分母自由度为 n-k的 F 分布, 即



#### 构造检验的统计量F





#### 做出决策

将统计量的值F与给定的显著性水平α的临界值Fα 进行比较(或计算出统计量的P值) , 做出决策

• 若F<Fα, 或P值小于α, 则拒绝原假设H0, 表明均值之间的差异是显著的, 所检验的因子对观察值有显著影响

• 若F>Fα, 或P值大于α, 则不能拒绝原假设H0, 无证据表明所检验的因子对观察值有显著影响

#### 多重比较

1. 在拒绝原假设的条件下, 通过对总体均值之间的配对比较来进一步检验到底哪些均值之间存在差异

2. 比较方法有多种, Fisher提出的最小显著差异方法, 简写为LSD. 大概有10几种.

|  |  |  |  |
| --- | --- | --- | --- |
| 方法 | 优点 | 缺点 | 适用 |
| LSD | 检验灵敏性高 | 没有对犯一类错误的概率问题加以控制 | 各总体方差相等的情况 |
| Bonferroni | 对犯一类错误的概率问题加以控制 |  | 样本组较小时效果较好 |
| Tukey | 对犯一类错误的概率问题加以控制 |  | 观测值个数相等及各总体方差相等 |
| Scheffe |  | 不如Tukey方法灵敏 |  |
| S-N-K |  |  | 观测值个数相等 |

### 实例演练

基于teaching数据集, 研究不同mothod是否对成绩有影响？

# 方差分析

# 比较不同的教学方法的效果. Control, Instructions, Lecture. Effectiveness0, 1, 2可以分别认为是数学, 语文, 英语成绩.

setwd("D:\\R\_edu\\data")

library(haven)

teach <- read\_sas("teaching.sas7bdat")

names(teach)

#正态性检验

shapiro.test(teach$Effectiveness0)

#方差齐性检验

# ?bartlett.test

# ?leveneTest

# var.test仅用于检验两变量的方差齐性

bartlett.test(Effectiveness0~Teaching,data=teach)

library(car)

leveneTest(Effectiveness0~Teaching,data=teach)

# 方式一, 单一方差分析, 比较连续型变量Effectiveness0和分类型变量教学方法之间的关系. var.equal = TRUE表示方差是相等的.结果为0.8027, 不同教学方法下该成绩不具有显著性差异. 可以认为不同方法下成绩是相等的.

oneway.test(Effectiveness0~Teaching,data=teach,var.equal = **TRUE**)

oneway.test(Effectiveness1~Teaching,data=teach,var.equal = **TRUE**)

# Effectiveness2的p值为0.001374很小, 拒绝原假设. 认为不同教学方法得到的成绩不全相等. 至于哪两个相等或不相等, 需要进一步进行多重比较才能判定.

oneway.test(Effectiveness2~Teaching,data=teach,var.equal = **TRUE**)

# 方式二

aov(Effectiveness0~Teaching,data=teach)

summary(aov(Effectiveness0~Teaching,data=teach))

# 方式三

anova(lm(Effectiveness0~Teaching,data=teach))

oneway.test(Effectiveness1~Teaching,data=teach,var.equal = **TRUE**)

### 多重比较软件实现

• #正态性检验

• shapiro.test(teach$Effectiveness0)

• #方差齐性检验

• bartlett.test(Effectiveness0~Teaching, data=teach)

• #方差分析

• oneway.test(Effectiveness0~Teaching, data=teach, var.equal = TRUE)

# 多重比较

# 前提是拒绝原假设

oneway.test(Effectiveness2~Teaching,data=teach,var.equal = **TRUE**)

# 关于农业科学的一个包

library(agricolae)

# 使用方式二做方差分析.

mo <- aov(Effectiveness2~Teaching,data=teach)

# 使用LSD进行多重比较

g1 <- LSD.test(mo,"Teaching",p.adj="none")

g1

# plot是泛型函数, 不用事先定义x,y

plot(g1)

g2 <- LSD.test(mo,"Teaching",p.adj="bonferroni")

# 得出的结果是control和lecture是类似的, lecture和instructions是类似的. 但为什么却不能得出三者都是类似的呢. 其实我们计算的是看比较的二者之间有没有显著性差异. 看起来二者是类似或相等的, 只是说二者的差异还没有到显著性的程度. 假设有3个数据, 规定当数据之间的差异小于0.5时, 认为它们之间没有显著性差异. A=1.0, B=1.3, C=0.7, 则B与A, C与A之间都没有显著性差异, 但却不能认为B和C之间也没有显著性差异.

g2

plot(g2)

# R 结语

上面的方法和步骤一方面可以用来进行独立的统计分析, 另一方面, 可以作为后继建模的基础. 如信用卡的例子, 可以检验信用的好坏与性别之间有没有关系, 就可以使用卡方独立性检验来进行, 如果性别对信用有显著性的影响, 就说明性别对信用好坏有预测作用. 如果想要检测收入到信用好坏的影响, 因为收入是连续型变量, 好坏是分类型变量, 可以使用独立样本的t检验. 假如得到的结果是信用好坏的人的收入之间没有显著性差异, 就说明收入的高低对信用的好坏没有显著性影响. 这里学习的描述性和推断性统计分析更多的是作为后面决策树, 神经网络, 逻辑回归模型中变量筛选的很有用的方法和步骤.