# 数据分析与处理技术

## 数据分析工具

南京审计大学商学院物流管理系

# 案例：Titanic数据集分析

1912年，行驶在大西洋上的Titanic巨型邮轮触碰冰山引发了沉船灾难

Titanic数据集记录了登船人员的身份信息，数据分析的经典入门案例之一。

Titanic邮轮(电影)

```
> dim(Titanic)
[1] 891  12
```

泰坦尼克灾难留给世界的震惊持续了百年之久，通过对登船人员身份记录数据的分析能为我们揭开许多问题的答案。
登船人员人数，以及年龄、性别等比例。生还者的身份具有哪些特征。

作为教学案例，仅提取这个数据集的一部分进行演示操作，请根据演示自行操作完整数据集。

仅截取前20行数据的子集作为操作演示

```
> titanic<-Titanic[1:20,]
```

```
> head(Titanic,20)
```

快速理清数据的大致情况

```
> str(titanic)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':       20 obs. of  12 variables:
 $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
 $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
 $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
 $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florenc
e Briggs Thayer)" "Heikkinen, Miss. Laina" "Futrelle, Mrs. Jacques Heath (Lily May
Peel)" ...
 $ Sex        : chr  "male" "female" "female" "female" ...
 $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
 $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
 $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
 $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
 $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
 $ Cabin      : chr  NA "C85" NA "C123" ...
 $ Embarked   : chr  "S" "C" "S" "S" ...
```

# 尝试一个简单问题

```
> mean(titanic$Age)
[1] NA
```

思考：为什么会出现空缺?

为了实现对均值的计算，打开mean函数中的na.rm参数

```
> mean(titanic$Age,na.rm = T)
[1] 28
```

na.rm在常见计算函数中是通用的

在Rstudio中使用View可以像Excel那样观察数据，可以使用筛选功能

```
> View(titanic)
```

但如果需要手动直接修改则需要使用fix函数打开数据编辑界面

```
> fix(titanic)
```

现实中的数据很难能够非常规则符合使用要求，在使用之前需要进行**数据清洗**，数据清洗在实际使用中占据了绝大部分工作时间。

数据质量中的一个关键问题：是否存在空缺?

```
> any(is.na(titanic))
[1] TRUE
```

检验空缺无法用比较运算，但有专门的is.na函数等效于比较运算

```
> titanic1<-titanic[-is.na(titanic$Age),]
```

使用is.na函数剔除某关键变量中的空值

```
> na.omit(titanic)
```

如果要清除所有包含空值的对象则更容易

# 初步分析

确定生还者变量中没有空值

```
> any(is.na(titanic$Survived))
[1] FALSE
```

一共多少人在这场灾难中生还

```
> sum(titanic$Survived==1)
```

生还者占了多大比例

```
> sum(titanic$Survived==1)/nrow(titanic)
[1] 0.5
```

此处操作并没有使用变量存储，即计算结束后便丢弃

登船人员的性别分布如何查看

```
> table(titanic$Sex)

female   male
    11      9
```

进一步利用table理清更多信息

如果性别和生还者的分布情况呢?

```
> table(titanic$Sex,titanic$Survived)

        0 1
female 2 9
male   8 1
```

如果能加上边际总量则更好

```
> a<-table(titanic$Sex,titanic$Survived)

> class(a)
[1] "table"
```

此时变量a是一种叫做table类型的变量

```
> addmargins(a)

          0  1 Sum
female    2  9  11
male      8  1   9
Sum      10 10  20
```

与addmargins类似，转成频率形式

```
> prop.table(a)

            0    1
female 0.10 0.45
male   0.40 0.05
```

# 问题总结

快速观察数据结构是处理一个数据集前期必要处理

分析数据前先解决数据质量问题，检查是否存在空值

mean函数是通用型函数，其中许多参数与同类参数是相通的

# 对样本进行排序

```
> titanic[order(titanic$Age),]
```

```
# A tibble: 20 x 12
   PassengerId Survived Pclass Name  Sex     Age SibSp Parch Ticket  Fare
         <int>    <int>  <int> <chr> <chr> <dbl> <int> <int> <chr>  <dbl>
 1           8        0      3 Pals… male      2     3     1 349909 21.1
 2          17        0      3 Rice… male      2     4     1 382652 29.1
 3          11        1      3 Sand… fema…     4     1     1 PP 95… 16.7
 4          10        1      2 Nass… fema…    14     1     0 237736 30.1
 5          15        0      3 Vest… fema…    14     0     0 350406  7.85
 6          13        0      3 Saun… male     20     0     0 A/5. …  8.05
```

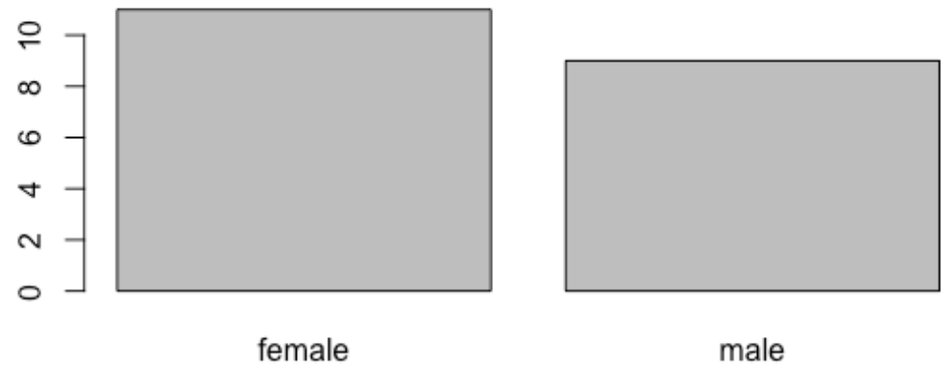
问题：提取年龄最小的5个乘客身份信息

# 可视化处理

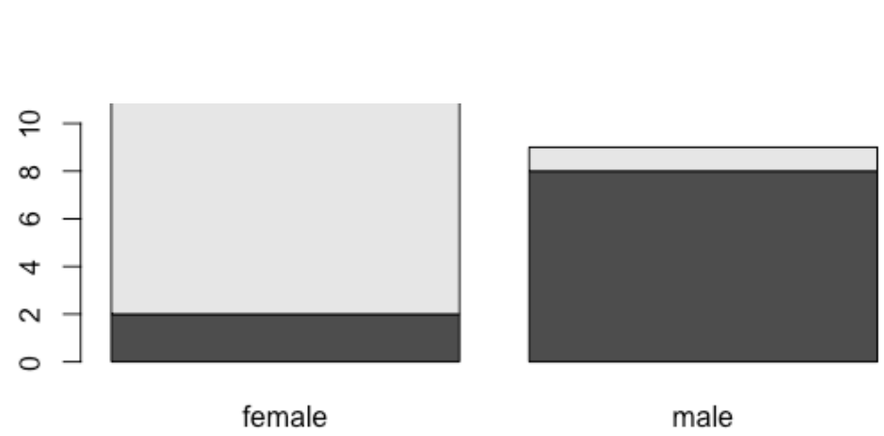
许多情况下，一张图片展示的信息胜过数字

使用barplot画出刚才生成的表

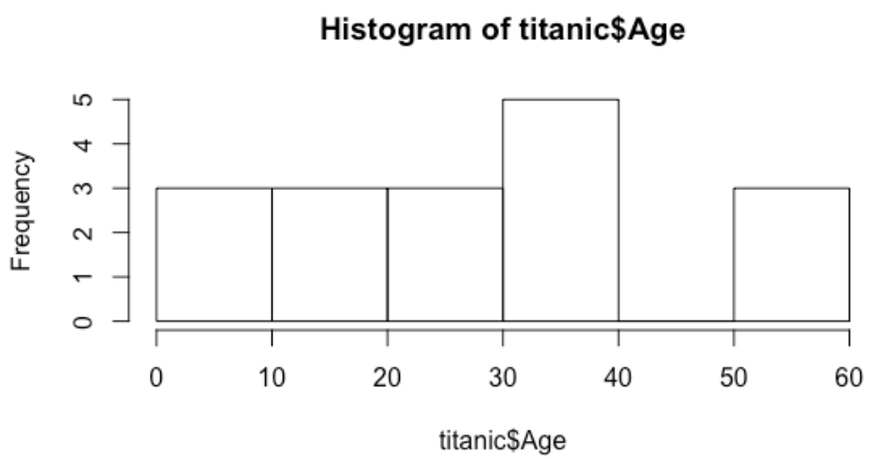
```
> a<-table(titanic$Sex)
> barplot(a)
```




```
> barplot(table(titanic$Survived,titanic$Sex))
```




```
> hist(titanic$Age)
```



# 数据的添加与修改

例如为各个对象增加一个国籍变量

```
> y<-rep(c('US','UK'),10)
> y
 [1] "US" "UK" "US" "UK" "US" "UK" "US" "UK" "US" "UK" "US" "UK" "US" "UK"
[15] "US" "UK" "US" "UK" "US" "UK"
```

cbind函数将这个向量拼接到了数据框中

```
> titanic<-cbind(titanic,y)
```

与cbind相对，rbind则用于拼接行，即加入新对象

但是数据中显示新变量名与刚才的向量一样

```
> names(titanic)[ncol(titanic)]<-'Nationality'
```

注意：修改变量名应该对names的结果加索引，例如

```
> names(titanic)[1]<-'id'
```

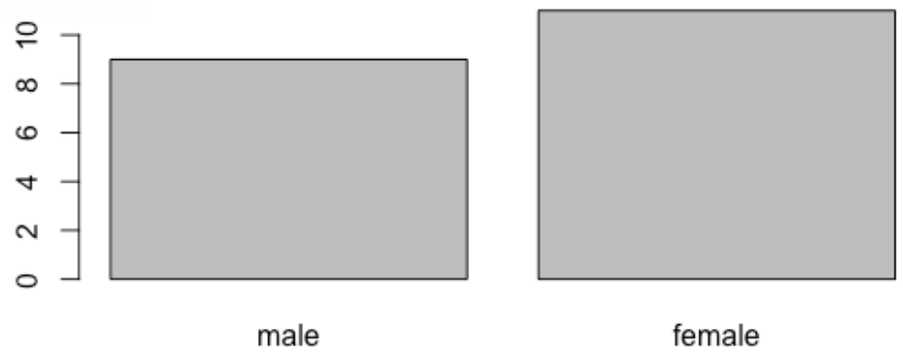
# 拆分数据

分析需要，我们经常需要从原数据中拆分出多个子集进行对比

例如将数据集拆分成男性和女性两个组

```
> male<-titanic[which(titanic$Sex=='male'),]
> female<-titanic[-which(titanic$Sex=='male'),]

> barplot(c(nrow(male),nrow(female)),names.arg  =c('male','female'))
```



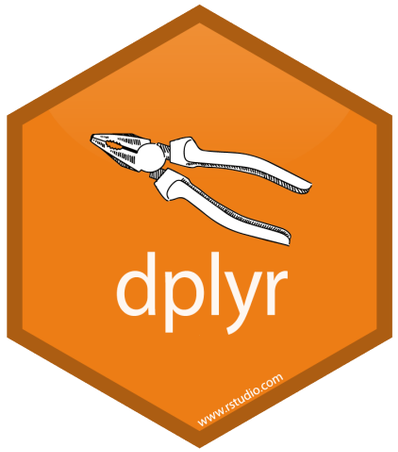

拆分样本的关键在于元素定位，主要用途在于一方面比较不同类别样本，另一方面做关联分析或建模时候做对照

针对data.frame更为快捷的方式是使用subset取子集方式

```
> subset(titanic, Pclass>=2, select=c(Age,Name))
```

# 数据框工具dplyr

工具包dplyr是一个专门用于data.frame类型的package，包含排序、筛选、取子集以及分组统计常用功能。目前已经与ggplot2、readr、tidyr等常用包归并在一个tidyverse工具包中。




```
> install.packages("dplyr")
```

```
> library(dplyr)
```

tidyverse系列的包定义了一个增强型的data.frame变量类型tbl

```
> male<-tbl_df(male)
```

```
> class(male)
[1] "tbl_df"     "tbl"        "data.frame"
```

tbl同时也是data.frame类，解决了许多原始数据框类型存在的缺陷

开发者主页https://dplyr.tidyverse.org

dplyr中的主要功能函数：

- mutate( )　融合列
- select( )　选择变量
- filter( )　筛选
- summarise( )　作用于数据框的泛函
- arrange( )　排序

- n( )
- desc( )
- filter( )
- summarise( )
- arrange( )

dplyr包中主要函数的格式非常规则，即

函数名(数据框名称，条件，条件，……)

```
> filter(titanic,Pclass==1,Age>20)

> arrange(titanic,Age,desc(Pclass))
```

# dplyr的分组处理：

```
> titanic2<-group_by(titanic,Pclass)
> summarise(titanic2,index1=n(),index2=mean(Age))
# A tibble: 3 x 3
  Pclass index1 index2
   <int>  <int>  <dbl>
1      1      4   46.2
2      2      3   NA
3      3     13   NA
> summarise(titanic2,index1=n(),index2=mean(Age,na.rm=T))
# A tibble: 3 x 3
  Pclass index1 index2
   <int>  <int>  <dbl>
1      1      4   46.2
2      2      3   34.5
3      3     13   20.2
```

对数据集加入一个分组标志

利用泛函对有分组的数据集进行计算

泛函即函数之函数，此处为调用其他函数进行计算的函数

出于简介易读的目的，管道运算符很实用，即%>%

```
> titanic%>%
+ group_by(Pclass)%>%
+ summarise(index1=n(),index2=mean(Age,na.rm = T))
# A tibble: 3 x 3
  Pclass index1 index2
   <int>  <int>  <dbl>
1      1      4   46.2
2      2      3   34.5
3      3     13   20.2
```

将左侧计算结果传导到下一个计算函数中作为其第一个参数

# 时间序列类型*

时间序列是经济管理中常见的数据类型，例如股票指数、销售量、采购量等按照时间排列的数据。

```
> u=rnorm(100,10,sd=5)
> sales=ts(u,start = 2000,frequency = 12)
> sales
```

时间序列变量本质上也是基于原子向量得到的，但建立起了数据与事件之间的联系，变量类型为ts

start参数设置起始时间，frequency设置时间模式，1代表年度数据，4代表季度数据，12为月份数据，52为周数据

对时间序列取子集的方式是使用窗口函数window，参数设置起始时间

```
> section<-window(sales,start=c(2001,1),end=c(2002,12))
```

plot函数(作图函数)会根据时间序列类型自动匹配它的画图模式

```
> plot(sales)
```