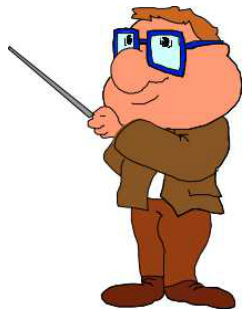
R 语言数据分析
Lecture 2 R 的数据导入

刘 寅

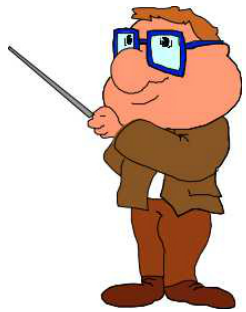
统计与数学学院

outline



1. 数据对象的类型
1.1 数据对象的属性(attributes)

- R中的任何对象都具有属性



1. 数据对象的类型
1.1 数据对象的属性(attributes)

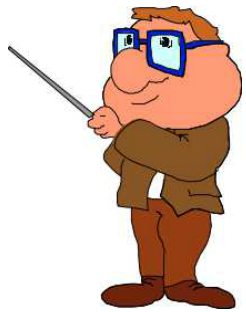
- R中的任何对象都具有属性
- 对象属性可以分为内在属性*(Intrinsic attributes)*和其他属性

1. 数据对象的类型
1.1 数据对象的属性(attributes)

- R中的任何对象都具有属性
- 对象属性可以分为内在属性*(Intrinsic attributes)*和其他属性
- 内在属性：模(mode)和长度(length)

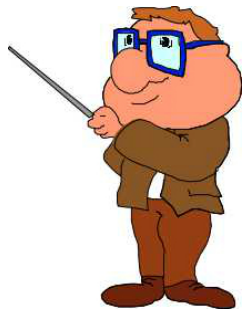
1. 数据对象的类型
1.1 数据对象的属性(attributes)

- R中的任何对象都具有属性
- 对象属性可以分为内在属性*(Intrinsic attributes)*和其他属性
- 内在属性：模(mode)和长度(length)
  - 模(mode):对象基本元素的类型，数据对象主要有4种：数值(numeric), 逻辑(logical), 复数(complex), 字符(character)



1. 数据对象的类型
1.1 数据对象的属性(attributes)

- R中的任何对象都具有属性
- 对象属性可以分为内在属性*(Intrinsic attributes)*和其他属性
- 内在属性：模(mode)和长度(length)
  - 模(mode):对象基本元素的类型，数据对象主要有4种：数值(numeric), 逻辑(logical), 复数(complex), 字符(character)
  - 长度(length):对象基本元素的个数



1. 数据对象的类型
1.1 数据对象的属性(attributes)

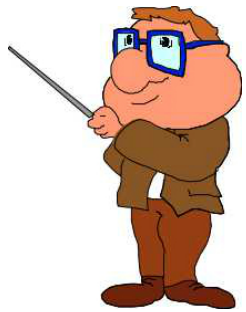
- R中的任何对象都具有属性
- 对象属性可以分为内在属性*(Intrinsic attributes)*和其他属性
- 内在属性：模(mode)和长度(length)
  - 模(mode):对象基本元素的类型，数据对象主要有4种：数值(numeric), 逻辑(logical), 复数(complex), 字符(character)
  - 长度(length):对象基本元素的个数
- 其他属性：维度(dim)、类(class)

1. 数据对象的类型
1.1 数据对象的属性(attributes)

- R中的任何对象都具有属性
- 对象属性可以分为内在属性*(Intrinsic attributes)*和其他属性
- 内在属性：模(mode)和长度(length)
  - 模(mode):对象基本元素的类型，数据对象主要有4种：数值(numeric), 逻辑(logical), 复数(complex), 字符(character)
  - 长度(length):对象基本元素的个数
- 其他属性：维度(dim)、类(class)
- 可以使用`attibutes`函数获取对象的一些属性值

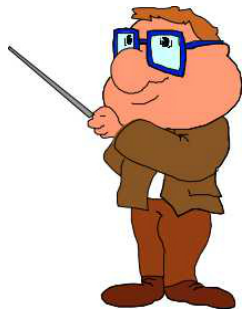
1.2 数据对象的类型(class)

- 向量(Vector)：有序数据



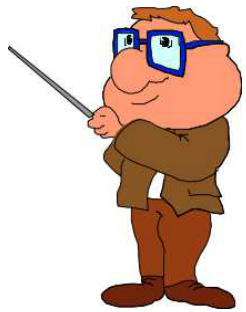
1.2 数据对象的类型(class)

- 向量(Vector)：有序数据
- 因子(Factor)：分类数据



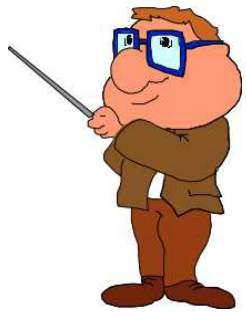
1.2 数据对象的类型(class)

- 向量(Vector)：有序数据
- 因子(Factor)：分类数据
- 时间序列(Time series)：按时间顺序排列的数据



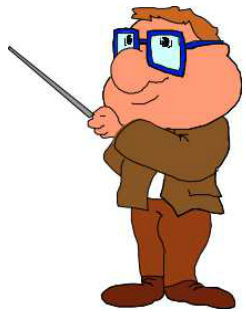
1.2 数据对象的类型(class)

- 向量(Vector)：有序数据
- 因子(Factor)：分类数据
- 时间序列(Time series)：按时间顺序排列的数据
- 矩阵(Matrix)：二维数组



1.2 数据对象的类型(class)

- 向量(Vector)：有序数据
- 因子(Factor)：分类数据
- 时间序列(Time series)：按时间顺序排列的数据
- 矩阵(Matrix)：二维数组
- 数组(Array)：多维有序数据

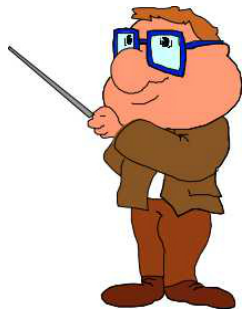


1.2 数据对象的类型(class)

- 向量(Vector)：有序数据
- 因子(Factor)：分类数据
- 时间序列(Time series)：按时间顺序排列的数据
- 矩阵(Matrix)：二维数组
- 数组(Array)：多维有序数据
- 列表(List)：可以包含任何类型对象

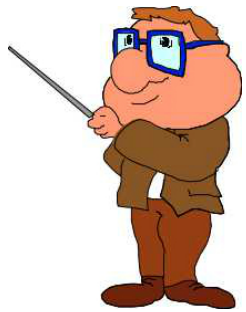
1.2 数据对象的类型(class)

- 向量(Vector)：有序数据
- 因子(Factor)：分类数据
- 时间序列(Time series)：按时间顺序排列的数据
- 矩阵(Matrix)：二维数组
- 数组(Array)：多维有序数据
- 列表(List)：可以包含任何类型对象
- 数据框(Data frame)：一种特殊的列表(各组成成分长度相等)



1.3 向量

- 向量是R中最为基本的类型
- 一个向量中的元素的类型必须**相同**，包括
  - 数值型
  - 逻辑型
  - 复值符
  - 字符型



♦ 数值型向量

- 定义数值型向量的常用方法
  - seq()或: 若向量或序列具有较为简单的规律。
  - rep() 若向量或序列具有较为复杂的规律。
  - c() 若向量或序列没有规律。

例:

> 1:10 $\Longrightarrow$ 1 2 3 4 5 6 7 8 9 10

> seq(1,5,by=0.5) $\Longrightarrow$ 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0

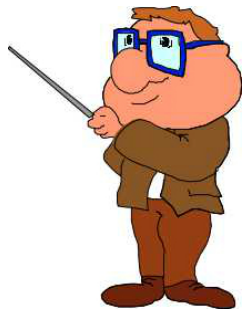
> seq(1,5,length=9) $\Longrightarrow$ 1.0 1.4 1.8 2.2 2.6 3.0 3.4 3.8 4.2 4.6 5.0

> rep(2:5,2) $\Longrightarrow$ 2 3 4 5 2 3 4 5

> rep(2:5,rep(2,4)) $\Longrightarrow$ 2 2 3 3 4 4 5 5

> x=c(24,10,16) $\Longrightarrow$ 24 10 **1**6

> length(x) $\Longrightarrow$ 3



- 向量运算+, −, ∗, /, ∧ 的含义是对向量中的每一个元素进行相应的运算。
- 两个等长的向量之间的+, −, ∗, /, ∧ 运算是对应元素间的四则运算。
- 两个不等长的向量之间的运算则是当长的向量的长度是短的向量的长度的整倍数时，长度短的向量将循环使用。

例：

> a<−seq(1,4,1) $\Longrightarrow$ 1 2 3 4

> b<−seq(5,8,1) $\Longrightarrow$ 5 6 7 8

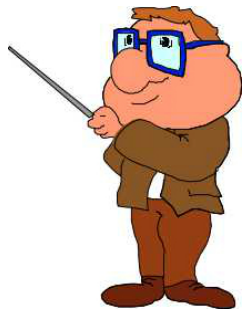
> c<−c(5,6) $\Longrightarrow$ 5 6

> 2∗a-1 $\Longrightarrow$ 1 3 5 7

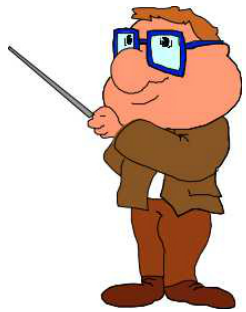
> a∧b $\Longrightarrow$ 1 64 2187 65536

> a+c $\Longrightarrow$ 6 8 8 10

> a∗c $\Longrightarrow$ 5 12 15 24

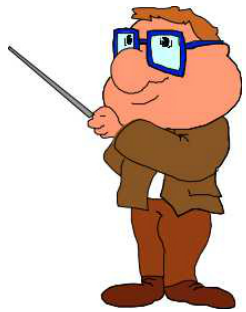


- sqrt()　开平方
- abs()　求绝对值
- sort(x)==sort(x, decreasing=FALSE)　返回按x 的元素从小到大排序的结果向量
- order(x)　返回x从小到大排列的元素在原向量中的位置序号
- sort(x)==x[order(x)]
- numeric(n)　表示长度为n的零向量
- **注意1:n-1与1:(n-1)的区别**



♦ 逻辑型向量

- 向量可以取逻辑值
  - x <– c(1.5, 4, 6)
  - l <– x>3
  - l $\Longrightarrow$ 返回 FALSE TRUE TRUE
- 两个向量也可以进行比较，常用的比较运算符有<, <=, >, >=, ==, ! =。
  log(10∗x)>x $\Longrightarrow$ 返回 TRUE FALSE FALSE
- 逻辑向量可以进行与(&)[表示同时满足]、或(| or ||)[表示两者之一]的运算。
  (x>2)&(x<5) $\Longrightarrow$ 返回 FALSE TRUE FALSE
- 判断一个逻辑向量是否全部为真值时的函数为all。
  all(log(10∗x)>x) $\Longrightarrow$ 返回FALSE
- 判断一个逻辑向量是有真值时的函数为any。
  any(log(10∗x)>x) $\Longrightarrow$ 返回TRUE
- 判断一个逻辑向量是否有缺失值时的函数为is.na。
  is.na(c(2,4,NA)) $\Longrightarrow$ 返回TRUE

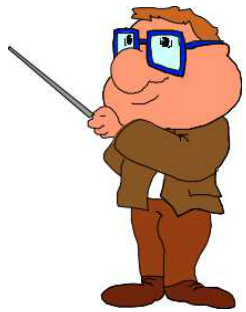


复值型向量

- 复值输入只需形如5+2i即可
- 利用complex(re, im, mod, arg)函数来定义复数：
  - re 定义实部
  - im 定义虚部
  - mod 定义复数模
  - arg 定义复数辐角
- Re(z)　计算实部
- Im(z)　计算虚部
- Mod(z)　计算复数模
- Arg(z)　计算复数辐角
- Conj(z)　计算复数共轭

♦ 字符型向量

- 字符用双引号引起来
  - c1 <– c(“a”, “b”)
  - c2 <– c(“weight”, “height”, “age”)
- paste函数用来把它的自变量连成字符串，中间用相应的分隔符
  - paste(“my”, “job”, sep=“”)#sep 表示所用的分隔符$\Longrightarrow$ 返回 “myjob’ ‘
  - paste(“my”, “job”, sep=“.”) $\Longrightarrow$ 返回 “my.job’ ‘



♦ 向量下标的运算

- R中向量的下标由1开始。访问向量的某个元素x[i]。
  - x <– c(14, 5, 7, 20)
  - x[2] $\Longrightarrow$ 返回 5
- 也可单独改变某个元素的值
  - x[3] <– 6, x $\Longrightarrow$ 返回 14 5 6 20
- 访问向量的一部分的几种方法：

(1) 正整数下标向量

x[c(1,4)] $\Longrightarrow$ 返回 14 20
x[2:4] $\Longrightarrow$ 返回 5 7 20

(2) 负整数下标向量

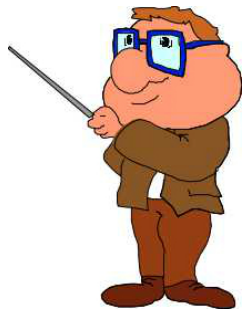
x[-1] $\Longrightarrow$ 返回 5 7 20 # 表示删除第一个元素

(3) 逻辑下标向量

x[x<10] $\Longrightarrow$ 返回 5 7

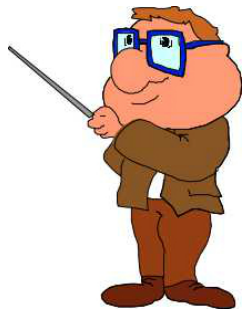
(4) 字符型下标向量

定义向量时可以给元素加上名字，访问的时候可以用前面的方法，也可以用元素名访问
ages <– c(Li=23, Zhang=33, Wang=45)
ages[“Zhang”] $\Longrightarrow$ 返回 Zhang 33



1.4 因子

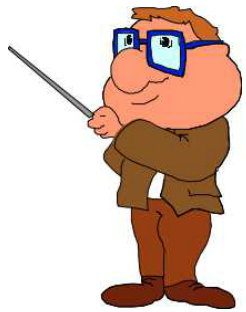
- R中的因子对象不仅包括分类变量的值，还包括所有分类变量所有可能的取值(levels).



1.4 因子

- R中的因子对象不仅包括分类变量的值，还包括所有分类变量所有可能的取值(levels).
- 可以用函数factor来创建一个因子对象

```
> a <- factor(1:3)
> b <- factor(1:3,levels=1:5)
> d <- factor(1:3,labels=c("A","B","C"))
```



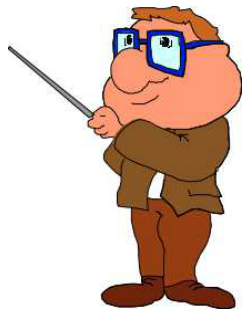
1.4 因子

- R中的因子对象不仅包括分类变量的值，还包括所有分类变量所有可能的取值(levels).
- 可以用函数factor来创建一个因子对象

```
> a <- factor(1:3)
> b <- factor(1:3,levels=1:5)
> d <- factor(1:3,labels=c("A","B","C"))
```

- 函数levels可以提取一个因子对象的水平

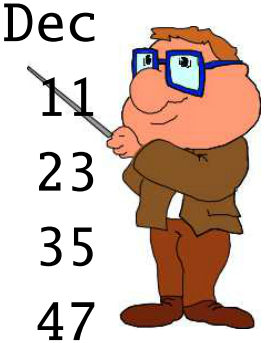
```
> b

[1] 1 2 3
Levels: 1 2 3 4 5

> d

[1] A B C
Levels: A B C
```



1.5 时间序列

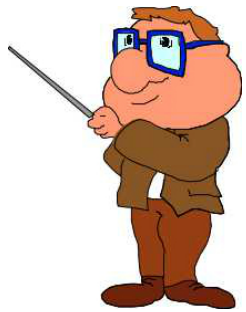
- 函数ts可以用来建立一个时间序列

```
> ts(1:10, frequency = 4, start = c(1959, 2))

     Qtr1 Qtr2 Qtr3 Qtr4
1959         1    2    3
1960    4    5    6    7
1961    8    9   10

> g <- ts(1:47, frequency = 12, start = c(1959, 2))
> print(g,calendar=T)

     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1959         1   2   3   4   5   6   7   8   9  10  11
1960  12  13  14  15  16  17  18  19  20  21  22  23
1961  24  25  26  27  28  29  30  31  32  33  34  35
1962  36  37  38  39  40  41  42  43  44  45  46  47
```



1.6 矩阵与多维数组

- 矩阵函数格式
  - matrix(data=NA, nrow=1, ncol=1, byrow=FALSE, dimnames=NULL)
    - data 为数据向量
    - nrow 为矩阵行数
    - ncol 为矩阵列数
    - byrow=TRUE 按行排列，否则按列排列(R中默认按列排)
    - dimnames 定义矩阵的行名和列名
  - rbind() 把向量按行排成矩阵
  - cbind() 把向量按列排成矩阵
- 多维数组函数格式
  - array(data=NA, dim=length(data), dimnames=NULL)
    - data为数据向量
    - dim为数组各维度的长度
    - dimnames定义数组各维的名称
- 访问矩阵元素和子矩阵
  - A[2,3] # 访问矩阵第2行第3列位置处的元素
  - A[i,] # 访问矩阵第i行的所有元素
  - A[,j] # 访问矩阵第j列的所有元素
  - rownames(A)=c(“a”, “b”, “c”) # 对矩阵的行进行命名
  - colnames(A)=c(“a”, “b”, “c”) # 对矩阵的列进行命名

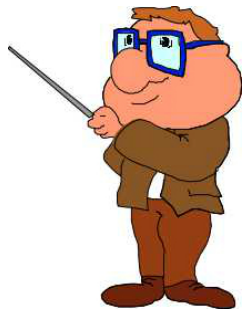


矩阵运算

- 矩阵可以进行四则运算(+, −, *, /, ∧)表示矩阵对应元素的四则运算，进行运算的一般是相同形状的矩阵。
- 形状不一致的矩阵和向量也可以进行四则运算，规则是矩阵的行数与向量的长度相等，将矩阵的数据按列与向量对应元素进行运算。
- A*B # 相同维数的矩阵对应元素相乘
- A%*%B # 矩阵乘法
- t(A) # 矩阵求转置
- diag(A) # 方阵取主对角线元素构成的向量
- diag(***a***) # 以向量***a***为主对角线上元素构成的对角矩阵
- det(A) # 方阵求行列式
- solve(A) # 方阵求逆矩阵
- eigen(A) # 方阵求特征值和标准化的特征向量

apply函数

- 对矩阵想按照行(或列)进行某种运算，可用apply函数。
- apply(X, margin, fun, ...)
- X 表示矩阵
- margin=1 表示按行计算，margin=2 表示按列计算
- fun 表示用来计算的函数，例如mean、var、sd.....

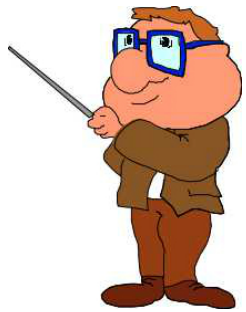


1.7 列表与数据框

- **列表(list):** list中各子对象的类型可以是任意对象，不同子对象不必是同一类型，并且不同子对象长度不必相等。此外，list还可以嵌套，只要定义第一层list的某个子对象为另一个list，就能继续包含不同类型、长度的对象了。
  - > result <– list(name="Fred", wife="Mary", no.children=3, child.ages=c(4,7,9))
  - 列表子对象引用格式：result[[下标]]=result[["子对象名"]]=result$子对象名
  - 注意：result[下标]则引用列表的一个子列表，结果类型仍为列表
- **数据框(data.frame):** 数据框是一种特殊的列表对象，各子对象必须是向量(数值型、字符型、逻辑型)、因子、数值型矩阵、列表或其它数据框。在数据框中以变量形式出现的向量结构必须长度一致，矩阵结构必须有一样的行数。
  - > result <– data.frame(name=c("Fred","Alice","Henry"), Sex=c("M", "F", "M"), child.ages=c(4,7,9), Height=c(96.5, 122.5, 136.8), Weight=c(18.6, 23.2, 33.0))
  - 数据框子对象引用格式：result[下标或下标向量]，result["子对象名"]=result$子对象名

2. 读、写数据文件
2.1 读取外部数据

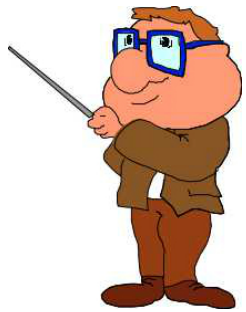
- 读纯文本文件：一个是read.table()函数，另一个是scan()函数：



2. 读、写数据文件
2.1 读取外部数据

- 读纯文本文件：一个是read.table()函数，另一个是scan()函数：
  - read.table()

```
> da <- read.table("...\\lecture 2\\d-ibm6298.txt", head=T)
```



2. 读、写数据文件
2.1 读取外部数据

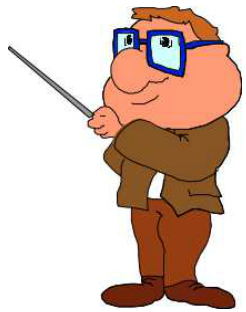
- 读纯文本文件：一个是read.table()函数，另一个是scan()函数：
  - read.table()

```
> da <- read.table("...\\lecture 2\\d-ibm6298.txt", head=T)
```

  - scan()

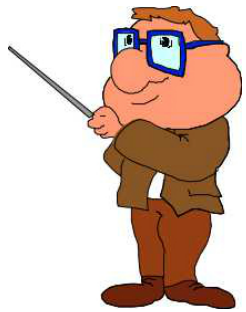
```
> scan("...\\lecture 2\\student.txt", what="c") # 以字符串的格式读取数据

Read 24 items
 [1] "学号"       "姓名"     "性别"     "年龄"
 [5] "2015026001" "王涛"     "男"       "22"
 [9] "2015026002" "蔡文龙"   "男"       "21"
[13] "2015026003" "汪美玲"   "女"       "21"
[17] "2015026004" "刘美云"   "女"       "22"
[21] "2015026005" "吕新亮"   "男"       "22"
```

```
> scan("...\\lecture 2\\student.txt", what="c", nlines=3)
> # 读取3行

Read 12 items
 [1] "学号"       "姓名"     "性别"     "年龄"
 [5] "2015026001" "王涛"     "男"       "22"
 [9] "2015026002" "蔡文龙"   "男"       "21"
```

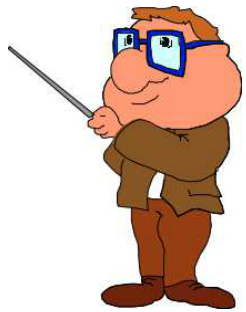


```
> scan("...\\lecture 2\\student.txt", what="c", nlines=3)
> # 读取3行

Read 12 items
 [1] "学号"       "姓名"     "性别"     "年龄"
 [5] "2015026001" "王涛"     "男"       "22"
 [9] "2015026002" "蔡文龙"   "男"       "21"

> scan("...\\lecture 2\\student.txt", what="c", skip=1)
> # 忽略第1行

Read 20 items
 [1] "2015026001" "王涛"     "男"       "22"
 [5] "2015026002" "蔡文龙"   "男"       "21"
 [9] "2015026003" "汪美玲"   "女"       "21"
[13] "2015026004" "刘美云"   "女"       "22"
[17] "2015026005" "吕新亮"   "男"       "22"
```

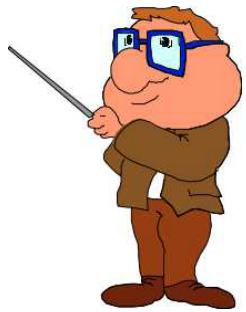
```
> scan("...\\lecture 2\\student.txt",
+ what=list(studentNo="",studentName="",
+ studentSex="",studentAge=0), skip=1)
> # 以列表形式读取数据

Read 5 records
$`studentNo`
[1] "2015026001" "2015026002" "2015026003" "2015026004"
[5] "2015026005"

$studentName
[1] "王涛"   "蔡文龙" "汪美玲" "刘美云" "吕新亮"

$studentSex
[1] "男" "男" "女" "女" "男"

$studentAge
[1] 22 21 21 22 22
```



- 读其他格式的数据文件



- 读其他格式的数据文件
  - 读SPSS，SAS，S-PLUS，Stata数据文件，首先调用“foreign”模块，library(foreign):
    - 读SPSS 文件的格式为read.spss(“XXX.sav”)；
    - 读SAS文件的格式为read.xport(“XXX.xpt”)；
    - 读S-PLUS文件的格式为read.S(“XXX”)；
    - 读Stata文件的格式为read.dta(“XXX.dta”)。

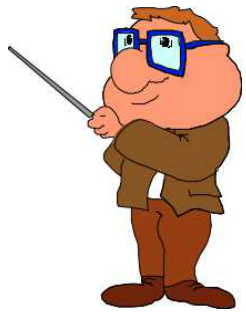


- 读Excel数据文件：直接读取xls格式的文件

```
> install.packages("gdata")
> library(gdata)
> read.xls("...数据.xls", sheet=1, fileEncoding="UTF-8",
>          sep=",")
```

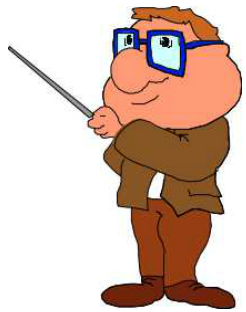
- 可能会提示报错。若报错，经查询，此错误是因为计算机中未安装Perl，下载Perl并安装(http://strawberryperl.com/)

```
>  DATA <- read.xls("...\\数据.xls", sheet=1,
>          fileEncoding="UTF-8",sep=",",
>          perl="C:\\Strawberry\\perl\\bin\\perl.exe")
```



- 读Excel数据文件，转化为“.csv”(逗号分隔)格式的文件进行读取，read.csv(“XXX.csv”)

```
> gnp <- read.csv("...\\lecture 2\\GNPC96.csv")
> gdp <- read.csv("...\\lecture 2\\GDP.csv")

> scan("...\\lecture 2\\GNPC96.csv", what="c", sep=",")
> # 指定逗号作为分隔符
```



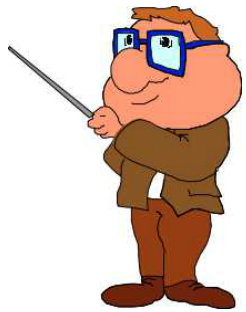
2.2 写数据文件

- write()函数， write(XX, file=“data”)；

2.2 写数据文件

- write()函数， write(XX, file=“data”)；
  - write.table() 函数写纯文本格式的数据

```
> write.table(gnp, file="...\\lecture 2\\GNP.txt")
```

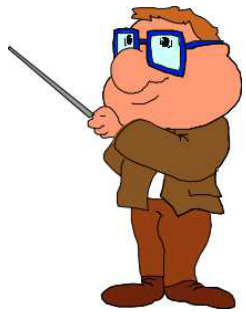


2.2 写数据文件

- write()函数， write(XX, file=“data”)；
  - write.table() 函数写纯文本格式的数据

    ```
    > write.table(gnp, file="...\\lecture 2\\GNP.txt")
    ```
  - write.csv() 函数写Excel数据文件

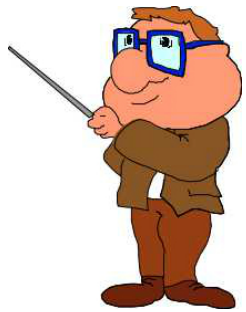
    ```
    > write.csv(gnp, file="...\\lecture 2\\GNP.csv")
    ```



3. 显示数据
3.1 显示数据类型

- 显示数据的类型：使用str函数

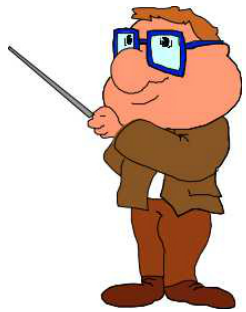
```
> str(gnp)

'data.frame':   283 obs. of  2 variables:
 $ DATE  : Factor 283 levels "1947-01-01","1947-04-01",..
 $ GNPC96: num  1947 1945 1943 1974 2004 ...
```



3.2 显示或修改数据

- 打出对象名称

```
> gnp
```



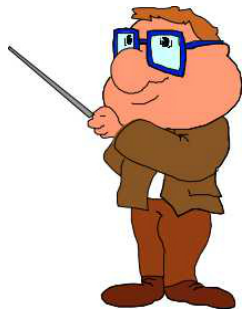
3.2 显示或修改数据

- 打出对象名称

```
> gnp
```

- 显示最开始几个数据：head

```
> head(gnp)
```



3.2 显示或修改数据

- 打出对象名称

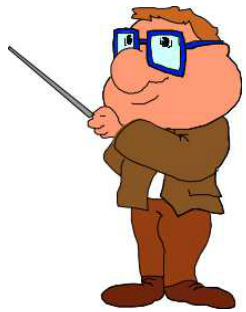
```
> gnp
```

- 显示最开始几个数据：head

```
> head(gnp)
```

- 显示最后几个数据:tail

```
> tail(gnp)
```



3.2 显示或修改数据

- 打出对象名称

```
> gnp
```

- 显示最开始几个数据：head

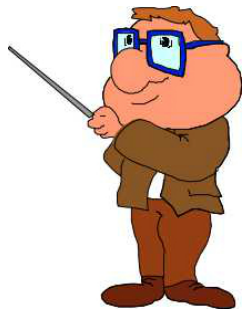
```
> head(gnp)
```

- 显示最后几个数据:tail

```
> tail(gnp)
```

- 调用文本编辑器修改R对象:edit

```
> edit(gnp)
```



3.2 显示或修改数据

- 打出对象名称

```
> gnp
```

- 显示最开始几个数据：head

```
> head(gnp)
```

- 显示最后几个数据:tail

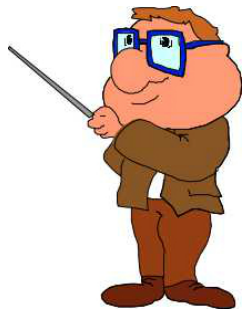
```
> tail(gnp)
```

- 调用文本编辑器修改R对象:edit

```
> edit(gnp)
```

- 调用edit以表格形式修改R对象:fix

```
> fix(gnp)
```



3.2 显示或修改数据

- 打出对象名称

  ```
  > gnp
  ```

- 显示最开始几个数据：head

  ```
  > head(gnp)
  ```

- 显示最后几个数据:tail

  ```
  > tail(gnp)
  ```

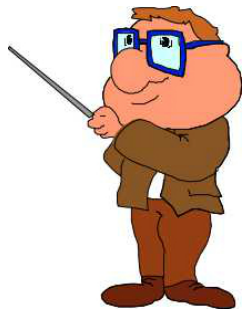
- 调用文本编辑器修改R对象:edit

  ```
  > edit(gnp)
  ```

- 调用edit以表格形式修改R对象:fix

  ```
  > fix(gnp)
  ```

- 调用表格形式修改R对象:data.entry
  仅对数值型或字符型有效



第二章结束了！

THANKS