数据分析与处理技术——2 基础数据结构

商学院　徐宁

参考书籍

- 《R语言实战》第二章
- 《R语言–实用数据分析与可视化技术》第四章
- 《R语言教程》(在线版) 3.-8.章
  https://www.math.pku.edu.cn/teachers/lidf/docs/Rbook/html/_Rbook/prog-type-intro.html

基础数据结构

1.数据模式

数值型数据

逻辑型数据

复数型数据

字符型数据

数值型数据

R中的数字可以分为整数型和双精度浮点型（即存在小数）

但变量会自动适应两种类型，等效混合为数值型

R Console

```
> a=2.5+3.7
> a
[1] 6.2
> typeof(a)
[1]“double"
```

```
> b=7L
> typeof(b)
[1] "integer"
```

提示：typeof()是用来测数据类型的函数，回忆pi是什么类型。

精度表示

- 数据的表现形式

- 科学计数法

```
> pi
[1] 3.141593
> options(digits = 10)
> pi
[1] 3.141592654

> 25^10
[1] 9.5367e+13
> 0.25^10
[1] 9.5367e-07
```

提示：options()是设置R整体环境的函数，可以调控包括digits在内的许多公共参数

特殊数据

- 一些特殊数字符号需要牢记：

圆周率	pi
空值	NA
空缺值	NaN
无/删除	NULL
无穷大	Inf

注意：大小写敏感

```
> 1+NA
[1] NA
```

问题：为什么计算结果是NA

```
> cos(pi)
[1] -1
> sin(pi)
[1] 1.224647e-16
```

问题：为什么sin(pi)结果不是0

复数型数据

复数的形式：a+bi
复数的类型：“complex”

```
> x=2+3i
> x^2
[1] -5+12i
> log(x)
[1] 1.2825+0.9828i
```

Re(x)	取实部
Im(x)	取虚部
Mod(x)	取模
Arg(x)	求相位
Conj(x)	求共轭

注意：复数在我们专业本科阶段几乎涉及不到

逻辑型数据

- 逻辑型（logical)数据只有两个取值：TRUE 和 FALSE
- 主要用于逻辑判断

```
> T
[1] TRUE
> 5>=4
[1] TRUE
```

```
> a=c(1,2,3,5,7)
> a>2
[1] FALSE FALSE  TRUE  TRUE  TRUE
```

提示：TRUE可以简写为T,FALSE可以简写为F

逻辑运算

- 比较运算符： > < >= <= == %in%
- 逻辑运算符 ： & | !
- any()函数，是否存在条件为真
- all()函数，是否全部为真

```
> a
[1] 1 2 3 5 7
> 3 %in% a
[1] TRUE
> (5>=4)&(3 %in% a)
[1] TRUE
> all(a>2)
[1] FALSE
> any(a>2)
[1] TRUE
```

提示：a是如何赋值的，转到15页看看

字符型数据

字符型数据：
-括在引号之内
-存在转义符

字符编码表：
-ASCII
-Unicode-8

```
> a='wang qi'
> b=c('zhang ke','wang qi','li shan')
> a %in% b
[1] TRUE
```

字符操作

- 打印字符
- 常用转义符
  - 替代特殊字符\\ \’
  - 换行符 \n

```
> print(y)
[1] "my"     "name"   "is"
"wangqi"
> cat(y)
my name is wangqi

>setwd(“d:\\Documents”)
```

字符拼接

- 粘贴两个字符串
- 粘贴字符向量

```
> paste('monkey','king')
[1] "monkey king"
```

```
> y
[1] "my"      "name"    "is"      “wanggi"
> paste(y,collapse = ' ')
[1] "my name is wanggi"
```

数据模式

基础数据类型包括四种：

- 数值型
- 字符型
- 逻辑型
- 复数型

此外，R中还有许多其他数据类型，但都是利用这四种基础类型构建起来的，只有这四种被称为mode

```
> a=2.75
> b="wang qi"
> c=TRUE
> d=3+4i
```

mode()函数

```
“numeric"
“character"
“logical"
"complex"
```

注意：numeric类型混合了integer和double两类

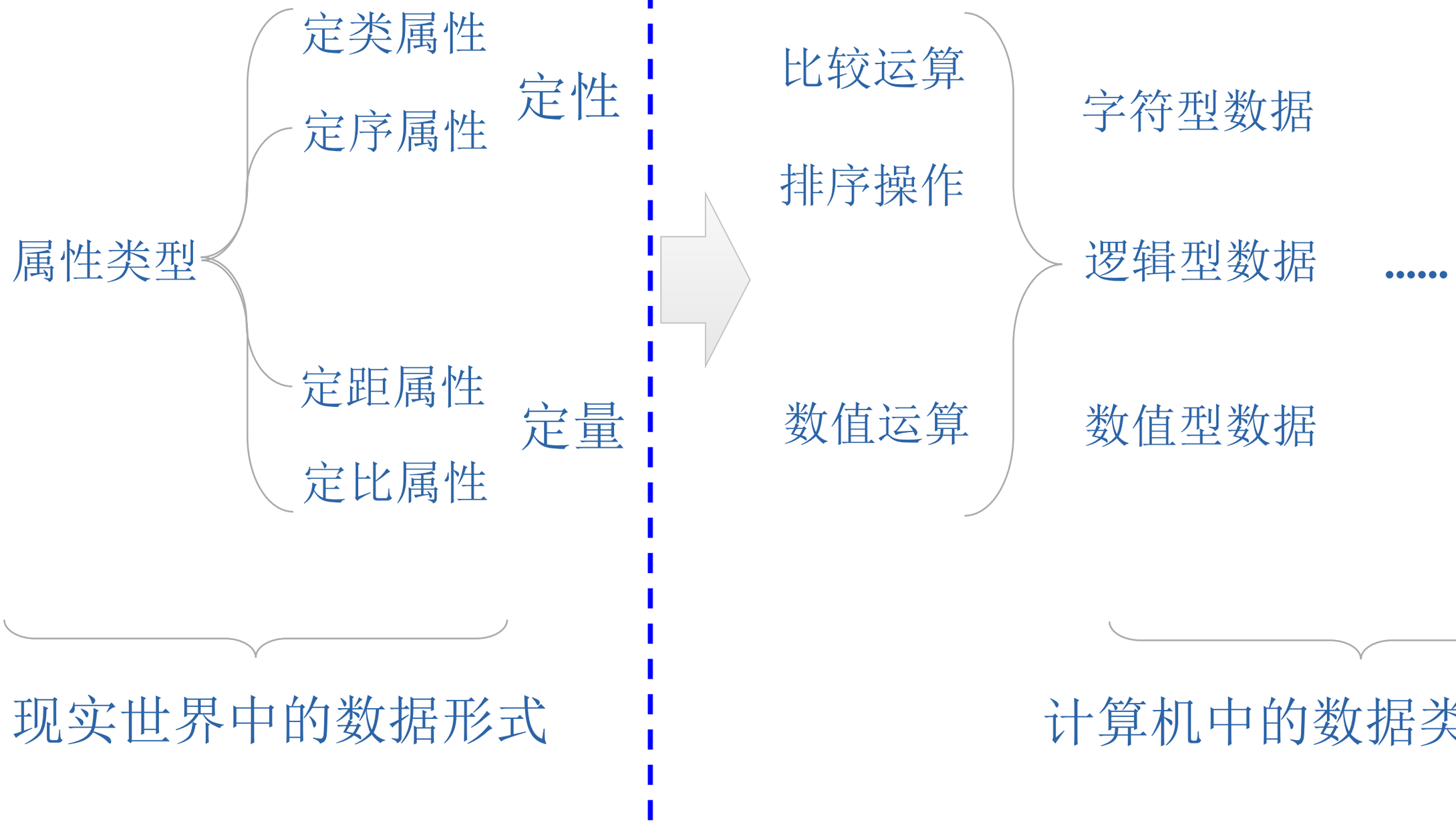
现实世界的数据分类

- 定类属性：标明类别
- 定序属性：标明顺序
- 定距属性：测量差距
- 定比属性：测量比率

姓名	性别	英语成绩	数学成绩	身高
李雷	M	A	92	175
韩梅梅	F	A	85	170
王萌	F	B	95	162
张小明	M	B	82	172
王小二	M	C	70	179

问题：身份证号、邮政编码是什么类型属性？

数据的信息化存储


属性类型
定类属性
定序属性
定性
定距属性
定比属性
定量
比较运算
排序操作
数值运算
字符型数据
逻辑型数据
.....
数值型数据
现实世界中的数据形式
计算机中的数据类型

基础数据结构

2.原子向量

向量基本操作

向量化运算

元素与索引

原子向量

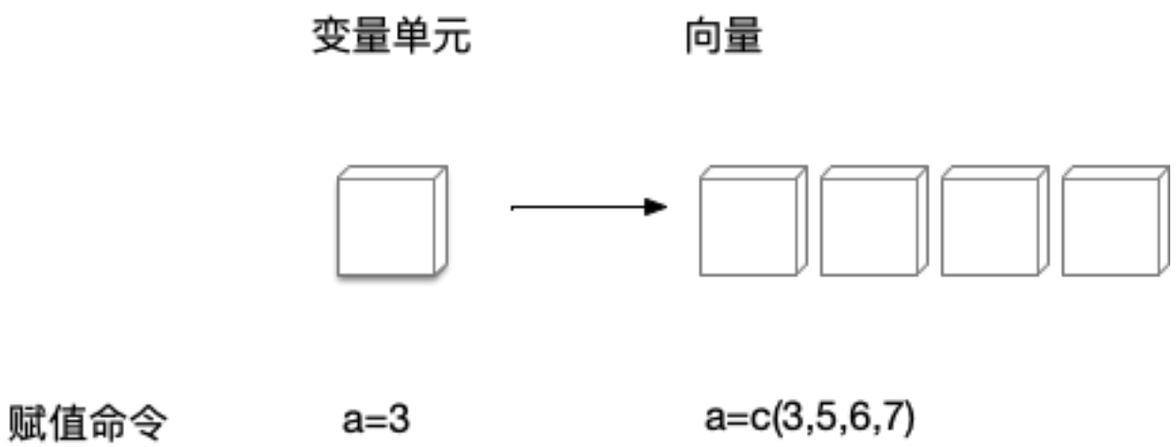
变量用来将数据存储在计算机中备用

原子向量：

- 元素仅有前后顺序
- 单模式，即元素的数据类型保持一致

数据

变量

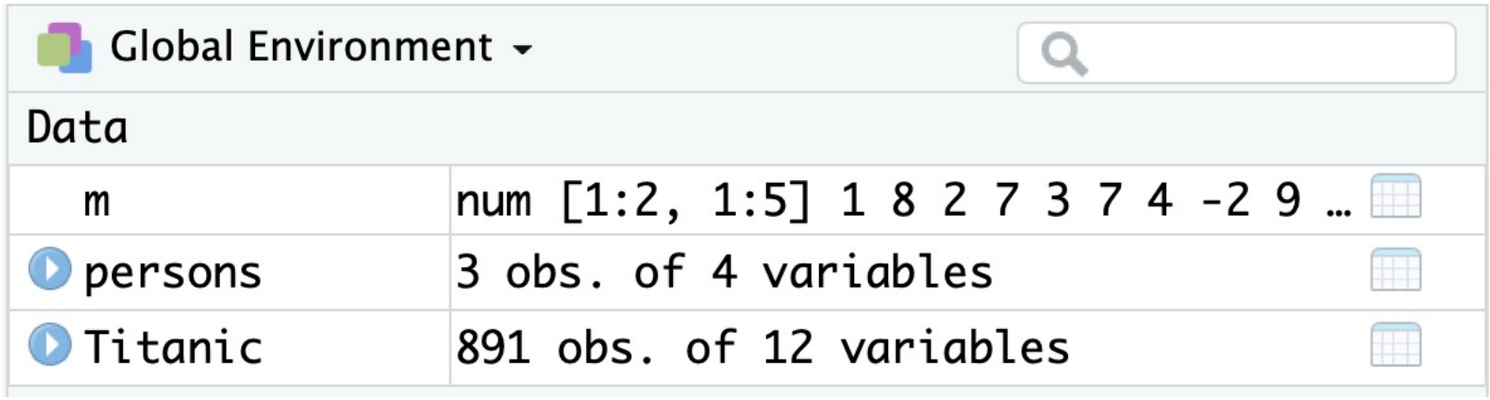


注意：原子向量不是线性代数上的向量概念

向量属性

变量属性携带了数据的许多信息

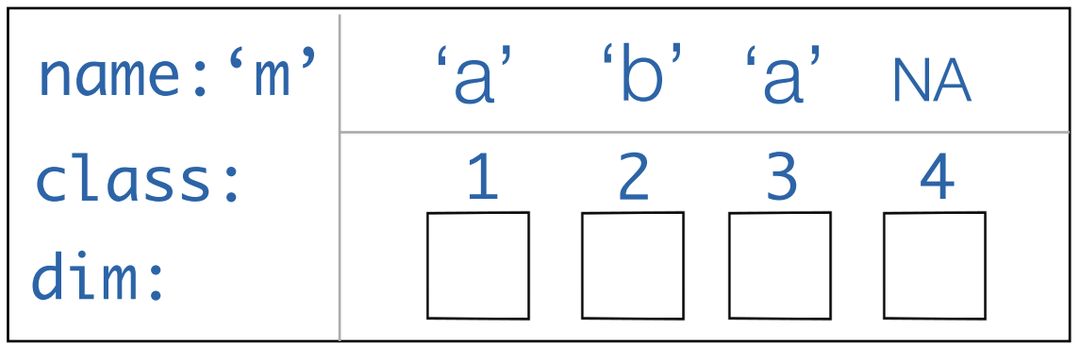
1. 变量类型
2. 维度属性
3. 元素名称



常用的属性操作函数：

```
names()
class()
mode()
dim()
```

变量结构的逻辑图



创建向量

- 使用c()函数创建原子向量

- 冒号：创建连续向量

```
> a <- c(1,2,3,5,7)
> a
[1] 1 2 3 5 7
> a <- c(a,c(-1,-10),9)
> a
[1]   1   2   3   5   7  -1
-10   9
> b <- 2:5
> b
[1] 2 3 4 5
```

创建向量

创建连续型向量：

- seq()函数

创建重复元素的向量：

- rep()函数

```
> seq(from=1,to=10,by=2)
[1] 1 3 5 7 9

> seq(1,10,2)
[1] 1 3 5 7 9
```

```
> rep(c(1,3),3)
[1] 1 3 1 3 1 3
```

向量元素名的操作

- 元素名称可以通过names()函数替代方式修改；
- 当元素有名称时，索引可以按名称取数据，如euro[‘BEF’]

```
> euro
      ATS       BEF       DEM       ESP
13.760300 40.339900  1.955830 166.386000
      FIM       FRF       IEP       ITL
 5.945730  6.559570  0.787564 1936.270000
      LUF       NLG       PTE
40.339900  2.203710 200.482000

> names(euro)
 [1] "ATS" "BEF" "DEM" "ESP" "FIM"
"FRF" "IEP" "ITL" "LUF" "NLG" "PTE"

> y=1:5
> names(y)=c('a','b','c','d','e')
> y
a b c d e
1 2 3 4 5
```

如上，names( )函数既可以调取元素名属性，也可以直接修改元素名

向量的修改操作

- 向量可以利用覆盖式赋值直接修改已有元素
- c()函数可以无限嵌套组合数据，从而实现在头部或尾部添加数据
- append()指定添加

```
> a <- c(9,4,-5,7,10)
> a <- c(a,3)
> a
[1]  9  4 -5  7 10  3

> c(a,4)
[1]  9  4 -5  7 10  3  4

> append(a,c(1,2),after=3)
[1]  9  4 -5  1  2  7 10  3
```

向量化运算

任何作用于向量的运算都等同于对每个元素的运算或操作

即向量名代表其所包含的每一个元素

```
> a <- c(1,2,3,5,7,-1,-10,9)
> sin(a)
[1]  0.84147  0.90930  0.14112
-0.95892  0.65699 -0.84147
0.54402  0.41212
```

注意：在没有特殊声明情况下，向量通常指原子向量，而非矩阵意义上的向量

分支计算

- ifelse()分支计算函数，向量化的分支判断，根据条件选择性执行计算任务

案例：10个学生的成绩进行判断

```
> a=c(72,81,54,80,92,68,85,88,76,45)
> ifelse(a>=60,'yes','no')
 [1] "yes" "yes" "no"  "yes" "yes" "yes"
"yes" "yes" "yes" "no"
```

进一步，ifelse的分支计算可以嵌套

```
> ifelse(a>=80,'>80', ifelse(a>=60&a<80,'60--
80','<60'))
[1] "60--80" ">80"    "<60"    ">80"    ">80"
[6] "60--80" ">80"    ">80"    "60--80" "<60"
```

向量的索引

- 向量索引指示元素位置
- 索引支持逆向调取元素
- 元素删除

```
> a <- c(1,2,3,5,7,-1,-10,9)
> a[4]
[1] 5
> a[c(1,3,5)]
[1] 1 3 7
> a[1:5]
[1] 1 2 3 5 7
> a[-4]
[1]   1   2   3   7  -1 -10   9
> a<-a[-4]
```

问题：a[-4]表达什么意思？

索引的逻辑性

- 索引的逻辑性

- 利用索引逻辑性可以实现数据筛选

```
> a <- 2:5
> a[c(T,F,T,F)]
[1] 2 4

> a[a>3]
[1] 4 5
```

索引的逻辑操作

索引具有逻辑性，根据判断运算返回的逻辑结果访问数据

同样原理，矩阵索引如何使用呢？

```
> y=c(1:4,9:7,c(7,-7,23))
> y
[1]  1  2  3  4  9  8  7  7 -7 23
> y[2:5]
[1] 2 3 4 9
> y[-(2:5)]
[1]  1  8  7  7 -7 23
> y>5
[1] FALSE FALSE FALSE FALSE  TRUE
TRUE  TRUE TRUE FALSE  TRUE
> y[y>5]
[1]  9  8  7  7 23
> y[!y>5]
[1]  1  2  3  4 -7
```

元素的定位

- which()函数获取满足条件的元素索引
- 即通过向量逻辑运算挑选数据元素
- which.max()
- which.min()

```
> y
[1]  1  2  3  4  9  8  7  7
-7 23
> which(y>5)
[1]  5  6  7  8 10
> y[which(y>5)]
[1]  9  8  7  7 23
```

元素排序

- sort( ) 直接对元素排序
- order( ) 获取元素的秩
- sort与order都有一个逆序参数decreasing默认是关闭状态
- rev() 向量元素倒置

```
> a <- c(9,4,-5,7,10)

> sort(a)
[1] -5  3  7  9 10
> order(a)
[1] 2 5 3 1 4

> sort(a,decreasing = T)
[1] 10  9  7  3 -5

> rev(a)
[1] 10  7 -5  4  9
```

练习：请用代码取出序列中第3小的数据

取子集

- 依靠索引逻辑取子集
- subset( )函数，依靠元素判断条件取子集

```
> a <- c(1:4,9:7,c(7,-2,23),rep(1:2,3))

> which(a<2)
[1]  1  9 11 13 15

> a[a<2]
[1]  1 -2  1  1  1

> subset(a,a<7 & a>2)
[1] 3 4
```

3.分类数据

无序分类数据

有序分类数据

连续分类数据

因子类型

- 无序型分类数据

```
> gender=factor(c('M','F','M','F'))
> gender
[1] M F M F
Levels: F M
```

- 有序型分类数据

```
> score=factor(c('A','B','A','C’),
levels = c('A','B','C'),ordered = T)
> score
[1] A B A C
Levels: A < B < C
```

因子类型：连续分类数据

- 数值数据转为类别数据

cut()函数对数值型数据切分区间段，转化为类别数据。

```
> ages=c(15,27,38,20,61,53,8,21,42)
> f_ages=cut(ages,breaks = c(0,16,29,50,Inf))
> f_ages
[1] (0,16]   (16,29]  (29,50]  (16,29]  (50,Inf]
(50,Inf] (0,16]   (16,29]  (29,50]
Levels: (0,16] (16,29] (29,50] (50,Inf]
```

cut()得到的结果是因子类型，原始数据对应为如上的因子标签。

注意breaks参数的不同带来的区间划分差异

4.时间型数据

日期/时间数据

操作时间数据

时间数据运算

时间型数据(日期/时间)

时间数据建立在字符型数据基础上

首先加载工具library(lubridate)

读取系统日期：

```
> today()
[1] "2019-04-10"
```

读取系统时间：

```
> now()
[1] "2019-04-10 21:56:28 CST"
```

时间型数据分为以日为单位计量的date类型和以秒为单位计量的POSIXct类型.

时间型数据(日期/时间)

- 时间数据存储了相对于参照时刻(1970年1月1日)以来的天数或秒数

时区：UTC 世界协调时间(经度0)
CST 中央标准时间(在中国即北京时间）
GMT 格林尼治时间

tz或tzone是设置时间时的常用时区参数

```
now(tz = 'Etc/GMT+8’)
now(tz = 'asia/shanghai')

OlsonNames()    #查看时区命名方式
```

创建时间数据

lubridate工具包将时间型数据操作大幅简化，字符格式能够接受非常随意的方式

计算机存储的是时间戳，以字符格式显示在屏幕上.
对于POSIXct数据，时间戳是一个以秒为单位的正整数。

```
> ymd('20190402')
[1] "2019-04-02"


> ymd_hms('20190902,23 50 59')
[1] "2019-09-02 23:50:59 UTC"

> a <- ymd_hms("2019-01-01 10:25:30")
> b <- ymd_hms("2019-06-15 10:25:30")

> mode(a)
[1] “numeric”

> y=seq(a,b,by=‘week’) #创建连续时间数据序列
```

获取时间数据的成分

时间数据的成分函数

- year()
- month()
- day()

既可以作为读取函数也可以作为替代函数

- ydays()一年中的第几天
- mdays()
- wdays()一周中的第几天

```
> year(a)
[1] 2019
> year(a) <- 2018
> d
[1] "2018-01-01 10:25:30 UTC"

> yday(a)
[1] 1
> yday(b)
[1] 166
> wdays(today())
> quarter(b) #第几个季度
```

时间数据运算

时间数据可以正常参与算术运算。

时段数据：

- 生成一周的时间段为：weeks(1)

  years()
  months()
  weeks()
  days()

```
> today()+1

> greatday <- ymd('1949-10-01')
> today()-greatday
Time difference of 25560 days

> greatday+years(70)
[1] "2019-10-01"
```

```
> today()+weeks(1) #当前一周后时间
```

```
> today()+months(2) #两个月后的时间
```

练习1

1．创建向量记录如下数据，其中NA代表空缺数据

5，19，3，5，19，3，NA，9，8，7，6，72，-1

（1）利用which函数对空缺数据进行定位，空缺数据替补为0；

（2）为每一个元素命名，依次为小写字母a,b,c,d……，其中命名为k的元素是多少；

（2）将其中大于等于7的元素提取出来，并将剩余元素存入另一个变量；

（3）向量中能够被3整除的元素是否超过5个？

练习2

1.利用随机函数生成100个期望为0方差为10的随机数，测试其中是否存在0，生成随机数的函数如下

```
> x<-rnorm(100,mean=0,sd=10)
```

（1）检验其中是否存在0元素

（2）探索这100个数字中是否存在不大于30的数字，若存在请核算一共有多少个，并将不大于30的数字所在位置显示出来。

练习3

1.右侧列出了常见的运算符，请探索在R中的运算符优先级。例如 1+2*3 运算中*优先级高于+

+	-	*	/
^	%%		
>	<	>=	<=
==	!=	%in%	
&	|	!	
=	<-	:	()

2.将一组数据存入变量中：1，3，-10，12，-7.5，-3，65，19，-45，17.2，0，22.72。

要求：所有数字均舍弃小数部分，其中非负数求自然指数运算，而负数则按照 $f(x)=x^3-3x^2+1$ 进行运算。

练习4

1.存储学生成绩信息：英语、数学，其中英语成绩分A优秀 B普通 C较差 三类，数学成绩为数值。

英语：A,C,B,C,A,B,B,C

数学：92，88，75，45，72，91，82，62

（1）根据数据类型分别创建两个向量存储以上数据；

（2）将数学成绩转化为优秀（超过90分）、良好（60到90之间）和较差（低于60分）三类

练习5

1.创建今天（当前日期）到2021年1月1日的日期时间型序列，问题：（1）这期间一共有多少日？（2）其中有多少个星期五？

2.作为路上丝绸之路重要一环，中欧铁路运输成为海运的补充方式之一，设郑州出发的列车时间为2018年8月9日晚22：11，到达德国汉堡时间是当地时间2018年8月25日10:20，计算列车全程运行时间多长。