

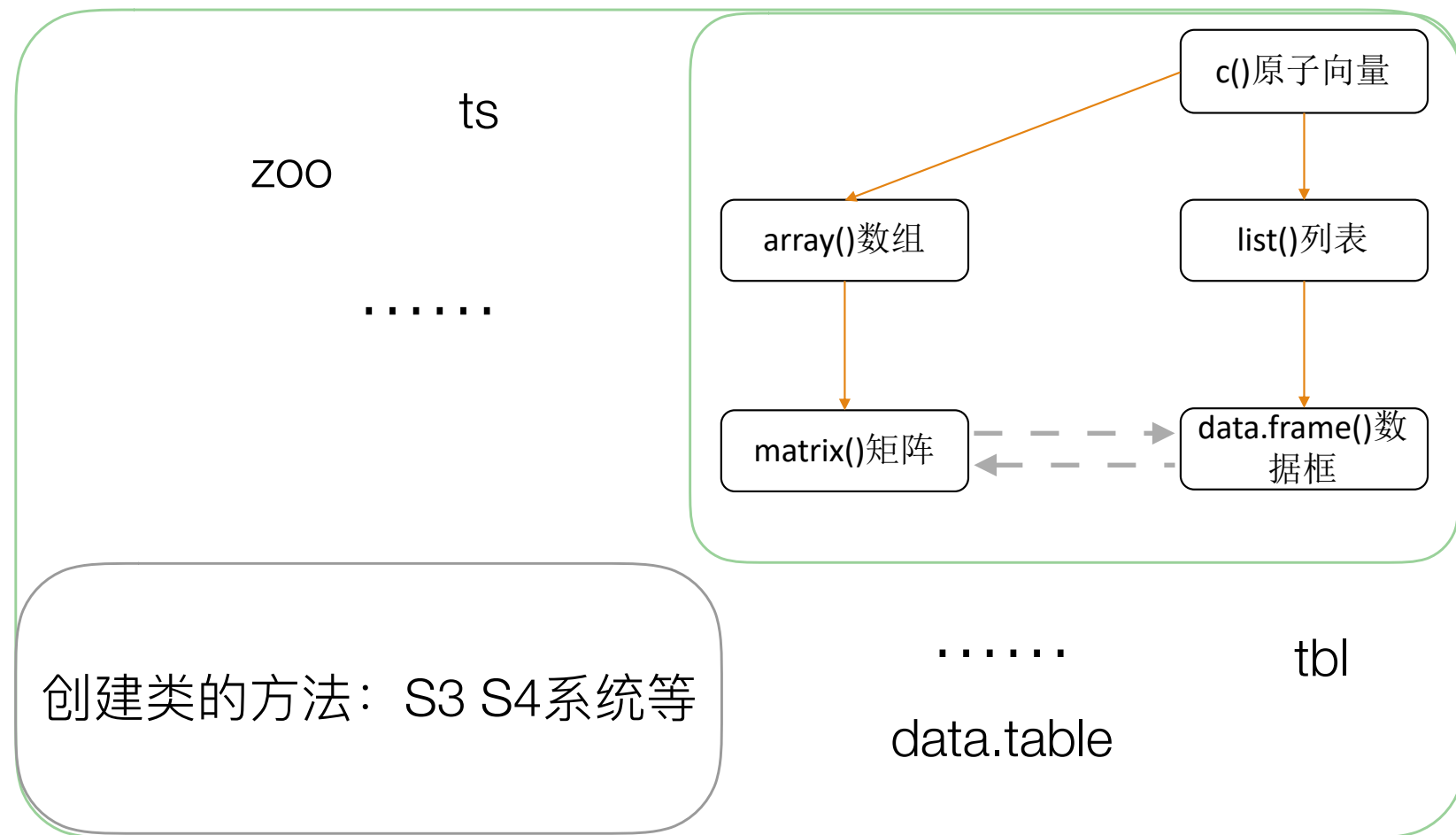
数据分析与处理技术

数据操作基础

南京审计大学商学院物流管理系

变量类型的拓展

原子向量是R中存储数据的基础类型，最数据的操作方法多数也具有继承关系。



查询变量类型class()

操作向量

在理解原子向量基础上，需要掌握灵活使用索引、基础函数等工具操作数据。这些操作很自然的被拓展到矩阵、列表、数据框等变量的操作当中。

规则向量生成

```
> 1:10
[1] 1 2 3 4 5 6 7 8 9 10
> seq(0,1,by=0.2)
[1] 0.0 0.2 0.4 0.6 0.8 1.0
> rep(1:3,3)
[1] 1 2 3 1 2 3 1 2 3
```

作为生成向量的基础工具，
rep()函数：重复生成
seq()函数：等间隔生成
其参数的调节参见课本P17-18

生成方式的混合非常方便

```
> c(1:4,9:7,c(7,-2,23))
[1] 1 2 3 4 9 8 7 7 -2 23
> c(1:4,rep(c(3,6),3))
[1] 1 2 3 4 3 6 3 6 3 6
```

数据修改

修改向量中的数据必须经过赋值过程写入内存，原理上所有修改都是生成新数据

c(2,3,5,9)
 ↑
 c()

c()函数可以无限嵌套
而不会产生层次

```
> b=c(1,2,5)  
> b=c(b,7)  
> b  
[1] 1 2 5 7
```

若指定添加位置，则需使用append添加

```
> b=append(b,10:15,after=2)
```

数据操作

参考：课本P19表2-2列出了常用向量操作函数。

向量的一些基础操作函数

查询数据类型mode()

测向量长度length()

取最大值max()

取最小值min()

求平均值mean()

求和函数sum()

累加函数cumsum()



拓展到矩阵的变形

ncol列

nrow行

ncol()

nrow()

ncolMean()

rowMean()

ncolSum()

rowSum()

元素定位

案例数据

```
> a<-c(1:4,9:7,c(7,-2,23),rep(1:2,3))  
> a  
[1] 1 2 3 4 9 8 7 7 -2 23 1 2 1 2 1 2
```

调取符合条件的元素本身相对容易，
但如何判定符合条件元素的位置？

寻找a中小于2的数据在第几个元素中

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

原理在于 $a < 2$ 产生的向量化运算逻辑结果

基于此，判别条件可以更为复杂

```
> which(a^0.7+2>5)  
[1] 5 6 7 8 10
```

定位的重要用途在于配合索引调取数据

```
> a[which(a^0.7+2>5)]  
[1] 9 8 7 7 23
```

工具：
which()
which.max()
which.min()

思考：如何定位
变量中的空缺值？

元素排序

案例数据 `> x<-c(1:3,9:5,3:7)`
`> x`

`[1] 1 2 3 9 8 7 6 5 3 4 5 6 7`

为了理解排序，重新回到原子向量

对x使用sort，直接产生排序好的新向量，

`> sort(x)`

`[1] 1 2 3 3 4 5 5 6 6 7 7 8 9`



`> x[order(x)]`

`[1] 1 2 3 3 4 5 5 6 6 7 7 8 9`

如果希望按降序则打开参数decreasing

`> sort(x,decreasing = T)`

`[1] 9 8 7 7 6 6 5 5 4 3 3 2 1`

除sort外，另一种排序并不关心大小，而仅是按顺序倒置：

`> rev(x)`

`[1] 7 6 5 4 3 5 6 7 8 9 3 2 1`

另外，许多排序问题需要的是元素序号而非排序结果：

```
> order(x)
[1] 1 2 3 9 10 8 11 7 12 6 13 5 4
```

观察与sort函数结果的差别

```
> sort(x)
[1] 1 2 3 3 4 5 5 6 6 7 7 8 9
```



```
> x[order(x)]
[1] 1 2 3 3 4 5 5 6 6 7 7 8 9
```

这种操作便与配合索引调用我们需要的内容

```
> x[order(x)[1:5]]
[1] 1 2 3 3 4
```

另一种与顺序相关的rank函数返回的是元素排序结果，也称为秩

```
> rank(x)
[1] 1.0 2.0 3.5 13.0 12.0 10.5 8.5 6.5 3.5 5.0 6.5 8.5 10.5
```


元素名称

变量对数据的标识方式影响我们如何使用索引调取。

基础的标识方式是按顺序编号，如右侧向量y

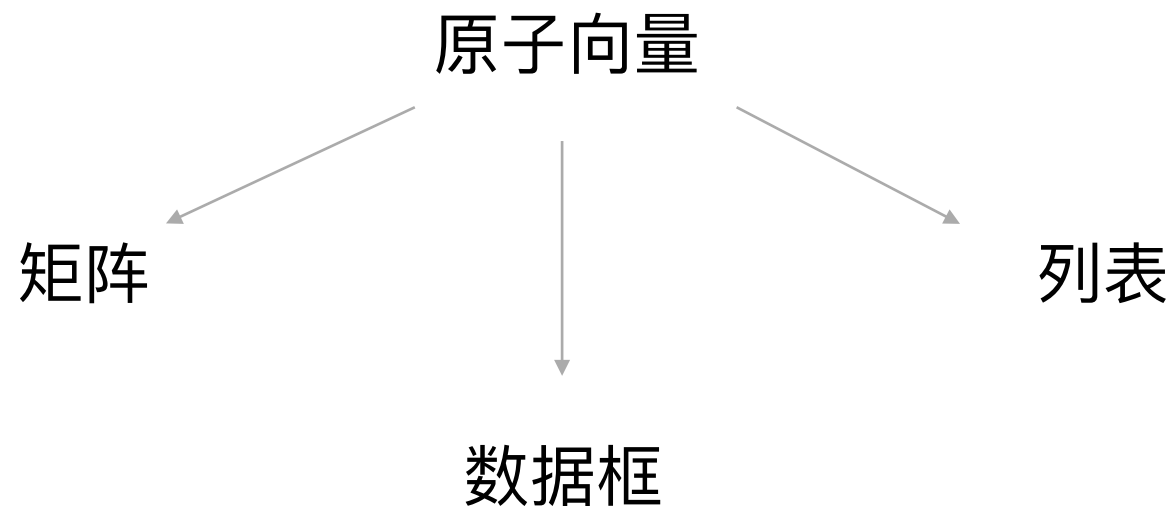
使用names()函数为元素创建名称

names()是另一种特殊函数，称为替换函数。它可以帮你取出元素名称，也可以作为被赋值对象直接将内容写入变量存储中。

```
> y
[1] 1 2 3 4 5 6 7 8 9 10 8
> names(y)<-letters[1:11]
> y
 a  b  c  d  e  f  g  h  i  j  k
 1  2  3  4  5  6  7  8  9 10  8
> y['c']
c
3

> y[3]
c
3
```

思考：如何取出y的第二个元素名？如何将其修改？



列名称: `row.names()`
行名称: `colnames()`

```
> m
      南京 上海
南京    0  300
上海  500    0
> row.names(m)
[1] "南京" "上海"
```

列表变量下的元素名称

```
> t<-list(first=x,second=letters[1:4])

> names(t)
[1] "first" "second"
```

列表中变量间层级关系清晰, 因此原子向量中的方法几乎都能使用, 而矩阵由于行和列没有归属关系, 向量中的方法则面临作用方向问题, 因此往往在必要时要加上row或者col来标明作用方向。

练习：尝试写代码做出如下带名称的矩阵，并使用名称调用坐标为（南京, 上海），即从南京运往上海的运量数据。

	南京	上海
南京	0	300
上海	500	0

数据探索

快速认识某个数据集需要从几个角度入手：

1. 数据量多大
2. 数据的构成结构
3. 数据的类型
4. 数值范围

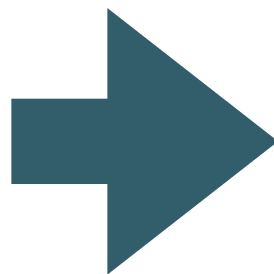
练习：尝试了解以下几个内置数据集

iris

cars

mtcars

LifeCycleSavings



```
> summary(iris)
```

```
> str(iris)
```

```
> class(iris)
```

```
> typeof(iris$Sepal.Length)
```

```
> head(iris)
```

```
> tail(iris)
```

可视化观察数据

`plot(x,y,...)` 可视化的基础函数，基本功能是将x,y作为坐标映射到画板的坐标系中

`plot`是通用型函数，能够识别不同数据类型的处理方法，即许多变量类型有一个对应的方法，如下图所示

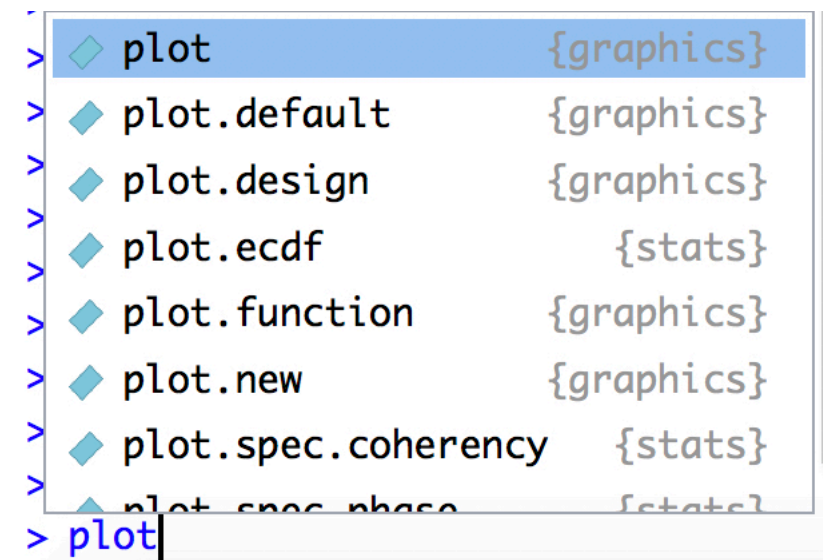
尝试对向量x进行可视化处理

```
> x  
[1] 1 2 3 9 8 7 6 5 3 4 5 6 7
```

```
> plot(x)
```

内置数据集cars是一个包含两个变量的数据框，尝试使用`plot`作图

```
> plot(cars)
```



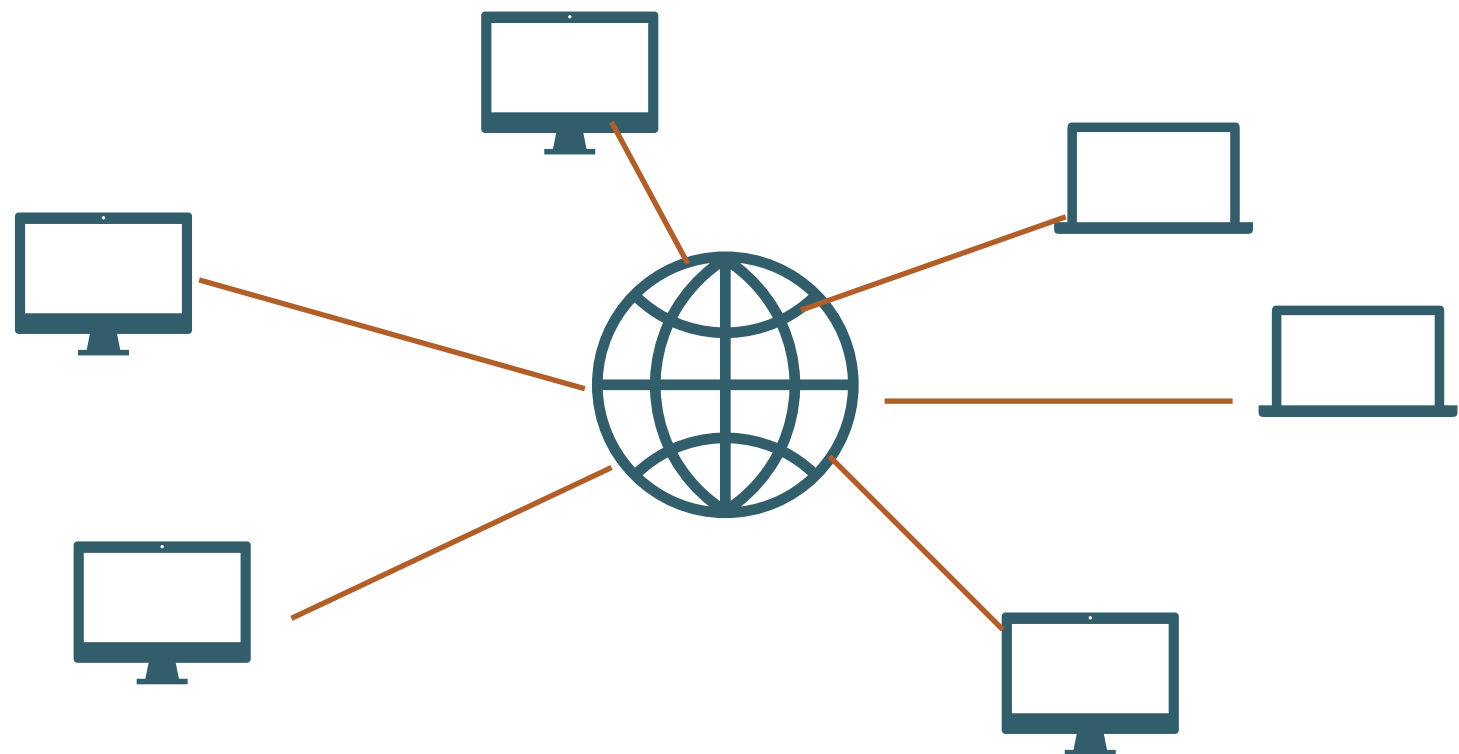
A screenshot of the R help documentation for the `plot` function. The table lists various methods for different data types, categorized by their class in curly braces.

<code>plot</code>	<code>{graphics}</code>
<code>plot.default</code>	<code>{graphics}</code>
<code>plot.design</code>	<code>{graphics}</code>
<code>plot.ecdf</code>	<code>{stats}</code>
<code>plot.function</code>	<code>{graphics}</code>
<code>plot.new</code>	<code>{graphics}</code>
<code>plot.spec.coherency</code>	<code>{stats}</code>
<code>plot.spec.phase</code>	<code>{stats}</code>

工具包的使用

Package是R的拓展工具包，即各行各业的专家将自己编写的工具代码封装在一个相对封闭且可以移植的包中，在R平台中需要的时候动态加载。

Rcran平台维护了一套规格极严格且开源的在线package系统，使得R成为实际上的数据科学家标准语言，即数据科学发布最新算法、模型和工具的平台。



默认方式从R官方平台在线安装package

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

这仅是将package安装在了R的本地目录中，还需将它调入内存工作区才能使用

```
> library(foreign)
```

library函数将包加入当前全局环境之上，最近一个加载过的包之下。

尝试加载几个常用工具包

dplyr

reshape2

ggplot2

e1025

forecast

zoo

help除查询函数帮助之外也可以查询包的全部介绍

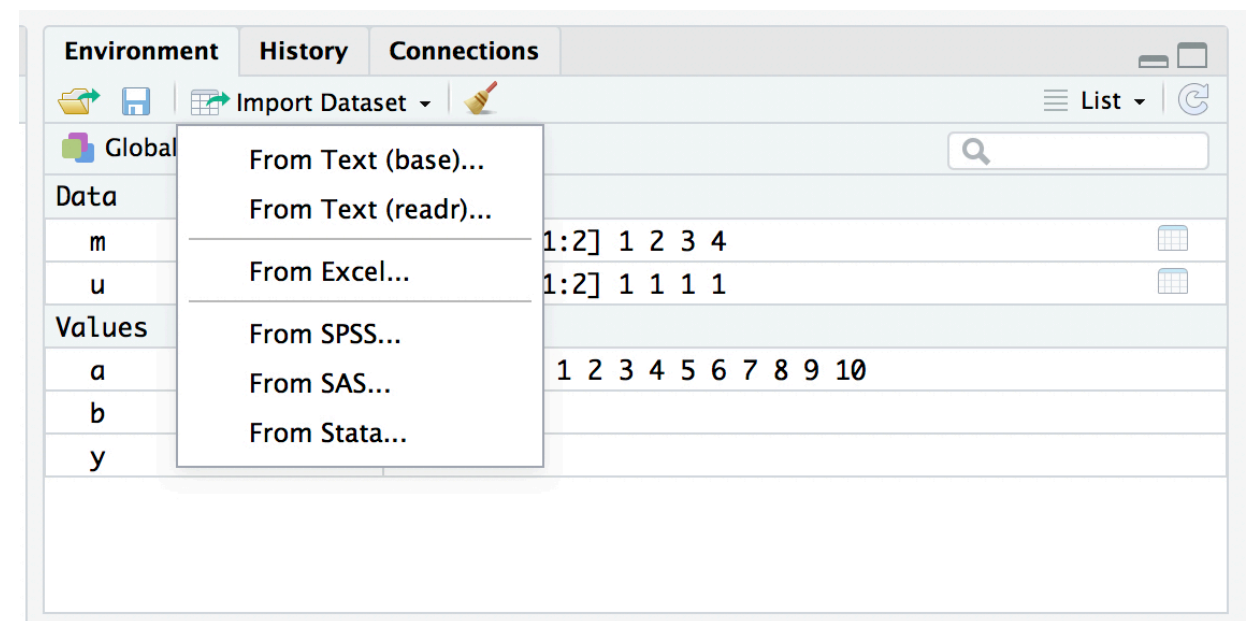
```
> help(package='dplyr')
```

数据集导入

学习完变量类型之后，我们已经有了装载大量数据的工具。实践中使用的数据大部分都是从现实中收集而来的，并非在程序中创造，因而导入数据便成了必不可少的一个环节。

导入数据的来源主要有：数据库系统、已保存成某种格式的数据文件、网络抓取等。

Rstudio将导入方法做成了图形界面操作，每一个导入过程都对应了一行r的代码



由于某些工具包需要用到java的运行环境，可能需要安装oracle公司的jdk

R自己的数据文件格式

.RDATA文件：默认环境便是使用这种文件保存，它能够保存多个数据集，甚至保存一个新的环境镜像

```
> save(m,file='mydata.rdata')      > save.image()
                                     > save.image(file="my.RData")
                                     > load('beer.rda')
```


.RDA文件：简洁的数据文件，一个文件保存一个数据集

```
> save(m,file='mydata.rda')
```

.RDA和.Rda文件可以直接使用load加载

```
> load('beer.rda')
```

结合数据库进行存取

 mydatabase.sqlite

```
> install.packages("RSQLite")
> library(RSQLite)

> con<-dbConnect(SQLite(),'data/mydatabase.sqlite')
> dbWriteTable(con,'table1',mtcars)
> dbListTables(con)
[1] "table1"
> dbGetQuery(con,'select * from table1')

> dbDisconnect(con)
```